Python-programmering för TI-84 Plus CE-T *Python Edition* Grafräknare

Version 84CE Bundle 5.6.0.

Lär dig mer om TI-teknologi genom onlinehjälpen på education.ti.com/eguide.

Viktig information

Utöver vad som uttryckligen anges i Licensen lämnar Texas Instruments inga garantier, vare sig uttryckliga eller underförstådda, inklusive men inte begränsade till underförstådda garantier om säljbarhet eller lämplighet för ett speciellt syfte, rörande program eller bokmaterial och gör endast sådant material tillgängligt på en "i befintligt skick"-basis. Under inga omständigheter kommer Texas Instruments att vara skyldigt för speciella skador, kollaterala skador, olycksfall eller följdskador i samband med eller uppkomna genom inköpet eller användningen av dessa material och det enda och exklusiva åtagande som åligger Texas Instruments, oavsett formen av åtgärd, skall inte överstiga det belopp som anges i licensen för programmet. Dessutom kommer inte Texas Instruments att vara förpliktigat i någon form av fordran från någon part rörande användningen av detta material.

"Python" och Pythons logotyper är varumärken eller registrerade varumärken som tillhör Python Software Foundation, och används av Texas Instruments Incorporated med tillstånd från denna Foundation.

Innehåll

Nyheter

Nyheter i programmeringsappen Python Programming App version 5.5.0

TI-84 Plus CE-T *Python Edition*

Python-programmering

TI-84 Plus CE-T *Python Edition*

- Stöder Python-programmering med Python App från 84CE Bundle version 5.6.0. Uppdatera till den senaste på education.ti.com/84cetupdate.
- Ta fram Python App med 2nd apps eller prgm när Python App är laddad.

Obs! Vad är din CE-räknare experience för TI-Python?

- TI-84 Plus CE-T *Python Edition* med 84CE Bundle version 5.6.0 eller högre

---

Överföra Python-program

Vid överföring av Python-program från en icke-TI-plattform till en TI-plattform ELLER från en TI-produkt till en annan:

- Python-program som använder språkets kärnfunktioner och standardbibliotek (math, random etc.) kan flyttas utan ändringar.
  Obs! Begränsning för listlängd är 100 element.
- Program som använder plattformsspecifika bibliotek - matplotlib (för dator), ti_plotlib/ti_system/ti_hub/etc. för TI-plattformar, kommer att behöva redigeringar innan de kan köras på en annan plattform.

Detta kan vara fallet även mellan TI-plattformar.

Nya funktioner och TI-Python-moduler

- Support för komplexa taltyper såsom a+bj.
  - Se menyn [Fns…] Types från Editor eller Shell.
- time-modul (tid)
- TI-moduler
  - ti_system
    - Återkalla OS-lista och OS-regressionsekvation i ett Python-program. Skapa listor i Python-program och lagra till OS-listvariabler. Begränsning för listlängd är 100 element.
  - ti_plotlib
    - Kör Python-program för att återge statistikdiagram och funktionsgrafer.
  - ti_hub
    - Skapa TI-Innovator™ Hub Python-program.
  - ti_rover
    - Styr TI-Innovator™ Rover med Python-programmering.

Skapa ”Nya” program-”Typer” med mallar

När ditt program kräver nödvändiga importförklaringar för moduler, använd fliken Types (Typer) när du skapar ett New (Nytt) program. Viktiga programrader klistras in i förväg i ditt nya program i Editorn. Detta är särskilt till hjälp för STEM-aktiviteter! Planeringsmetodmallen har support för den första erfarenheten när du skriver ett program med hjälp av ti_plotlib.

Argument Helpers (Argumenthjälpare) och Menu Screen Hints (Menyskärmtips)

En argumenthjälpare hjälper dig att välja rätt argument från en meny när metoder innehåller strängargument. Inget skrivande! Behöver inte leta efter rätt sträng!

Menyskärmtips tillhandahålls med argumentintervall, standardvärden, eller knapptryckningstips.

Python App knappsatsuppdateringar

math fortsätter att visa alla tillgängliga moduler.

2nd [i] (ovanför [.]) visar imaginära j för Python komplexa tal a+bj.

Se även: Knappsats

Programvaruinformation

TI Connect™ CE

Support för anslutningsbarhet och *.py <> PY AppVar-konvertering för TI-84 Plus CE-T *Python Edition*.

TI-SmartView™ CE-T

TI-84 Plus CE-T *Python Edition* emulator har support för Python App version 5.5.0

Exempelprogrammen HELLO, GRAPH och LINREGR laddas vid installationen och återställs.

Data Import Wizard (Dataimport-guide) konverterar riktigt formaterade *.csv-filer till räknarlistor för CE-emulatorn. Den här funktionen är till hjälp när du använder ti_system-modulen och externa data för Python-programmering.

- Om decimaltal är representerade med ett komma i *.csv-filen konverteras filen inte med Data Import Wizard (Dataimportguiden). Kontrollera nummerformateringen i din dators operativsystem och konvertera *.csv-filen till att använda decimalpunkt. CE-räknarens list- och matrisredigerare använder nummerformat såsom, till exempel 12.34 och inte 12,34.

Obs! För att köra TI-Innovator™ Hub- eller TI-Innovator™ Rover-program, skicka programmen till räknaren med TI Connect™ CE. Avsluta Python App innan en Emulatorutforskare överför till datorn och sedan till räknaren.
TI-Innovator™ Hub- och TI-Innovator™ Rover-program kan inte köras från TI-SmartView™ CE-T.

För mer information om nya och uppdaterade funktioner, gå till education.ti.com/84cetupdate.

Python App

Se följande för att använda, navigera och köra Python App.

- Använda Python App
- Python App-navigering
- Exempel på aktivitet

Använda Python App

Python App är tillgänglig för TI-84 Plus CE-T *Python Edition*. Informationen i denna eGuide är avsedd för TI-84 Plus CE-T *Python Edition* uppdaterad med den senaste CE Bundle.

När du kör Python App på din TI-84 Plus CE-T *Python Edition* för första gången, kan Appen instruera dig att uppdatera till den senaste CE Bundle för den senaste Python App.
Gå till education.ti.com/84cetupdate för att uppdatera din TI-84 Plus CE-T *Python Edition*.

Python App har en File Manager, en Editor för att skapa program och ett Shell-gränssnitt för att köra program och växelverka med Python-tolken. Python-program lagrade eller skapade som Python AppVars exekveras från RAM. Du kan arkivera Python AppVars via operativsystemets (OS) minneshanteringsskärm för att hjälpa till med lagringshantering av Python-filer.

Obs! Om din räknare är TI-84 Plus CE-T, gå till education.ti.com/84cetupdate för att hitta den senaste informationen för din CE-räknare.

Python App-navigering

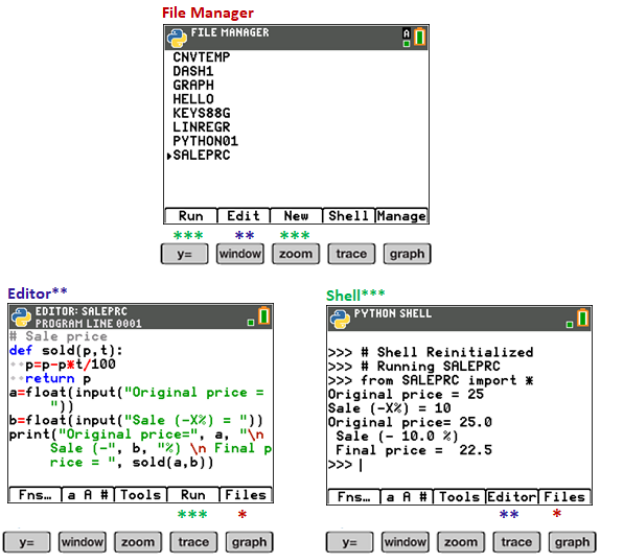
Använd kortkommandona på skärmen i Appen för att navigera mellan de olika arbetsområdena i Python App. På bilden anger etiketterna på genvägsflikarna:

* Navigering till File Manager [Files]

** Navigering till Editor [Edit] eller [Editor]

*** Navigering till Shell [Shell]

Öppna flikar på skärmen med hjälp av den grafiska knappraden omedelbart under skärmen. Se även Knappsats. Menyerna Editor>Tools och Shell>Tools innehåller också navigeringsåtgärder.



Exempel på aktivitet

Använd givet aktivitetsexempel som en mall för att bekanta dig med de olika arbetsområdena i Python App.

- Skapa ett nytt program från File Manager
- Skriv programmet i Editor
- Exekvera programmet i Shell i Python App.

För mer information om Python-programmering på din CE-T-räknare, se resurserna för TI-84 Plus CE-T *Python Edition*.

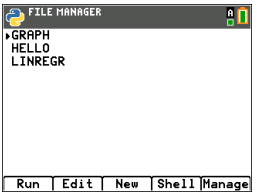
Komma igång:

- Kör Python App.

Obs! De faktiska skärmarna kan variera något från de bilder som visas.

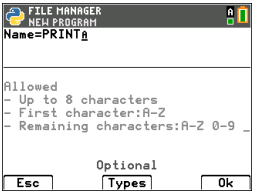
Ange ett nytt programnamn från File Manager (Filhanteraren).

- Tryck på [zoom] ([New]) (Nytt) för att skapa ett nytt program.



Inmatning av nytt filnamn

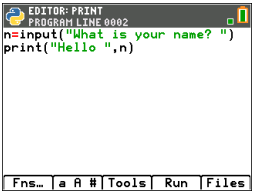
- Exempelprogrammet ska heta "PRINT". Ange programmets namn och tryck på [graph] ([Ok]).
- Observera att markören är i ALPHA lock-läge. Ange alltid ett programnamn enligt de givna kraven på skärmen.



Tips: Om markören inte är i ALPHA lock-läge, tryck på [2nd] [alpha] [alpha] för versaler.

Mata in programmet så som visas.

Tips: I appen finns också snabbinmatning! Titta alltid på markörens tillstånd medan du matar in ditt program!



Alfabetiska tecken på knappsatsen  	[alpha] växlar inmatningsmarkörens tillstånd i Editor och Shell. _ non-alpha a alpha med gemener A alpha med versaler
Var är likhetstecknet?	Tryck på [sto→] när

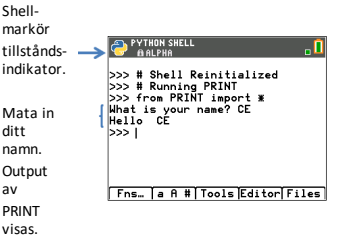
	markören är _. 
Var hittar man dessa? input() print()	[Fns...] I/O 1:print() 2:input()
Var är det dubbla citattecknet?	alpha ["] 
Var finns (och)?	Använd knappsatsen när markören är _. 

Försök! [a A #] och 2nd [catalog] är också hjälpare för att åstadkomma snabb inmatning!

Exekvera programmet PRINT.

- Från Editor, tryck på trace ([Run]) (Kör) för att exekvera ditt program i Shell.
- Ange ditt namn vid prompten ”What is your name?”.
- Output visar ”HELLO” med ditt namn.

Obs! Vid Shell-prompten >>>, kan du utföra ett kommando, såsom 2+3. Om du använder någon metod från math, random eller andra tillgängliga moduler, förvissa dig om att först ha med en import modul-sats, precis som i alla Python kodningsmiljöer.



Färdigställa en Python-session med dina program

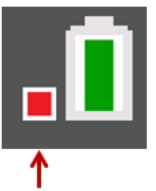
När Python App startas, synkroniseras CE-räknarens anslutning till TI-Python experience för din aktuella Python-session. Du ser en lista med dina program i RAM och dynamiska moduler medan de synkroniseras till Python experience.

När Python-sessionen är etablerad innehåller statusfältet en grön, kvadratisk indikator nära batteriikonen som anger att Python- sessionen är redo att användas. Om indikatorn är röd, vänta då tills indikatorn ändras tillbaka till grön och Python experience är tillgängligt igen.

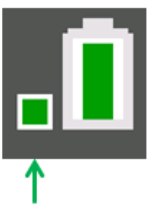
Du kan få se en uppdatering av Python-listan, tillsammans med programsynkroniseringen när du startar Python App efter den senaste uppdateringen av din
TI-84 Plus CE-T *Python Edition* från education.ti.com/84cetupdate.

Koppla bort och återansluta Python App

När Python App är aktiv, innehåller statusfältet en indikator som anger om Python är redo för användning eller inte. Innan anslutningen har etablerats kan det hända att CE-räknarens knappsats inte reagerar. Det bästa sättet är att ha koll på statusfältets anslutningsindikator medan du är i din Python-session.



Python inte redo



Python redo

Skärminfångningar

Använda TI Connect™ CE på education.ti.com/84cetupdate, skärminfångningar från valfri Python App är tillåtna.

Python arbetsområden

Python App har tre arbetsområden för ditt arbete med Python-programmering.

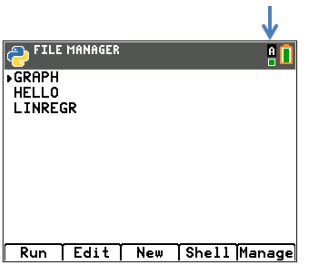
- File Manager
- Editor
- Shell

Python File Manager

File Manager (Filhanteraren) listar de Python AppVars som finns tillgängliga i RAM på din räknare. Du kan såväl skapa, redigera och köra program som navigera till Shell-gränssnittet.

I alpha-läge trycker du på valfri bokstav på knappsatsen för att hoppa till program som börjar med den bokstaven.

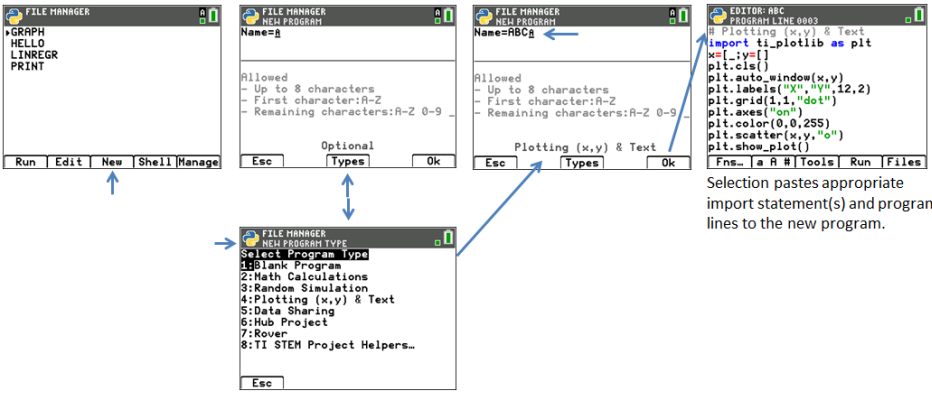
Tryck på alpha om det behövs när **A**-indikatorn inte finns i statusfältet.



File Manager kortkommandon och menyer		
Menyer	**Tangenttryckning**	**Beskrivning**
[Run]	y=	Välj ett program med ▲ eller ▼. Välj sedan [Run] (Kör) för att exekvera ditt program.
[Edit]	window	Välj ett program med ▲ eller ▼. Välj sedan [Edit] (Redigera) för att visa programmet i Editorn och redigera ditt program.
[New]	zoom	Välj [New] (Nytt) för att ange ett nytt programnamn och fortsätta till Editorn för att mata in ditt nya program. På skärmen [New] (Nytt), välj [Types] (Typer) (tryck på [zoom]), för att välja en typ av program. Genom att välja en typ av program kommer en mall med import-kommandon och ofta använda funktioner och metoder att klistras in i ditt nya program för den aktiviteten.
[Shell]	trace	Välj [Shell] för att visa Shell-prompten (Python-tolken). Shell-gränssnittet kommer att vara i det aktuella tillståndet.
[Manage]	graph	Välj [Manage] (Hantera) för att: • Visa versionsnummer. • Reproducera, ta bort eller byta namn på ett valt program.

File Manager kortkommandon och menyer		
Menyer	**Tangenttryckning**	**Beskrivning**
		• Visa skärmen About (Om). • Avsluta appen. Använd även 2nd [quit]

Skapa ett nytt program med programtypsmallar

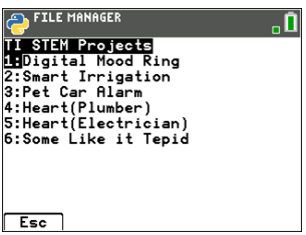


Selection pastes appropriate import statement(s) and program lines to the new program.

- Select [Types] to display Select Program Type menu.
- Import(s) displays on status bar.

Skapa ett nytt STEM-aktivitetsprogram med mallar

När TISTEMEN AppVar är laddad i Arkiv, visas menyobjektet ”TI STEM Project Helpers…” i menyn Select Program Type. Välj STEM-aktivitetsmall efter behov som hjälp för att påbörja ett nytt STEM-program.



Python Editor

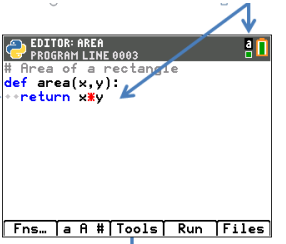
Python Editor visas från ett valt program i File Manager (Filhanteraren) eller från Shell-gränssnittet. Editorn visar nyckelord, operatorer, kommentarer, strängar och indrag i färg. Snabbinklistring av vanliga Python nyckelord och funktioner finns tillgängliga såväl som direkt inmatning med knappsatsen och [a A #]-teckeninmatning. När du klistrar in kodblock såsom if.. elif.. else, erbjuder Editorn automatiska indrag vilka kan ändras efter behov allteftersom du skriver ditt program.

Markören är alltid i infoga-läge. Använd 2nd och alpha för att växla markör. Markörtillstånd är numeriskt, a och A. [del] uppför sig som bakåtsteg och borttagning av ett tecken.

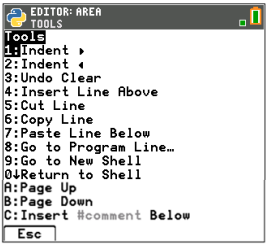
Programradens plats för markören.

Automatisk indragning av kodblock.

Grå punkter som en visuell indikering av indragna rader.



Användbara verktyg för redigering och arbete i Shell-gränssnittet. Fullständig beskrivning nedan.



Python Editor kortkommandon och menyer		
Menyer	**Tangenttryckning**	**Beskrivning**
[Fns…]	y=	Välj [Fns…] för att ta fram menyer med ofta använda funktioner, nyckelord och operatorer. Ta även fram utvalt innehåll från math- och random-modulerna. **Obs!** 2nd [catalog] är också bra för snabbinklistring.

<table>
<tr><th colspan="3">Python Editor kortkommandon och menyer</th></tr>
<tr><th>Menyer</th><th>Tangenttryckning</th><th>Beskrivning</th></tr>
<tr><td>[a A #]</td><td>window</td><td>Välj [a A #] för att ta fram en teckenpalett som ett alternativt sätt att mata in många tecken.</td></tr>
<tr><td>[Tools]</td><td>zoom</td><td>Välj [Tools] (Verktyg) för att ta fram funktioner som hjälper dig vid redigering eller vid din interaktion med Shell-gränssnittet.
<table>
<tr><td>1: Indent ▶</td><td>Gör indrag på programraden, markören flyttas åt höger till radens första tecken.</td></tr>
<tr><td>2: Indent ◀</td><td>Reducerar indraget av programraden åt vänster. Markören flyttas till radens första tecken.</td></tr>
<tr><td>3: Undo Clear</td><td>Klistrar in den senast rensade raden på en ny rad nedanför programraden med markören. Markören visas vid slutet av den inklistrade raden.</td></tr>
<tr><td>4: Insert Line Above</td><td>Infogar en rad ovanför programraden med markören. Raden blir indragen och visar indragspunkter när det är lämpligt.</td></tr>
<tr><td>5: Cut Line</td><td>Aktuell programrad med markör klipps ut. Markören visas på programraden nedanför den urklippta raden.</td></tr>
<tr><td>6: Copy Line</td><td>Kopierar aktuell programrad med markör. En kopierad programrad kan klistras in i Shell-prompten. Se Shell nedan.</td></tr>
<tr><td>7: Paste Line Below</td><td>Klistrar in den senast lagrade programraden på raden nedanför markörens position.</td></tr>
<tr><td>8: Go to Program Line...</td><td>Visar markören i början av den specificerade programraden.</td></tr>
<tr><td>9: Go to New Shell</td><td>Visar återinitialiserat Shell-gränssnitt.</td></tr>
<tr><td>0: Return to Shell</td><td>Visar Shell i aktuellt tillstånd.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="3">Python Editor kortkommandon och menyer</th></tr>
<tr><th>Menyer</th><th>Tangenttryckning</th><th>Beskrivning</th></tr>
<tr><td></td><td></td><td><table>
<tr><td>A: Page up</td><td>Visar 11 programrader ovanför aktuell markörposition efter tillgänglighet.</td></tr>
<tr><td>B: Page Down</td><td>Visar 11 programrader nedanför aktuell markörposition efter tillgänglighet.</td></tr>
<tr><td>C: Insert #comment Below</td><td>Infogar # på en ny rad nedanför markörpositionen.</td></tr>
</table></td></tr>
<tr><td>[Run]</td><td>trace</td><td>Välj [Run] (Kör) för att exekvera ditt program.</td></tr>
<tr><td>[Files]</td><td>graph</td><td>Välj [Files] för att visa File Manager.</td></tr>
</table>

Python Shell

Python Shell är konsolen där du kan växelverka med Python-tolken eller köra dina Python-program. Såväl snabbinklistring av vanliga Python nyckelord och funktioner är tillgängliga som direkt inmatning med knappsatsen och [a A #]-teckeninmatning. Shell-prompten kan användas till att testa en rad med kod inklistrad från Editorn. Flera rader av kod kan också matas in och köras vid en Shell-prompt >>>.

Shell-markörens tillstånds-indikator.

Shell återinitialiserar när ett nytt program exekveras.

Användbara verktyg för att arbeta i Shell-gränssnittet.
Se detaljer nedan.

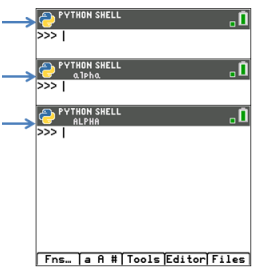
Shell-markörens tillstånd.

non-alpha

2nd alpha om du behöver växla

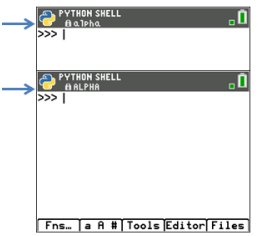
alpha alpha

alpha igen ALPHA



2nd alpha lås alpha

alpha igen lås ALPHA



<table>
<tr><th colspan="3">Python Shell kortkommandon och menyer</th></tr>
<tr><th>Menyer</th><th>Tangenttryckning</th><th>Beskrivning</th></tr>
<tr><td>[Fns…]</td><td>y=</td><td>Välj [Fns…] för att ta fram menyer med ofta använda funktioner, nyckelord och operatorer. Ta även fram utvalt innehåll från math- och random-modulerna.
Obs! 2nd [catalog] är också bra för snabbinklistring.</td></tr>
<tr><td>[a A #]</td><td>window</td><td>Välj [a A #] för att ta fram en teckenpalett som ett alternativt sätt att mata in många tecken.</td></tr>
<tr><td>[Tools]</td><td>zoom</td><td>Välj [Tools] för att visa följande menyobjekt.
<table>
<tr><td>1: Rerun last program</td><td>Kör programmet som senast kördes i Shell igen.</td></tr>
<tr><td>2: Run…</td><td>Visar en lista på Python-program som finns tillgängliga att köra i Shell.</td></tr>
<tr><td>3: Paste from Editor</td><td>Klistrar in den senast kopierade programraden från Editorn till Shell-prompten.</td></tr>
<tr><td>4: Vars…</td><td>Visar vars från det senaste programmet som kördes. Visar inte programdefinierade vars från ett importerat program.</td></tr>
<tr><td>5: Clear Screen</td><td>Rensar Shell-skärmen. Återinitialiserar inte ett nytt Shell.</td></tr>
<tr><td>6: New Shell</td><td>Återinitialiserar ett nytt Shell.</td></tr>
<tr><td>7: Go to Program Line…</td><td>Visar Editorn från Shell med markören på den specificerade programraden.</td></tr>
<tr><td>8: Last Entry>>> ▲ ▼</td><td>Visar upp till de 8 senaste inmatningarna vid Shell-prompten under en Shell-session.</td></tr>
<tr><td>9: View History 2nd ▲ 2nd ▼</td><td>Bläddra Shell-skärmen för att visa upp till de sista 60 raderna utmatning i Shell under en Shell-session.</td></tr>
<tr><td>0: Tab Complete 2nd [enter]</td><td>Visar namnen på variablerna och funktionerna som finns tillgängliga för åtkomst i den aktuella Shell-sessionen.
När en bokstav för en tillgänglig</td></tr>
</table></td></tr>
</table>

Python Shell kortkommandon och menyer			
Menyer	**Tangenttryckning**	**Beskrivning**	
			variabel eller funktion skrivs in, tryck på 2nd [enter] för att automatiskt komplettera namnet om en match finns tillgänglig i den aktuella Shell-sessionen.
		A: from PROGRAM import *…	Vid första exekveringen i en Shell-session, körs PROGRAM och vars kan bara visas med Tab Complete. När det exekveras igen i samma Shell-session, framstår exekveringen som ingen exekvering. Det här kommandot kan också klistras in från 2nd [catalog].
[Editor]	trace	Välj [Editor] för att visa Editorn med de senaste programmen i Editor. Om Editorn är tom, kan du visa File Manager.	
[Files]	graph	Välj [Files] för att visa File Manager.	

Obs!

- För att avbryta ett Python-program som körs i Shell-gränssnittet, t.ex. när ett program är i en kontinuerlig loop, tryck på on. Tryck på **[Tools]** (zoom) > **6:New Shell** som en alternativ metod till att stoppa ett program som körs.
- När du använder ti_plotlib-modulen för att plotta i diagramområdet i Shell, tryck på clear för att rensa diagrammet och återgå till Shell-prompten.

Exekveringsfel: Gå till programrad med Shell >Tools

TI-Python experience visar Python felmeddelanden i Shell-gränssnittet när kod exekveras. Om ett fel visas när ett program exekveras, visas ett programradsnummer. Använd **Shell>Tools 7:Go to Program Line…** Ange radnumret och tryck på **[OK]**. Markören visas på första tecknet på denna programrad i Editor. Programradens nummer visas i den andra raden i statusfältet i Editor.

Inmatning - knappsats, katalog, teckenuppsättning och menyer

Tips för snabb inmatning

- Knappsats
- Katalog
- [a A #] Teckenuppsättning
- [Fns…] Menyer

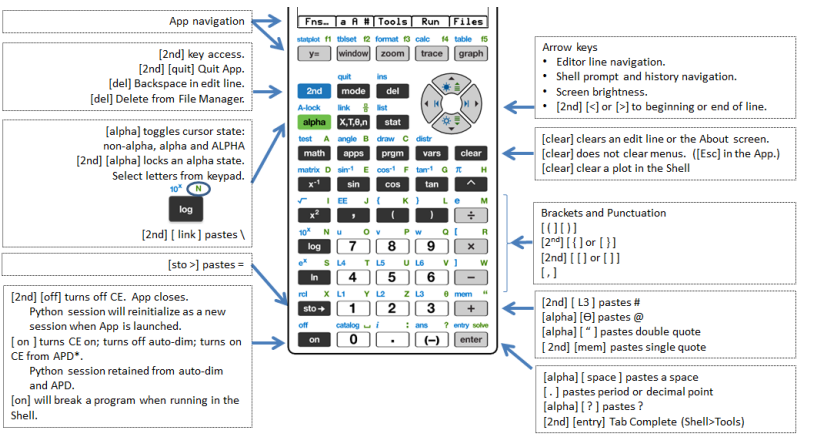
Använda knappsatsen, katalogen, [a A #] och Fns…-menyer

När du matar in kod i Editorn eller i Shell, använd följande inmatningsmetoder för att snabbt klistra in på redigeringsraden.

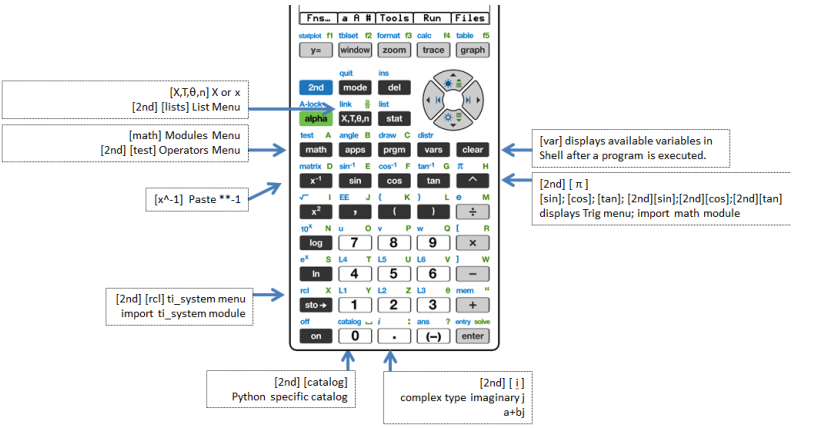
Knappsats

När Python App är aktiv, är knappsatsen utformad att klistra in lämpliga Python-operationer eller öppna menyer utformade för enkel inmatning av funktioner, nyckelord, metoder, operatorer, etc. Tryck på 2nd och alpha ger åtkomst till andra- och tredje-funktioner på en tangent, precis som i räknarens operativsystem.

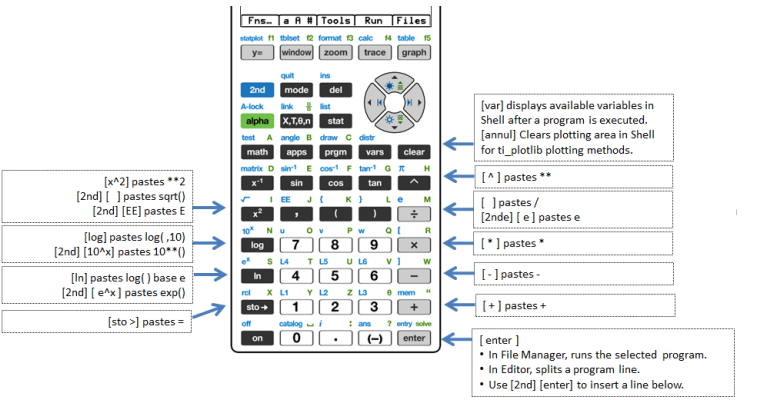
Python App navigering, redigering och speciella tecken efter knappsatsrader



Python App specifika tangenttryckningar för menyer och funktioner efter knappsatsrader



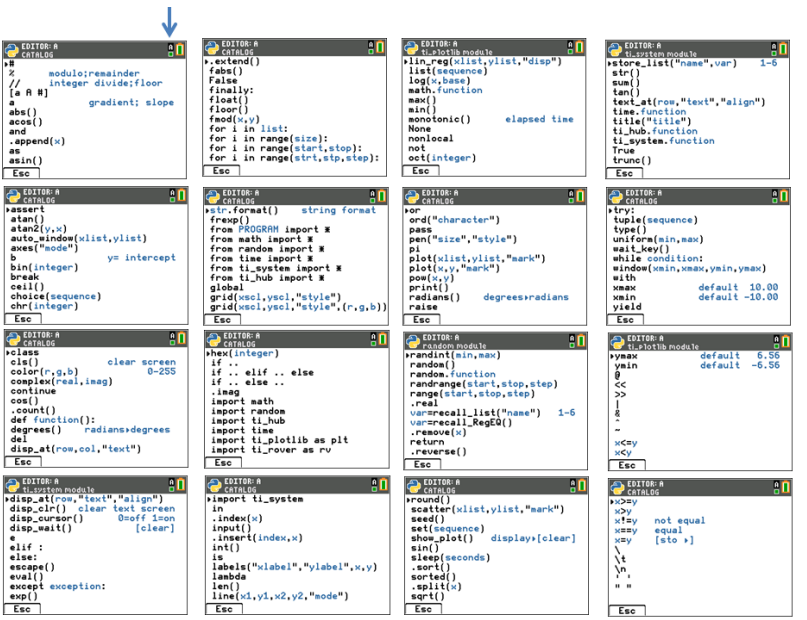
Python App specifika tangenttryckningar för menyer och funktioner efter knappsatsrader (fortsättning)



Katalog

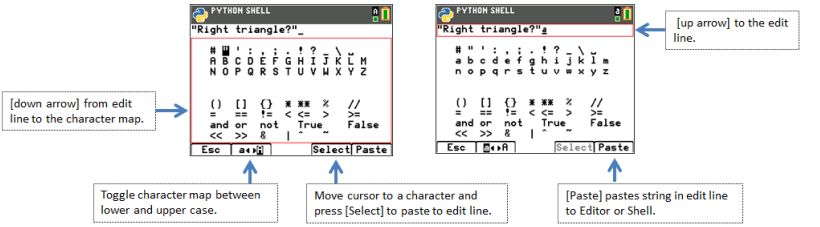
När Python App är aktiv, visar [2nd] [catalog] en lista på ofta använda avgränsare, nyckelord, funktioner och operatorer för att snabbt klistra in på en redigeringsrad. [2nd] [catalog] är endast tillgängligt i Editor och Shell-gränssnittet. För en mer detaljerad beskrivning av varje katalogobjekt, se Referensguiden. Från högst upp i menyn Catalog, tryck på ▲ för en cirkulär navigering av katalogen.

När du är i katalogen, välj [alpha] och en bokstavstangent för att visa listan som börjar med den bokstaven.



[a A #] Teckenuppsättning

[a A #] genvägsflik till en teckenpalett är en bekväm funktion för att mata in strängar i Editor eller Shell-gränssnittet.



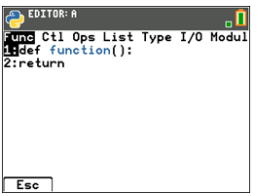
Obs! När markörens fokus är i redigeringsraden för [a A #], är valda tangenter på knappsatsen inte tillgängliga. När fokus är i teckenuppsättningen, är knappsatsen begränsad.

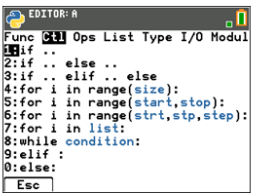
[Fns…] Menyer

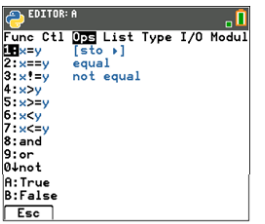
[Fns…] genvägsflikar visar menyer med ofta använda funktioner, nyckelord och operatorer i Python. Menyerna ger även åtkomst till utvalda funktioner och konstanter från math- och random-modulerna. Visst kan du mata in tecken ett i taget från knappsatsen men med dessa menyer går det snabbare att klistra in i Editor eller Shell. Tryck på [Fns…] när du är i Editor eller Shell. Se även Katalog och Knappsats för alternativa inmatningsmetoder.

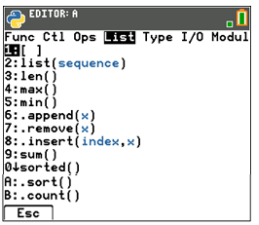
Funktion- och modul-undermenyer

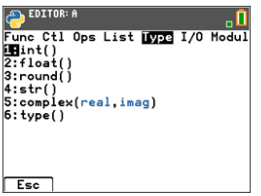
Built-in, operatorer och nyckelord

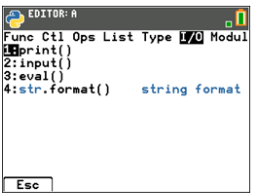










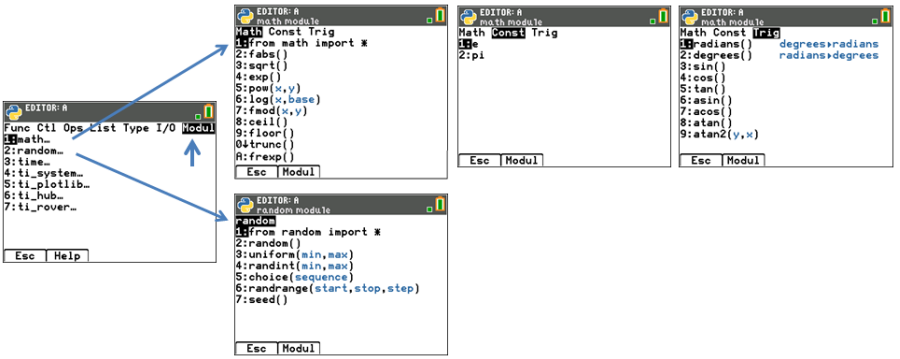


Modul-undermenyer

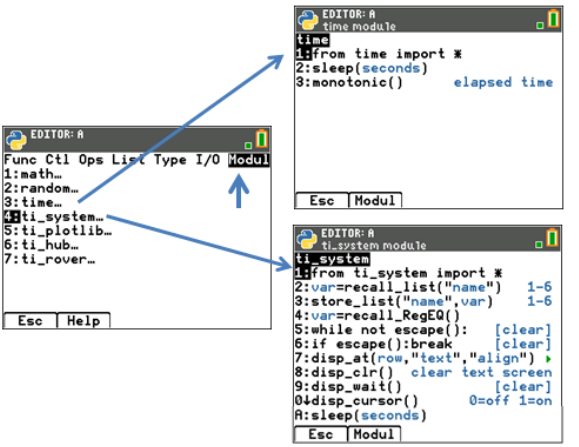
När du använder en Python-funktion eller -konstant från en modul, använd alltid ett import-kommando för ange platsen i modulen för funktionen, metoden eller konstanten.

Se Vad är Python-programmering experience?

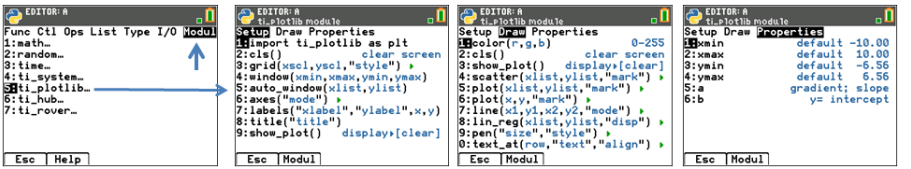
[Fns…]>Modul: math- och random-moduler



[Fns…]>Modul: time- och ti_system-moduler



[Fns…]>Modul: ti_plotlib



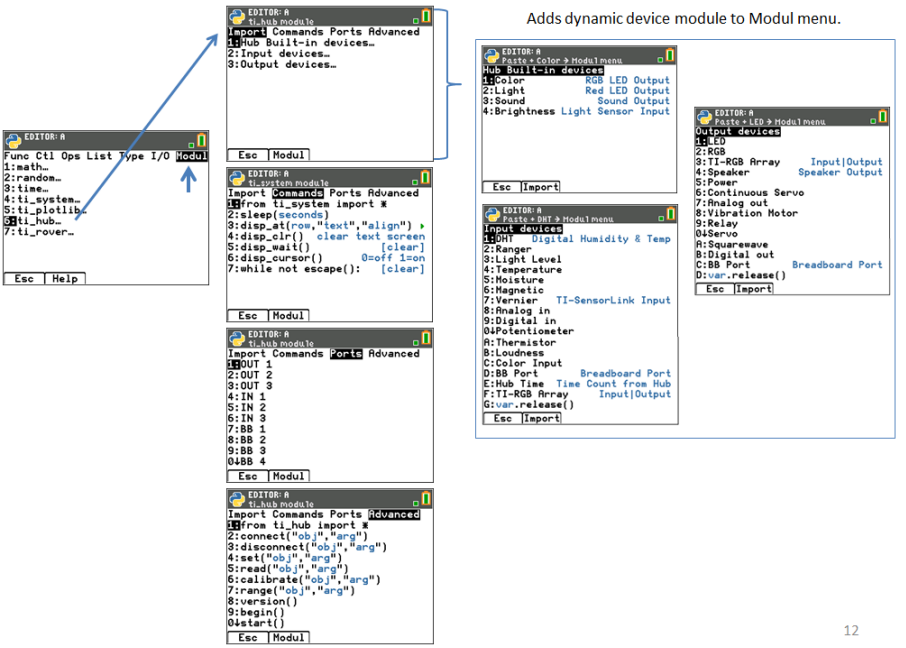
Viktigt om plottning:

- Ordningen på programrader för plottning måste följa ordningen som i Setup-menyn för att garantera förväntade resultat.
- Plottning visas när plt.show_plot() exekveras vid slutet av plottningsobjekten i ett program. För att rensa plottningsområdet i Shell, tryck på [clear].
- Kör du ett andra program som antar att standardvärden är inställda i samma Shell-miljö blir resultat i allmänhet ett oväntat beteende för färg eller andra inställningar för standardargument. Redigera program med förväntade argumentvärden eller återinitialisera Shell innan du kör något annat plottningsprogram.

[Fns…]>Modul: ti_hub module

ti_hub-metoder är inte listade i Katalog och således inte listade i referensguiden. Använd informationen på skärmen i menyerna för detaljer om argument och arguments standardvärden eller tillåtna värden. Mer information om Python-programmering för
TI-Innovator™ Hub och TI-Innovator™ Rover kommer att finnas tillgänglig på education.ti.com.

Obs! TI-Innovator™ Hub ska vara ansluten när du kör dina Python-program.



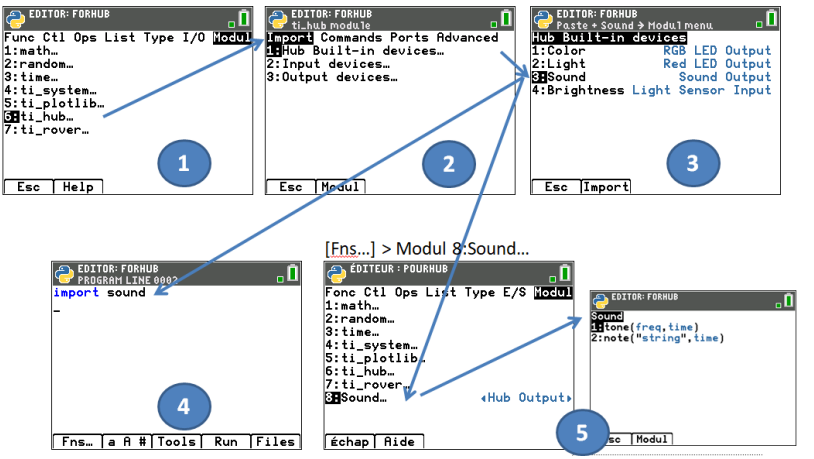
ti_hub module – Lägg till import till Editor och lägg till ti_hub sensor module till Modul-menyn

Skärmexempel: Importera ljud

För att importera TI-Innovator™ sensormetoder till ditt Python-program, från Editor,

1. Välj **[Fns…] > Modul 6:ti_hub**
2. Välj menyn ti_hub Import. Välj en sensortyp från Built-in, Input och Output.
3. Välj en sensor.
4. Ett import-kommando klistrar in till Editorn och sensormodulen kommer att finnas tillgänglig i **[Fns…] > Modul** när du återgår till den menyn från ditt program.
5. Välj **[Fns…] > Modul 8:Sound…** för att klistra in lämpliga metoder för den här sensorn.

[Fns…]>Modul 6:ti_hub



Obs! Brightns är ett "built-in" objekt på TI-Innovator Hub.

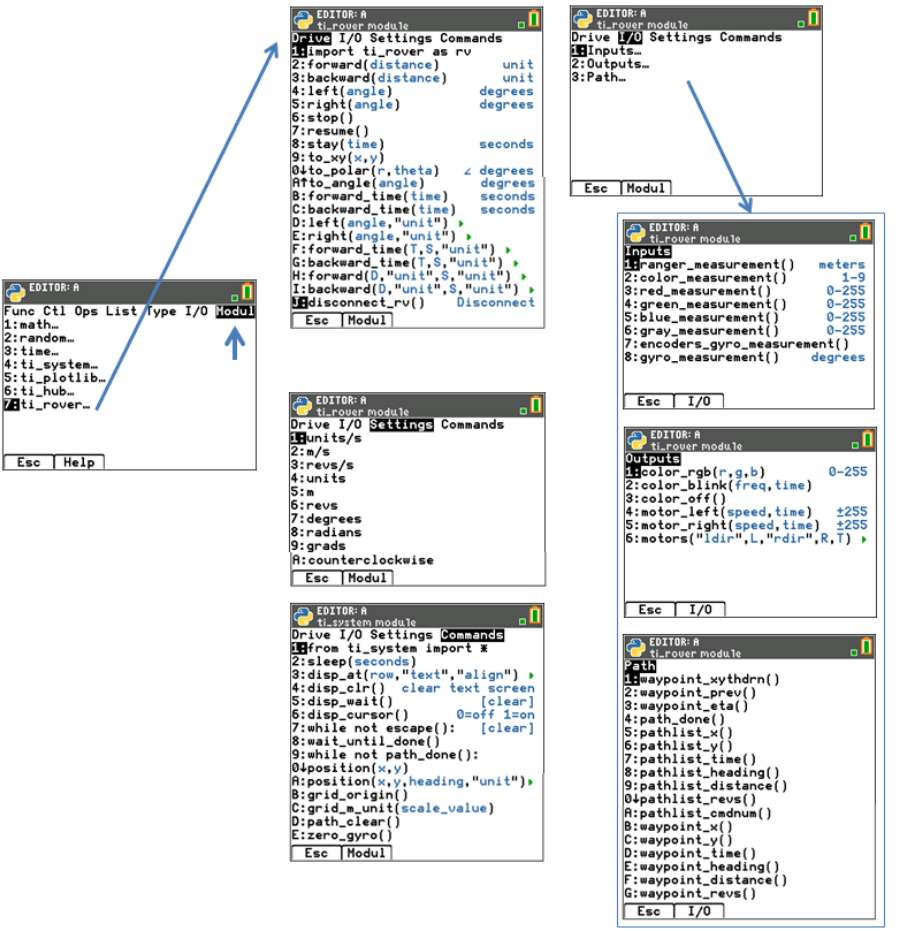
När du använder kommandot 'import brightns', ange 'brightns.range(0,100)' för att garantera rätt standardintervall i början på programmet.

Exempel:

```
import brightns
brightns.range(0,100)
b=brightns.measurement()
print(b)
```

[Fns...]>Modul ti_rover-modul

ti_rover-metoder är inte listade i Katalog och således inte listade i referensguiden. Använd informationen på skärmen i menyerna för detaljer om argument och arguments standardvärden eller tillåtna värden. Mer information om Python-programmering för
TI-Innovator™ Hub och TI-Innovator™ Rover kommer att finnas tillgänglig på education.ti.com.



Obs!

- I TI-Python-programmering, behöver du inte inkludera metoder för att ansluta och koppla bort TI-Innovator™ Rover. TI-Innovator™ Rover Python-metoder hanterar

anslutning och bortkoppling utan några extra metoder. Det här är lite annorlunda från att programmera TI-Innovator™ Rover i TI-Basic.

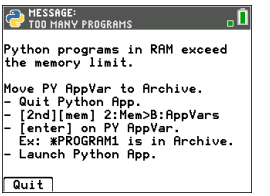
- rv.stop() exekverar som en paus och sedan resume som fortsätter med Roverns rörelser i kön. Om ett annat rörelsekommando exekveras efter rv.stop(), då rensas rörelsekön. Det här är återigen lite annorlunda från att programmera TI-Innovator™ Rover i TI-Basic.

Python App-meddelanden

Det finns många meddelanden som kan visas medan du är i en Python-session. Vissa utvalda meddelanden visas i tabellen. Följ instruktionerna på skärmen och navigera med [Quit] (Avsluta), [Esc] eller [Ok] efter behov.

Hantering av minne

Det tillgängliga minnet för Python experience kommer att vara maximalt 100 Python-program (PY AppVars) eller 50K minne. Modulerna som ingår med appen i den här Python-releasen delar samma utrymme med alla filer.



Använd [2nd] [quit] för att avsluta Appen.

Du blir tillfrågad om du verkligen vill avsluta Appen. När du avslutar Appen stoppas din Python-session. När du kör Python App igen, synkroniseras dina Python AppVar-program och moduler. Shell återinitialiseras.

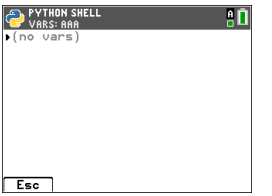
I File Manager, trycker du på del på ett valt Python-program eller du väljer från **File Manager>Manage 2:Delete Program...**.

Du får se en dialog för att ta bort eller escape tillbaka till File Manager.

Du försöker skapa ett nytt eller duplicera ett Python-program som redan finns på din CE-räknare, antingen i RAM eller Arkiv eller är inaktiverat för provläge. Ange ett annat namn.

Du försöker navigera från Shell till Editorn men Editorn är tom. Välj ett annat lämpligt alternativ för ditt arbete.

När du exekverar ett Python-program, listas definierade variabler från det senast exekverade programmet i menyn **Shell>Tools> 4:Vars…** för att användas och är tillgängliga att användas i Shell. Om inga variabler visas måste du kanske köra ditt program igen.



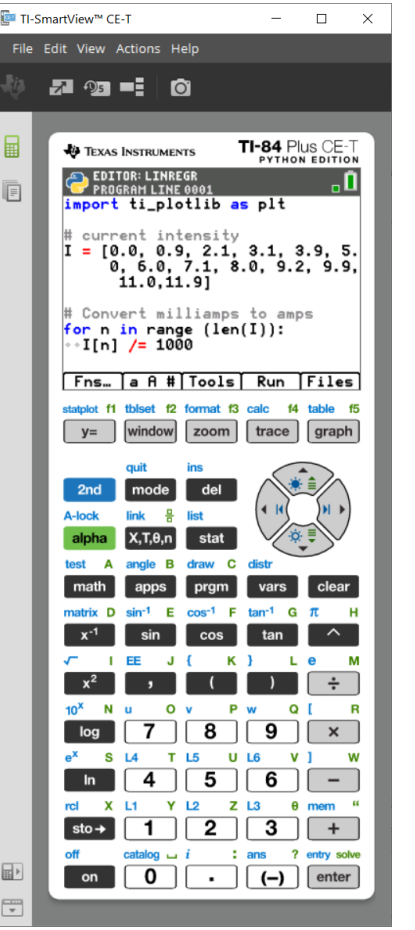
Använda TI-SmartView™ CE-T och Python Experience

Denna guidebok förutsätter den senaste uppdateringen av TI-SmartView™ CE-T. Uppdatera till senaste TI-SmartView™ CE-T på education.ti.com/84cetupdate.

Uppdateringen inbegriper senaste TI-84 Plus CE-T *Python Edition* emulator OS som kör den senaste Python App. De uppdaterade modulerna time, ti_system, ti_plotlib, ti_rover* och ti_hub* är inkluderade.

Kör Python App på TI-84 Plus CE-T *Python Edition*-emulatorn.

- Python App erbjuder
  - File Manager
  - Editor
  - Exekvering av dina Python-program i gränssnittet Shell*



Hub/Rover-program

- Skapa ti_hub/ti_rover Python-program i CE-emulatorn när Python App körs.

  * **Obs!** Det finns ingen anslutningsbarhet mellan TI-SmartView™ CE-T och TI-Innovator™ Hub eller TI-Innovator™ Rover. Program kan skapas och sedan köras på CE-T-räknaren.

- Avsluta Python App för att förbereda överföring av Python AppVar(s) från emulatorn. Emulatorn får inte vara ”upptagen” med att köra en App eller ett program för nästa steg.
- Byt till arbetsområdet Emulatorutforskare och skicka programmet/n till datorn.

– Använd TI Connect™ CE för att skicka Python AppVars från datorn till CE-T-räknaren för TI-Innovator™ Hub/TI-Innovator™ Rover experience.

Obs! För att avbryta ett Python-program som körs i Shell-gränssnittet, t.ex. när ett program är i en kontinuerlig loop, tryck på **[on]**. Tryck på **[Tools] [zoom] > 6:New Shell** som en alternativ metod till att stoppa ett program som körs.

Kom ihåg: För alla datorer/TI-Python experience: Efter att ha skapat ett Python-program i en utvecklingsmiljö för Python på datorn, validera att ditt program kan köras på räknaren/emulatorn TI-Python experience. Ändra programmet efter behov.

SmartPad CE App fjärrknappsats

– När den körs agerar SmartPad CE App på din anslutna CE-räknare som en fjärrknappsats, inklusive den särskilda *knappsats*-mappning som finns när Python App körs.

Arbetsområdet Emulatorutforskare

– Avsluta Python App så att emulatorn inte är aktiv när du utnyttjar alla funktioner hos arbetsområdet Emulatorutforskare.

– program.py < > PY AppVar konverteringar är tillåtna. Detta liknar TI Connect™ CE experience när program skickas till den anslutna CE-T-räknaren.

– När en fil, program.py, som skapats i en annan Python-miljö skickas, måste PY AppVar redigeras för att fungera som förväntat i TI-Python. Använd Python App Editor för att ändra efter behov för de unika modulerna såsom ti_plotlib, ti_system, ti_hub och ti_rover.

Data Import Wizard (Dataimport-guide)

– *.csv-filer med data, formaterade som det anges i guidens dialogruta, konverterar data till listvariabler i CE. Metoder i ti_system kan sedan användas för att dela listor mellan emulator CE OS och Python App. Den här funktionen liknar den i Data Import Wizard (Dataimport-guide) i TI Connect™ CE.

- Om decimaltal är representerade med ett komma i *.csv-filen konverteras filen inte med Data Import Wizard (Dataimportguiden). Kontrollera nummerformateringen i din dators operativsystem och konvertera *.csv-filen till att använda decimalpunkt. CE-räknarens list- och matrisredigerare använder nummerformat såsom, till exempel 12.34 och inte 12,34.

Använda TI Connect™ CE för att konvertera Python-program

Uppdatera till TI Connect™ CE för de senaste funktionerna inklusive konvertering av *.py-program till en PY AppVar som CE-räknarens filformat.

Se TI-84 Plus CE-T e-Guide för mer information om CE-räknaren, TI-SmartView™ CE-T och TI Connect CE.

Vad är Python-programmering experience?

TI-Python är baserat på CircuitPython, en variant av Python designat att passa i små mikrostyrenheter. Den ursprungliga implementeringen av CircuitPython har anpassats för användning av TI.

Den interna lagringen av tal för beräkningar i den här varianten av Circuit Python är med binära flyttal med begränsad precision och kan därför inte exakt representera alla möjliga decimala värden. Skillnaden från den faktiska representationen som uppstår vid lagringen av dessa värden kan leda till oväntade resultat i därpå följande beräkningar.

- **För flyttal** - Visar en precision med upp till 16 signifikanta siffror. Internt lagras värden med 53 bitars precision, vilket grovt räknat motsvarar 15-16 decimaler.
- **För heltal** - Storleken på heltal begränsas endast av det tillgängliga minnet då beräkningarna utförs.

Moduler som ingår i TI-84 Plus CE-T Python Edition

- Built-ins
- math module
- random module
- time
- ti_system
- ti_plotlib
- ti_hub
- ti_rover

Obs! Om du har befintliga Python-program som har skapats i andra Python-utvecklingsmiljöer, redigera ditt/dina program enligt TI-Python-lösningen. Moduler kan använda andra metoder, argument, och annan ordningsföljd för metoder i ett program jämfört med modulerna ti_system, ti_plotlib, ti_hub och ti_rover. I allmänhet ska du beakta kompatibiliteten för alla versioner av Python och Python-moduler.

Vid överföring av Python-program från en icke-TI-plattform till en TI-plattform ELLER från en TI-produkt till en annan:

- Python-program som använder språkets kärnfunktioner och standardbibliotek (math, random etc.) kan flyttas utan ändringar

  Obs! Begränsning för listlängd är 100 element.

- Program som använder plattformsspecifika bibliotek - matplotlib (för dator), ti_plotlib,
  ti_system, ti_hub, etc. för TI-plattformar, kommer att behöva redigeringar innan de kan köras på en annan plattform.
- Detta kan vara fallet även mellan TI-plattformar.

Precis som för alla versioner av Python, måste du inkludera import-kommandon såsom "from math import *", för att använda funktioner, metoder eller konstanter som finns i math-modulen. Till exempel för att utföra funktionen cos(), använd import för att importera math-modulen för användning.

Se KATALOG-lista.

Exempel:

```
>>>from math import *
>>>cos(0)
1.0
```

Alternativt exempel:

```
>>>import math
>>>math.cos(0)
1.0
```

Tillgängliga moduler kan visas i Shell-gränssnittet med följande kommando

```
>>> help("modules")
__main__ sys gc
random time array
math builtins collections
```

Innehållet i modulerna kan visas i Shell-gränssnittet såsom visas med ”import module” och ”dir(module)”.

Inte allt modulinnehåll dyker upp i snabbinklistringsmenyerna såsom [Fns…] eller [2nd] [catalog].

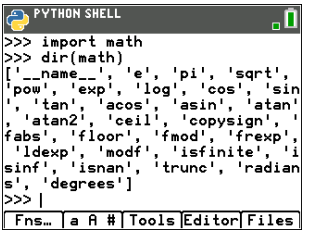
Innehåll för utvalda moduler och nyckelord

För en lista på modulerna som ingår i denna release, se:

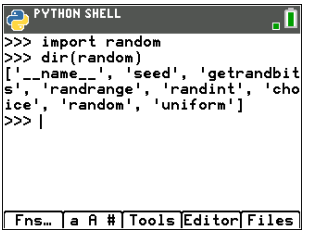
Bilaga: Utvalda TI-Python Built-in, nyckelord och modulinnehåll

Kom ihåg: För alla datorer/TI-Python experience: Efter att ha skapat ett Python-program på datorn, validera att ditt program kan köras på räknaren i TI-Python experience. Ändra programmet efter behov.

Dessa skärmar visar modulinnehållet för math och random.

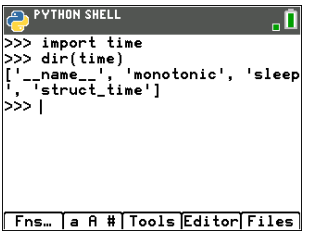


math module

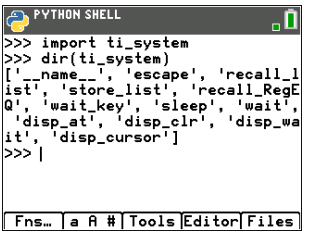


random module

Dessa skärmar visar modulinnehållet för time och ti_system.

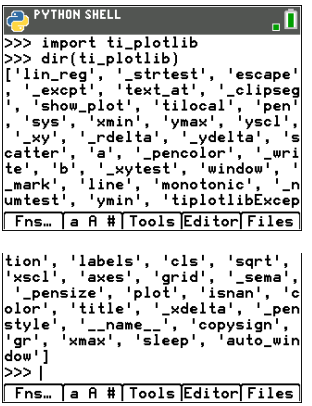


time



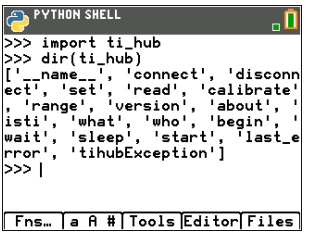
ti_system

Dessa skärmar visar modulinnehållet för ti_plotlib.



ti_plotlib

Den här skärmen visar modulinnehållet för ti_hub.



ti_hub

Dessa skärmar visar modulinnehållet för ti_rover.

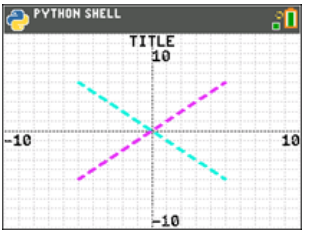
ti_rover

Exempelprogram

Använd de följande exempelprogrammen till att bekanta dig med metoderna från Referens-avsnittet. Dessa exempel innehåller även flera program för TI-Innovator™ Hub och
TI-Innovator Rover™ för att hjälpa dig att komma igång med TI-Python.

COLORLIN

```
import ti_plotlib as plt
plt.cls()
plt.window(-10,10,-10,10)
plt.axes("on")
plt.grid(1,1,"dot")
plt.title("TITLE")
plt.pen("medium","solid")
plt.color(28,242,221)
plt.pen("medium","dash")
plt.line(-5,5,5,-5,"")
plt.color(224,54,243)
plt.line(-5,-5,5,5,"")
plt.show_plot()
```



Tryck på clear för att visa Shell-prompten

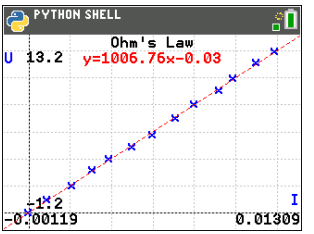
REGEQ1

Förbered en regressionsekvation innan du kör Python-programmet i Python App. Mata t.ex. in två listor i CE-räknarens OS. Beräkna sedan [stat] CALC 4:LinReg(ax+b) för dina listor. Detta lagrar regressionsekvationen till RegEQ i OS. Här är ett program för att återkalla RegEQ till Python experience.

```
# Exempel på recall_RegEQ()
from ti_system import *

reg=recall_RegEQ()
print(reg)
x=float(input("Input x = "))
print("RegEQ(x) = ",eval(reg))
```

LINREGR (Tillhandahålls i CE Bundle)

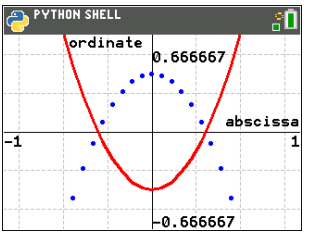
```
import ti_plotlib as plt

# ström
I = [0.0, 0.9, 2.1, 3.1, 3.9, 5.0, 6.0, 7.1, 8.0, 9.2, 9.9, 11.0,11.9]

# voltage
for n in range (len(I)):
I[n] /= 1000

# Spänning
U = [0, 1, 2, 3.2, 4, 4.9, 5.8, 7, 8.1, 9.1, 10, 11.2, 12]

plt.cls()
plt.auto_window(I,U)
plt.pen("thin","solid")
plt.axes("on")
plt.grid(.002,2,"dot")
plt.title("Ohm's Law")
plt.color (0,0,255)
plt.labels("I","U",11,2)
plt.scatter(I,U,"x")
plt.color (255,0,0)
plt.pen("thin","dash")
plt.lin_reg(I,U,"center",2)
plt.show_plot()
plt.cls()
a=plt.a
b=plt.b
print ("a =",round(plt.a,2))
print ("b =",round(plt.b,2))
```



Tryck på clear för att visa Shell-prompten

GRAPH (Tillhandahålls i CE Bundle)

```
import ti_plotlib as plt
# När du har kört programmet så trycker du på [clear] för att rensa
plottningen och återgå till Shell.

def f(x):
  return 3*x**2-.4

def g(x):
  return -f(x)

def plot(res,xmin,xmax):
  #setup plotting area
  plt.window(xmin,xmax,xmin/1.5,xmax/1.5)
  plt.cls()
  gscale=5
  plt.grid((plt.xmax-plt.xmin)/gscale*(3/4),(plt.ymax-
plt.ymin)/gscale,"dash")
  plt.pen("thin","solid")
  plt.color(0,0,0)
  plt.axes("on")
  plt.labels("x-axis"," y-axis",6,1)
  plt.pen("medium","solid")

# plotta f(x) och g(x)
dX=(plt.xmax -plt.xmin)/res
x=plt.xmin
x0=x
  for i in range(res):
    plt.color(255,0,0)
    plt.line(x0,f(x0),x,f(x),"")
    plt.color(0,0,255)
    plt.plot(x,g(x),"o")
    x0=x
    x+=dX
  plt.show_plot()

#plot(resolution (upplösning),xmin,xmax)
plot(30,-1,1)
# Skapa en graf med parametrar(resolution (upplösning),xmin,xmax)
# After clearing the first graph, press the [var] key. Med plot()-
funktionen kan du ändra displayinställningarna (resolution,xmin,xmax).
```

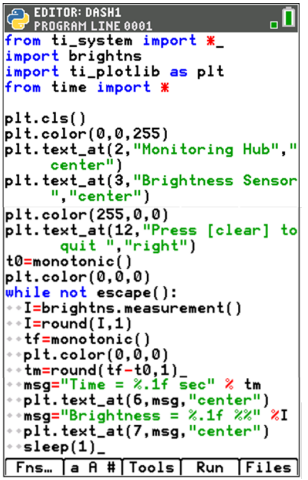


Tryck på clear för att visa Shell-prompten

DASH1 – Exempelprogram för TI-Innovator™ Hub

Se: [Fns…]>Modul: ti_hub module

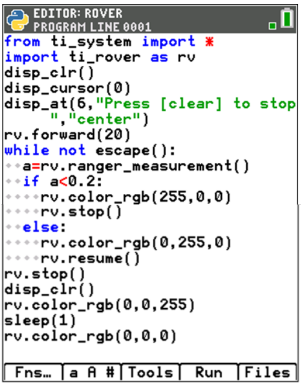
```
from ti_system import *
import brightns
import ti_plotlib as plt
from time import *
plt.cls()
plt.color(0,0,255)
plt.text_at(2,"Monitoring Hub","center")
plt.text_at(3,"Brightness Sensor","center")
plt.color(255,0,0)
plt.text_at(12,"Press [clear] to quit ","right")
t0=monotonic()
plt.color(0,0,0)
while not escape():
  I=brightns.measurement()
  I=round(I,1)
  tf=monotonic()
  plt.color(0,0,0)
  tm=round(tf-t0,1)
  msg="Time = %.1f sec" % tm
  plt.text_at(6,msg,"center")
  msg="Brightness = %.1f %%" %I
  plt.text_at(7,msg,"center")
  sleep(1)
```



ROVER – Exempelprogram för TI-Innovator™ Rover

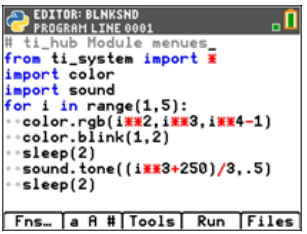
Se: [Fns…]>Modul ti_rover-modul

```
from ti_system import *
import ti_rover as rv
disp_clr()
disp_cursor(0)
disp_at(6,"Press [clear] to stop","center")
rv.forward(20)
while not escape():
  a=rv.ranger_measurement()
  if a<0.2:
    rv.color_rgb(255,0,0)
    rv.stop()
  else:
    rv.color_rgb(0,255,0)
    rv.resume()
rv.stop()
disp_clr()
rv.color_rgb(0,0,255)
sleep(1)
rv.color_rgb(0,0,0)
```



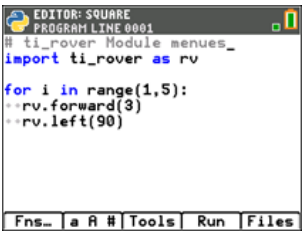
BLNKSND - Exempelprogram för TI-Innovator™ Hub

Se: [Fns…]>Modul: ti_hub module



SQUARE - Exempelprogram för TI-Innovator™ Rover

Se: [Fns...]>Modul ti_rover-modul



Referensguide för TI-Python Experience

Python App innehåller menyer med funktioner, klasser, kontroller, operatorer och nyckelord för snabb inklistring i Editor eller Shell. Den följande referenstabellen listar egenskaper i 2nd [catalog] när Appen är aktiv. För en fullständig lista på Python funktioner, klasser, operatorer och nyckelord som finns tillgängliga i den här versionen, se "Utvalda TI-Python Built-in, nyckelord och modulinnehåll."

Den här tabellen är inte avsedd att vara en uttömmande lista av Python som finns tillgänglig i den här utgåvan. Andra funktioner som har support i den här Python-utgåvan kan matas in med alpha-tangenterna från knappsatsen.

De flesta exempel som ges i den här tabellen körs i Shell-prompten (>>>).

KATALOG-lista

Alfabetisk lista

- A
- B
- C
- D
- E
- F
- G
- H
- I
- L
- M
- N
- O
- P
- R
- S
- T
- U
- W
- X
- Y
- Symboler

A

#

Avgränsare

2nd [catalog]

Syntax: #Din kommentar om ditt program.

[a A #]

Beskrivning: I Python börjar en kommentar med tecknet för hash-tagg, #, och den sträcker sig till slutet på raden.

Exempel:

```
#A short explanation of the code.
```

%

Operator

2nd [catalog]

Syntax: x%y or x % y

Beskrivning: Returnerar resten av x/y. Rekommenderad användning är när x och y är heltal.

[a A #]

Exempel:

```
>>>57%2
1
```

Se även fmod(x,y).

//

Operator

2nd [catalog]

Syntax: x//y or x // y

[a A #]

Beskrivning: Returnerar golvvärdet vid divisionen x/y.

Exempel:

```
>>>26//7
3
>>>65,4//3
21,0
```

[a A #]

Beskrivning: Öppnar teckenpaletten [a A #].

Inbegriper accentuerade tecken såsom ç à â è é ê ë î ï ô ö ù û

[a A #] genvägen är på skärmen vid window i Editorn eller Shell-gränssnittet

a gradient; slope

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.a gradient; slope

[Fns...]>Modul eller math 5:ti_plotlib...> Properties 5:a

Beskrivning: Efter plt.linreg() senast kördes i ett program lagras de beräknade värdena av lutning, a, och skärningspunkt, b, i plt.a och plt.b.

Standardvärden: = 0.0

Exempel:

Se exempelprogram: LINREGR.

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

abs()

Modul: Built-in

2nd [catalog]

Syntax: abs(x)

Beskrivning: Returnerar absolutvärdet för ett tal. I den här releasen kan argumentet vara ett heltal eller ett flyttal.

Obs! fabs() är en funktion i math-modulen.

Exempel:

```
>>>abs(-35.4)
35.4
```

acos()

Modul: math

Syntax: acos(x)

Beskrivning: Returnerar arcus cosinus av x i radianer.

Exempel:

```
>>>from math import *
>>>acos(1)
0.0
```

Alternativt exempel: [Tools] > 6:New Shell

```
>>>import math
>>>math.acos(1)
0.0
```

[sin] 7:acos()

[2nd] [catalog]

[Fns…] Modul
1:math… >
Trig
7:acos()

import-
kommandon
finns i
[2nd] [catalog]

and

Nyckelord

Syntax: x and y

Beskrivning: Kan returnera True eller False.
Returnerar ”x” om ”x” är False och annars ”y”.
Klistras in med mellanslag före och efter and.
Redigera efter behov.

Exempel:

```
>>>2<5 and 5<10
True
>>>2<5 and 15<10
False
>>>{1} and 3
3
>>>0 and 5 < 10
0
```

[2nd] [test]
Ops 8:and

[Fns…] > Ops
8:and

[2nd] [catalog]

[a A #]

.append(x)

Modul: Built-in

Syntax: listname.append(item)

Beskrivning: Metoden append() bifogar ett objekt i en lista.

Exempel:

```
>>>listA = [2,4,6,8]
>>>listA.append(10)
>>>print(listA)
[2,4,6,8,10]
```

2nd [list]
List
6: .append(x)

2nd [catalog]

[Fns…] > List
6:.append(x)

as

Nyckelord

Beskrivning: Använd "as" för att skapa en alias vid import av en modul. Se Python-dokumentationen för mer information.

2nd [catalog]

asin()

Modul: math

Syntax: asin()

Beskrivning: Returnerar arcus sinus av x i radianer.

Exempel:

```
>>>from math import *
>>>asin(1)
1.570796326794897
```

Alternativt exempel:

```
>>>import math
>>>math.asin(1)
1.570796326794897
```

sin 6:asin()

2nd [catalog]

[Fns…] >
Modul
1:math... >
Trig
6:asin()

import-
kommandon
finns i
2nd [catalog]

assert

Nyckelord 2nd [catalog]

Beskrivning: Använd "assert" för att testa ett villkor i din kod. Returnerar None eller om inte, exekvering av programmet visar ett fel, AssertionError.

atan()

Modul: math

sin 8:atan()

Syntax: atan(x)

Beskrivning: Returnerar arcus tangens av x i radianer.

[Fns…]>Modul 1:math... > Trig 8 :atan()

Exempel:

```
>>>from math import *
>>>atan(1)*4
3.141592653589793
```

Alternativt exempel:

2nd [catalog]

```
>>>import math
>>>math.atan(1)*4
3.141592653589793
```

import-kommandon finns i 2nd [catalog]

atan2(y,x)

Modul: math

sin 9:atan2()

Syntax: atan2(y,x)

Beskrivning: Returnerar arcus tangens av y/x i radianer. Resultat är i [-pi, pi].

[Fns…] > Modul 1:math... > Trig 9:atan2()

Exempel:

```
>>>from math import *
>>>atan2(pi,2)
1.003884821853887
```

2nd [catalog]

Alternativt exempel:

```
>>>import math
>>>math.atan2(math.pi,2)
1.003884821853887
```

import-kommandon finns i 2nd [catalog]

auto_window(xlist,ylist)

Modul: ti_plotlib

Syntax: plt.auto_window(xlist,ylist)

Beskrivning: Skalförändrar automatiskt plottningsfönstret till att passa de dataintervall inom xlist och ylist som angivits i programmet före auto_window().

Obs! max(list) - min(list) > 0.00001

Exempel:

Se exempelprogram: LINREGR.

2nd [catalog]

[Fns...]>Modul eller math 5:ti_plotlib...> Setup 5:auto_window ()

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

axes("mode")

Modul: ti_plotlib

Syntax: plt.axes("mode")

Beskrivning: Visar axlar i det specificerade fönstret i plottningsområdet.

Argument:

Argumentalternativ för "mode":

"off"	no axes
"on"	axes+labels
"axes"	axes only
"window"	window labels only

plt.axes() använder den aktuella inställningen för pennans färg. För att garantera att plt.axes() alltid ritas som förväntat, använd plt.color() FÖRE plt.axes() för att garantera förväntade färger.

Exempel:

Se exempelprogram LINREGR.

[2nd] [catalog]

[Fns...]>Modul eller [math] 5:ti_plotlib...> Setup 6:axes()

import-kommandon finns i [2nd] [catalog] eller i ti_plotlib Setup-menyn.

b y= intercept

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.b y= intercept

Beskrivning: Efter att plt.linreg() har körts i ett program lagras de beräknade värdena av lutning, a, och skärningspunkt, b, i plt.a och plt.b.

[Fns...]>Modul eller math 5:ti_plotlib...> Properties 6:b

Standardvärden: = 0.0

Exempel:

Se exempelprogram LINREGR.

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

bin(integer)

Modul: Built-in

2nd [catalog]

Syntax: bin(integer)

Beskrivning: Visar heltalsargumentet i binärt format.

Se Python-dokumentationen för mer information.

Exempel:

```
>>> bin(2)
'0b10'
>>> bin(4)
'0b100'
```

break

Nyckelord

2nd [catalog]

Beskrivning: Använd break för att avsluta en for- eller while-loop.

ceil()

Modul: math

Syntax: ceil(x)

Beskrivning: Returnerar det minsta heltalet större än eller lika med x.

Exempel:

```
>>>from math import *
>>>ceil(34.46)
35
>>>ceil(678)
678
```

[math] Modul
1:math... Math
8:ceil()

[2nd] [catalog]

[Fns...] Modul
1:math...Math
8:ceil()

import-kommandon finns i
[2nd] [catalog]

choice(sequence)

Modul: random

Syntax: choice(sequence)

Beskrivning: Returnerar slumpmässigt ett element från en icke-tom sekvens.

Exempel:

```
>>>from random import *
>>>listA=[2,4,6,8]
>>>choice(listA) #Ditt resultat kan bli ett annat.
4
```

[math] Modul
2:random...
Random
5:choice(sequence)

[2nd] [catalog]

[Fns...] Modul
2:random...
Random
5:choice(sequence)

import-kommandon finns i
[2nd] [catalog]

chr(integer)

Modul: Built-in

2nd [catalog]

Syntax: chr(integer)

Beskrivning: Returnerar en sträng från ett inmatat heltal som representerar unicode-tecknet.

Se Python-dokumentationen för mer information.

Exempel:

```
>>> char(40)
'('
>>> char(35)
'#'
```

class

Nyckelord

2nd [catalog]

Beskrivning: Använd class för att skapa en klass. Se Python-dokumentationen för mer information.

cls() clear screen

Modul: ti_plotlib

Syntax: plt.cls() clear screen

Beskrivning: Rensar Shell-skärmen för plottningen. Kortkommandon visas inte vid plottning.

Obs! plt.cls() har ett annorlunda beteende än ti_system module disp_clr().

Exempel:

Se exempelprogram: GRAPH.

2nd [catalog]

[Fns...]>Modul eller math
5:ti_plotlib...>
Setup
2:cls()

[Fns...]>Modul eller math
5:ti_plotlib...>
Draw
2:cls()

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

color(r,g,b) 0-255

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.color(r,g,b) 0-255

[Fns...]>Modul eller math 5:ti_plotlib...> Draw 1:color()

Beskrivning: Ställer in färgen för all därpå följande grafik/plottning. (r,g,b)-värden måste anges 0-255. Specificerad färg används i plottvisning tills color() utförs igen med en annan färg.

Standardfärgen är svart vid import av ti_plotlib.

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

Exempel:

Se exempelprogram: COLORLIN.

complex(real,imag)

Modul: Built-in

2nd [catalog]

Syntax: complex(real,imag)

[Fns...]>Type> 5:complex()

Beskrivning: Komplex taltyp

Exempel:

```
>>>z = complex(2, -3)
>>>print(z)
(2-3j)
>>>z = complex(1)
>>>print(z)
(1+0j)
>>>z = complex()
>>>print(z)
0j
>>>z = complex("5-9j")
>>>print(z)
(5-9j)
```

Obs! "1+2j" är rätt syntax. Mellanslag såsom "1 + 2j" kommer att visa en Exception (ett undantag).

continue

Nyckelord 2nd [catalog]

Beskrivning: Använd continue i en for- eller while-loop för att avsluta den aktuella iterationen. Se Python-dokumentationen för mer information.

cos()

Modul: math

sin Trig
4: cos()

Syntax: cos(x)

Beskrivning: Returnerar cos av x. Vinkelargumentet är i radianer.

2nd [catalog]

Exempel:

```
>>>from math import *
>>>cos(0)
1.0
>>>cos(pi/2)
6.123233995736767e-17
```

[Fns...] Modul
1:math... > Trig
4:cos()

Alternativt exempel:

```
>>>import math
>>>math.cos(0)
1.0
```

Obs! Python visar grundpotensform med e eller E. Vissa resultat från math i Python skiljer sig därför från de i CE-räknarens OS.

.count()

Modul: Built-in

2nd [catalog]

Syntax: listname.count(objekt)

Beskrivning: count() är en metod som returnerar antalet förekomster av ett objekt i en lista, tupel, bytes, str, bytearray eller array.array objekt.

Exempel:

```
>>>listA = [2,4,2,6,2,8,2,10]
>>>listA.count(2)
4
```

def function ():

Nyckelord

2nd [catalog]

Syntax: def function(var, var,...)

Beskrivning: Definiera en funktion beroende av angivna variabler. Används vanligtvis med nyckelordet return.

[Fns...]>Func
1:def function():

Exempel:

[Fns...]>Func
2:return

```
>>> def f(a,b):
…return a*b
…
…
…
>>> f(2,3)
6
```

degrees()

Modul: math

sin Trig
2:degrees()

Syntax: degrees(x)

Beskrivning: Konverterar en vinkel x i radianer till grader.

2nd [catalog]

Exempel:

```
>>>from math import *
>>>degrees(pi)
180.0
>>>degrees(pi/2)
90.0
```

[Fns...]>Modul
1:math...>Trig
2:degrees()

del

Nyckelord

2nd [catalog]

Beskrivning: Använd "del" för att ta bort objekt såsom variabler, listor, etc.
Se Python-dokumentationen för mer information.

disp_at(row,col,"text")

Modul: ti_system

Syntax: disp_at(row,col,"text")

Beskrivning: Visar text som börjar vid positionen för en rad och kolumn i plottningsområdet.

REPL with cursor >>>| visas efter texten om det är vid slutet av programmet. Använd disp_cursor() för att styra markörens visning.

Argument:

row	1 - 11, heltal
column	1 - 32, heltal
"text"	är en sträng som radbryts i skärmområdet

Alternativa argument för färg och bakgrund visas här: disp_at(row,col,"text","align",color 0-15, background color 0-5)

Exempel:

Exempelprogram:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,6,"hello")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
7:disp_at()

[Fns...]>Modul
eller math
4:ti_system
7:disp_at()

import-kommandon finns i 2nd [catalog] eller i ti_system Modul-menyn.

disp_at(row,"text","align")

Modul: ti_system

Syntax: disp_at(row,"text","align")

Beskrivning: Visar text justerad enligt specifikation på plottningsskärmen för rad 1-11. Rad rensas före visning. Om det används i en loop uppdateras innehållet vid varje visning.

REPL with cursor >>>| visas efter texten om det är vid slutet av programmet. Använd disp_cursor() för att styra markörens visning innan du använder disp_at() i ditt program.

Argument:

row	1 - 11, heltal
"text"	är en sträng som radbryts i skärmområdet
"align"	"left" (standard) "center" "right"

Tillvalsargument visas här: disp_at (row,"text","align","color 0-15, background color 0-15)

Exempel:

Exempelprogram:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
7:disp_at()

[Fns...]>Modul
eller math
4:ti_system
7:disp_at()

import-kommandon finns i 2nd [catalog] eller i ti_system Modul-menyn.

disp_clr() clear text screen

Modul: ti_system

Syntax: disp_clr() clear text screen

Beskrivning: Rensar skärmen i Shell-miljön. Rad 0-11, heltal kan användas som ett alternativt argument för att rensa en visningsrad i Shell-miljön.

Exempel:

Exempelprogram:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
8:disp_clr()

[Fns...]>Modul
eller math
4:ti_system
8:disp_clr()

import-
kommandon
finns i 2nd
[catalog] eller i
ti_system
Modul-menyn.

disp_cursor() 0=off 1=on

Modul: ti_system

Syntax: disp_cursor() 0=off 1=on

Beskrivning: Styr visningen av markören i Shell när ett program körs.

Argument:

0 = av

inte 0 = på

Exempel:

Exempelprogram:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

[2nd] [catalog]

[2nd] [rcl]
ti_system
0:disp_cursor()

[Fns…]>Modul eller
[math]
4:ti_system
0:disp_cursor()

import-kommandon finns i [2nd] [catalog] eller i ti_system Modul-menyn.

disp_wait() [clear]

Modul: ti_system

Syntax: disp_wait() [clear]

Beskrivning: Stoppar exekveringen av programmet vid denna punkt och visar skärminnehåll tills [clear] trycks in och skärmen rensas.

Exempel:

Exempelprogram:

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nd [catalog]

2nd [rcl]
ti_system
9:disp_wait()

[Fns…]>Modul
eller math
4:ti_system
9:disp_wait()

import-
kommandon
finns i 2nd
[catalog] eller i
ti_system
Modul-menyn.

e

Modul: math

2nd [e] (ovanför ÷)

Syntax: math.e eller e om math-modulen importerats

Beskrivning: Konstanten e visas enligt nedan.

Exempel:

[Fns...] > Modul 1:math... > Const 1:e

```
>>>from math import *
>>>e
2.718281828459045
```

Alternativt exempel:

```
>>>import math
>>>math.e
2.718281828459045
```

elif :

Nyckelord

2nd [catalog]

Se if..elif..else.. för ytterligare information.

[Fns...] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

else:

Nyckelord

Se if..elif..else.. för ytterligare information.

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

escape()

Modul: ti_system

Syntax: escape()

Beskrivning: escape() returnerar True eller False.

Initialvärdet är False.

När tangenten [clear] trycks in på CE-räknaren, ställs värdet in på True.

När funktionen utförs återställs värdet till False.

Användningsexempel:

while not escape():

I en while-loop som körs i ett program där programmet erbjuder att avsluta loopen men fortsätta skriptet.

if escape():break

Kan användas för att felsöka program för att inspektera variabler med Shell [vars] efter att ha kört programmet och använt break.

2nd [catalog]

Som en programrad:

2nd [rcl]
ti_system
5:while not escape():
6:if escape():break

[Fns…]>Modul eller math
4:ti-system
5:while not escape():
6:if escape():break

import-kommandon finns i 2nde [catalog] eller i ti_system Modul-menyn.

eval()

Modul: Built-in

2nd [catalog]

Syntax: eval(x)

[Fns…] I/O
3:eval()

Beskrivning: Returnerar evalueringen av uttrycket x.

Exempel:

```
>>>a=7
>>>eval(”a+9”)
16
>>>eval(’a+10’)
17
```

except exception:

Nyckelord

2nd [catalog]

Beskrivning: Använd except i ett kodblock av typen try..except. Se Python-dokumentationen för mer information.

exp()

Modul: math

Syntax: exp(x)

Beskrivning: Returnerar e**x.

Exempel:

```
>>>from math import *
>>>exp(1)
2.718281828459046
```

Alternativt exempel: [Tools] > 6:New Shell

```
>>>import math
>>>math.exp(1)
2.718281828459046
```

2nd [e^x] (ovanför ln)

2nd [catalog]

[Fns…] > Modul 1:math… 4:exp()

import-kommandon finns i 2nd [catalog].

.extend()

Modul: Built-in

Syntax: listname.extend(newlist)

Beskrivning: Metoden extend() är en metod för att utöka newlist vid slutet av en lista.

Exempel:

```
>>>listA = [2,4,6,8]
>>>listA.extend([10,12])
>>>print(listA)
[2,4,6,8,10,12]
```

2nd [catalog]

fabs()

Modul: math

2nd [catalog]

Syntax: fabs(x)

Beskrivning: Returnerar absolutvärdet av x

[Fns…] > Modul 1:math... 2:fabs()

Exempel:

```
>>>from math import *
>>>fabs(35-65.8)
30.8
```

import-kommandon finns i 2nd [catalog].

Se även Built-in-funktionen abs().

False

Nyckelord

2nd [test] (ovanför math)

Beskrivning: Returnerar False när exekverat påstående är False. "False" representerar det falska värdet av objekt av typen booleska objekt.

2nd [catalog]

Exempel:

```
>>>64<=32
False
```

[Fns…] > Ops B:False

[a A #]

finally:

Nyckelord

2nd [catalog]

Beskrivning: Använd finally i ett kodblock av typen try..except..finally. Se Python-dokumentationen för mer information.

float()

Modul: Built-in

2nd [catalog]

Syntax: float(x)

Beskrivning: Returnerar x som ett flyttal.

[Fns…] > Type
2:float()

Exempel:

```
>>>float(35)
35.0
>>>float(”1234”)
1234.0
```

floor()

Modul: math

math Modul
1:math
9:floor()

Syntax: floor(x)

Beskrivning: Returnerar det största heltalet mindre än eller lika med x.

2nd [catalog]

Exempel:

```
>>>from math import *
>>>floor(36.87)
36
>>>floor(-36.87)
-37
>>>floor(254)
254
```

[Fns…] > Modul
1:math
9:floor()

import-kommandon finns i
2nd [catalog]

fmod(x,y)

Modul: math

Syntax: fmod(x,y)

Beskrivning: Se Python-dokumentationen för mer information. Rekommenderad användning är när x och y är flyttal.

Kanske inte returnerar samma resultat som x%y.

Exempel:

```
>>>from math import *
>>>fmod(50.0,8.0)
2.0
>>>fmod(-50.0,8.0)
-2.0
>>>-50.0 - (-6.0)*8.0 #validation from description
-2.0
```

Se även: x%y.

math Modul
1:math
7:fmod()

2nd [catalog]

[Fns…] > Modul
1:math…
7:fmod()

import-kommandon finns i
2nd [catalog]

for i in list:

Nyckelord

Syntax: for i in list:

Beskrivning: Används för att iterera över listelement.

Exempel:

```
>>> for i in [2,4,6]:
… print(i)
…
…
…
2
4
6
```

[Fns…] Ctl
7:for i in list:

2nd [catalog]

for i in range(size):

Nyckelord

[Fns…] Ctl 4:for i in range (size):

Syntax: for i in range(size)

Beskrivning: Används för att iterera över ett intervall.

Exempel:

2nd [catalog]

```
>>> for i in range(3):
… print(i)
…
…
…
0
1
2
```

for i in range(start,stop):

Nyckelord

[Fns…] Ctl 5:for i in range (start,stop):

Syntax: for i in range(start,stop)

Beskrivning: Används för att iterera över ett intervall.

Exempel:

2nd [catalog]

```
>>> for i in range(1,4):
… print(i)
…
…
…
1
2
3
```

for i in range(start,stop,step):

Nyckelord

[Fns…] Ctl
6:for i in range
(start,stop,step):

Syntax: for i in range(start,stop,step)

Beskrivning: Används för att iterera över ett intervall.

Exempel:

2nd [catalog]

```
>>> for i in range(1,8,2):
… print(i)
…
…
…
1
3
4
7
```

str.format() string format

Modul: Built-in

2nd [catalog]

Syntax:str.format()

Beskrivning: Formaterar den givna strängen. Se Python-dokumentationen för mer information.

Exempel:

```
>>> print("{+f}".format(12.34))
+12.340000
```

frexp()

Modul: math

Syntax: frexp(x)

Beskrivning: Returnerar ett talpar (y,n) där x == y * 2**n. y är ett flyttal 0.5<abs(y)<1; och n är ett heltal.

Exempel:

```
>>>from math import *
>>>frexp(2000.0)
(0.9765625, 11)
>>>0.9765625 * 2**11 #validate description
2000.0
```

[math] Modul
1:math
A:frexp()

[2nd] [catalog]

[Fns…] > Modul
1:math
A:frexp()

import-kommandon finns i [2nd] [catalog]

from PROGRAM import *

Nyckelord

Syntax: from PROGRAM import *

Beskrivning: Används för att importera ett program. Importerar de allmänna attributen för en Python-modul till den aktuella namnrymden.

Shell [Tools]
A:from PROGRAM import *

[2nd] [catalog]

from math import *

Nyckelord

Syntax: from math import *

Beskrivning: Används för att import alla funktioner och konstanter från math module.

[math] Modul
1:math…
1:from math import *

[Fns..] > Modul
1:math…
1:from math import *

[2nd] [catalog]

from random import *

Nyckelord

Syntax: from random import *

Beskrivning: Används för att import alla funktioner från random module.

[math] Modul
2:random…
1:from random import *

[Fns..] > Modul
2:random…
1:from random import *

[2nd] [catalog]

from time import *

Nyckelord

2nd [catalog]

Syntax: from time import *

math Modul 3:time… 1:from time import *

Beskrivning: Används för att importera alla metoder från time-modulen.

Exempel:

[Fns…]>Modul 3:time… 1:from time import *

Se exempelprogram: DASH1.

from ti_system import *

Nyckelord

2nd [catalog]

Syntax: from ti_system import *

math Modul 4:ti_system… 1:from system import *

Beskrivning: Används för att importera alla metoder från ti_system-modulen.

Exempel:

[Fns…]>Modul 4:ti_system… 1:from system import *

Se exempelprogram: REGEQ1.

from ti_hub import *

Nyckelord 2nd [catalog]

Syntax: from ti_hub import *

Beskrivning: Används för att importera alla metoder från ti_hub module. För enskilda inmatnings- och utmatningsenheter, använd den dynamiska modulfunktionaliteten genom att välja enheten från [Fns…]>Modul>ti_hub>Import-menyn när du är i Editorn.

Se:ti_hub module – Lägg till import till Editor och lägg till ti_hub sensor module till Modul-menyn.

Exempel:

Se exempelprogram: DASH1.

global

Nyckelord 2nd [catalog]

Beskrivning: Använd global för att skapa globala variabler inuti en funktion.

Se CircuitPython-dokumentationen för mer information.

grid(xscl,yscl,"style")

Modul: ti_plotlib 2nd [catalog]

Syntax: plt.grid(xscl,yscl,"style")

[Fns...]>Modul eller math 5:ti_plotlib...> Setup 3:grid()

Beskrivning: Visar ett rutnät med specificerade skalor för x- och y-axlar. Obs! All plottning äger rum när plt.show_plot() utförs.

Ställa in färgen på rutnätet är det valfria argumentet av (r,g,b) med värden 0-255 där standardvärdet är grå (192,192,192).

Standardvärde för xscl eller yscl = 1.0.

"style" = "dot" (standard), "dash", "solid" eller "point"

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

Exempel:

Se exempelprogram: COLORLIN eller GRAPH.

grid(xscl,yscl,"style",(r,g,b))

Modul: ti_plotlib

Syntax: plt.grid(xscl,yscl,"style",(r,g,b))

Beskrivning: Visar ett rutnät med specificerade skalor för x- och y-axlar. Obs! All plottning äger rum när plt.show_plot() utförs.

Ställa in färgen på rutnätet är det valfria argumentet av (r,g,b) med värden 0-255 där standardvärdet är grå (192,192,192).

Standardvärde för xscl eller yscl = 1.0.

"style" = "dot" (standard), "dash", "solid" eller "point" .

Om xscl- eller yscl- värden är mindre än 1/50 av skillnaden mellan xmax-xmin eller ymax-ymin så får man 'Invalid grid scale value'.

Exempel:

Se exempelprogram: GRAPH.

[2nd] [catalog]

[Fns...]>Modul eller [math] 5:ti_plotlib...> Setup 3:grid()

import-kommandon finns i [2nd] [catalog] eller i ti_plotlib Setup-menyn.

hex(integer)

Modul: Built-in

2nd [catalog]

Syntax: hex(integer)

Beskrivning: Visar heltalsargumentet i hexadecimalt format. Se Python-dokumentationen för mer information.

Exempel:

```
>>> hex(16)
'0x10'
>>> hex(16**2)
'0x100'
```

"if :"

Se if..elif..else.. för ytterligare information.

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

if..elif..else..

Nyckelord

2nd [catalog]

Syntax: ••Gråa identifierare för indrag tillhandahålls automatiskt i Python App för förenklad användning.

if :

••

elif :

••

else:

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

Beskrivning: if..elif..else är ett villkorligt påstående. Editorn tillhandahåller automatiskt indrag som gråa prickar för att hjälpa dig med riktiga programmeringsindrag.

Exempel: Skapa och kör det här programmet, säg S01, från Editorn

```
def f(a):
••if a>0:
••••print(a)
••elif a==0:
••••print(”zero”)
••else:
••••a=-a
••••print(a)
```

Shell-interaktion

```
>>> # Shell Reinitialized
>>> # Running S01
>>>from S01 import * #automatically pastes
>>>f(5)
5
>>>f(0)
zero
>>>f(-5)
5
```

if..else..

Nyckelord

2nd [catalog]

Se if..elif..else.. för ytterligare information.

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

.imag

Modul: Built-in

2nd [catalog]

Syntax:var.imag

Beskrivning: Returnerar den imaginära delen av en specificerad variabel som är av taltypen komplext tal.

Exempel:

```
>>>a=complex(4,5)
>>>a.real
4
>>>a.imag
5
```

import math

Nyckelord

Syntax: import math

2nd [catalog]

Beskrivning: Ta fram math module med det här kommandot. Den här instruktionen importerar de allmänna attributen i "math"-modulen till sin egen namnrymd.

import random

Nyckelord

Syntax: import random

2nd [catalog]

Beskrivning: Ta fram random module med det här kommandot. Den här instruktionen importerar de allmänna attributen i "random"-modulen till sin egen namnrymd.

import ti_hub

Nyckelord

2nd [catalog]

Syntax: import ti_hub

Beskrivning: Ta fram ti_hub module med det här kommandot. Den här instruktionen importerar de allmänna attributen i ti_hub module till sin egen namnrymd.

För enskilda inmatnings- och utmatningsenheter, använd den dynamiska modulfunktionaliteten genom att välja enheten från [Fns…]>Modul>ti_hub>Import-menyn när du är i Editorn.

Se:[Fns…] > Modul: ti_hub module.

import time

Nyckelord

2nd [catalog]

Syntax: import time

Beskrivning: Ta fram time module med det här kommandot. Den här instruktionen importerar de allmänna attributen i time-modulen till sin egen namnrymd.

Se:[Fns…] > Modul: time- och ti_system-moduler.

import ti_plotlib as plt

Nyckelord

Syntax: import ti_plotlib as plt

Beskrivning: Ta fram ti_plotlib module med det här kommandot. Den här instruktionen importerar de allmänna attributen i ti_plotlib module till sin egen namnrymd. Attributen i ti_plotlib module måste anges som plt.attribute.

Exempel:

Se exempelprogram: COLORLIN.

2nd [catalog]

math Modul
5:ti_plotlib...
1:import ti_plotlib as plt

[Fns…]>Modul
5:ti_plotlib...
1:import ti_plotlib as plt

import ti_rover as rv

Nyckelord

Syntax: import ti_rover as rv

Beskrivning: Ta fram ti_rover module med det här kommandot. Den här instruktionen importerar de allmänna attributen i ti_rover module till sin egen namnrymd. Attributen i ti_rover module måste anges som rv.attribute.

Exempel:

Se exempelprogram: ROVER.

2nd [catalog]

math Modul
7:ti_rover...
1:import ti_rover as rv

[Fns…]>Modul
7:ti_rover...
1:import ti_rover as rv

import ti_system

Nyckelord 2nd [catalog]

Syntax: import ti_system

Beskrivning: Ta fram ti_system module med det här kommandot. Den här instruktionen importerar de allmänna attributen i ti_system module till sin egen namnrymd.

Exempel:

Se exempelprogram: REGEQ1.

in

Nyckelord 2nd [catalog]

Beskrivning: Använd "in" för att kontrollera om ett värde finns i en sekvens eller för att iterera en sekvens i en for-loop.

.index(x)

Modul: Built-in 2nd [catalog]

Syntax:var.index(x)

Beskrivning: Returnerar index eller positionen för ett element i en lista. Se Python-dokumentationen för mer information.

Exempel:

```
>>> a=[12,35,45]
>>> print(a.index(12))
0
>>> print(a.index(35))
1
>>> print(a.index(45))
2
```

input()

Modul: Built-in 2nd [catalog]

input()

Syntax: input()

Beskrivning: Prompt för inmatning

[Fns…] I/O
2:input()

Exempel:

```
>>>input("Name? ")
Name? Me
'Me'
```

Alternativt exempel:

```
CreateProgram A
len=float(input("len: "))
print(len)


RunProgram A
>>> # Shell Reinitialized
>>> # Running A
>>>from A import *
len: 15 (mata in 15)
15.0 (output float 15.0)
```

.insert(index,x)

Modul: Built-in

2nd [list] List
8:.insert(index,x)

Syntax: listname.insert(index,x)

Beskrivning: Metoden method insert() infogar ett objekt x efter index inom en sekvens.

2nd [catalog]

Exempel:

```
>>>listA = [2,4,6,8]
>>>listA.insert(3,15)
>>>print(listA)
[2,4,6,15,8]
```

[Fns…] > List
8:.insert(index,x)

int()

Modul: Built-in

2nd [catalog]

Syntax: int(x)

Beskrivning: Returnerar x som ett heltalsobjekt.

[Fns…] > Type
1:int()

Exempel:

```
>>>int(34.67)
34
>>>int(1234.56)
1234
```

is

Nyckelord

2nd [catalog]

Beskrivning: Använd "is" för att testa om två objekt är samma objekt.

labels("xlabel","ylabel",x,y)

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.labels("xlabel","ylabel",x,y)

[Fns...]>Modul eller math 5:ti_plotlib...> Setup 7:labels()

Beskrivning: Visar förklaringar "xlabel" och "ylabel" på diagramaxlarna vid radpositioner x och y. Justera efter behov för din diagramvisning.

"xlabel" är positionerad på specificerad rad x (standardrad 12) och är högerjusterad.

"ylabel" är positionerad på specificerad rad y (standardrad 2) och är vänsterjusterad.

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

Obs! plt.labels("|","",12,2) klistras in med x- och y-rader standard, 12,2 , vilket sedan kan ändras för ditt program.

Exempel:

Se exempelprogram: GRAPH.

lambda

Nyckelord

2nd [catalog]

Syntax: lambda arguments : expression

Beskrivning: Använd lambda för att definiera en anonym funktion. Se Python-dokumentationen för mer information.

len()

Modul: Built-in

Syntax: len(sequence)

Beskrivning: Returnerar antalet objekt i argumentet. Argumentet kan vara en sekvens eller en samling.

Se Python-dokumentationen för mer information.

Exempel:

```
>>>mylist=[2,4,6,8,10]
>>>len(mylist)
5
```

[2nd] [list] (ovanför [stat])
List
3:len()

[2nd] [catalog]

[Fns...] > List
3:len()

line(x1,y1,x2,y2,"mode")

Modul: ti_plotlib

Syntax: plt.line(x1,y1,x2,y2,"mode")

Beskrivning: Visar ett linjesegment från (x1,y1) till (x2,y2)

Storlek och stil ställs in med pen() och color() före line().

Argument:

x1,y1, x2,y2 är reella flyttal.

"mode": Om standard "", ritas inga pilspetsar. Om "arrow" ritas en vektorpilspets vid (x2,y2).

Exempel:

Se exempelprogram: COLORLIN.

[2nd] [catalog]

[Fns...]>Modul eller [math]
5:ti_plotlib...> Draw
7:line eller vector

import-kommandon finns i [2nd] [catalog] eller i ti_plotlib Setup-menyn.

lin_reg(xlist,ylist,"disp",row)

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.lin_reg(xlist,ylist,"disp",row)

[Fns...]>Modul eller math 5:ti_plotlib...> Draw 8:lin_reg()

Beskrivning: Beräknar och ritar efter den linjära regressionsmodellen, ax+b från xlist, ylist. Den här metoden måste följa spridningsdiagrammetoden Standardvisning av ekvationen är "center" (mitten) vid rad 11.

Argument:

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

"disp"	"left"
	"center"
	"right"
row	1 - 12

plt.a (lutning) och plt.b (skärningspunkt) lagras när lin_reg utförs.

Exempel:

Se exempelprogram: LINREGR.

list(sequence)

Modul: Built-in

Syntax: list(sequence)

Beskrivning: Mutable sekvens av objekt av typen spara.

list()" konverterar sitt argument till typen "list". Precis som många andra sekvenser behöver elementen i en lista inte vara av samma typ.

Exempel:

```
>>>mylist=[2,4,6,8]
>>>print(mylist)
[2,4,6,8]
```

Exempel:

```
>>>mylist=[2,4,6,8]
>>>print(mylist)
[2,4,6,8]
>>> list({1,2,"c", 7})
[7, 1, 2, 'c']
>>> list("foobar")
['f', 'o', 'o', 'b', 'a', 'r']
```

2nd [list] (ovanför stat) List
2:list(sequence)

2nd [catalog]

[Fns…] > List
2:list(sequence)

log(x,base)

Modul: math

Syntax: log(x,base)

Beskrivning: log(x) utan base returnerar den naturliga logaritmen av x.

Exempel:

```
>>>from math import *
>>>log(e)
1.0
>>>log(100,10)
2.0
>>>log(32,2)
5.0
```

2nd log for log (x,10)

2nd ln for log(x) (naturlig log)

math Modul 1:math… 6:log(x,base)

2nd [catalog]

[Fns…] > Modul 1:math… 6:log(x,base)

import-kommandon finns i 2nd [catalog]

math.funktion

Modul: math

2nd [catalog]

Syntax: math.funktion

Beskrivning: Använd efter kommandot import math för att använda en funktion i math module.

Exempel:

```
>>>import math
>>>math.cos(0)
1.0
```

max()

Modul: Built-in

2nd [list] (ovanför stat) List 4:max()

Syntax: max(sequence)

Beskrivning: Returnerar det högsta värdet i sekvensen. Se Python-dokumentationen för mer information om max().

2nd [catalog]

Exempel:

```
>>>listA=[15,2,30,12,8]
>>>max(listA)
30
```

[Fns…] > List 4:max()

min()

Modul: Built-in

2nd [list] (ovanför stat) List 5:min()

Syntax: min(sequence)

Beskrivning: Returnerar det minsta värdet i sekvensen. Se Python-dokumentationen för mer information om min().

2nd [catalog]

Exempel:

```
>>>listA=[15,2,30,12,8]
>>>min(listA)
2
```

[Fns…] > List 5:min()

monotonic() elapsed time

Modul: time

2nd [catalog]

Syntax: monotonic() elapsed time

Beskrivning: Returnerar ett tidsvärde från exekveringspunkten. Använd returvärdet för att jämföra med andra värden från monotonic().

[Fns…]>Modul eller math 3:time 3:monotonic()

Exempel:

Exempelprogram:

import-kommandon finns i 2nd [catalog] eller i time Modul-menyn.

```
from time import *
a=monotonic()
sleep(15)
b=monotonic()
print(b-a)
```

```
Kör programmet EXAMPLE tills exekveringen
slutar.
>>>15.0
```

None

Nyckelord 2nd [catalog]

Beskrivning: None representerar avsaknaden av ett värde.

[a A #]

Exempel:

```
>>> def f(x):
…x
…
…
…
>>> print(f(2))
None
```

nonlocal

Nyckelord 2nd [catalog]

Syntax: nonlocal

Beskrivning: Använd nonlocal för att deklarera en variabel som inte är lokal. Se Python-dokumentationen för mer information.

not

Nyckelord 2nd [test] Ops 0:not

Syntax: not x

Beskrivning: Evaluerar till True om x är False och annars till False. Klistras in med mellanslag före och efter keyword not . Redigera efter behov.

[Fns…] > Ops 0:not

Exempel:

```
>>> not 2<5 #edit the space before not
False
>>>3<8 and not 2<5
False
```

2nd [catalog]

[a A #]

oct(integer)

Modul: Built-in

2nd [catalog]

Syntax: oct(integer)

Beskrivning: Returnerar den oktala representationen av heltalet. Se Python-dokumentationen för mer information.

Exempel:

```
>>> oct(8)
'0o10'
>>> oct(64)
'0o100'
```

or

Nyckelord

2nd [test] Ops 9:or

Syntax: x or y

[Fns…] > Ops 9:or

Beskrivning: Kan returnera True eller False. Returnerar x om x evaluerar till True och annars y. Klistras in med mellanslag före och efter or. Redigera efter behov.

2nd [catalog]

Exempel:

[a A #]

```
>>>2<5 or 5<10
True
>>>2<5 or 15<10
True
>>>12<5 or 15<10
False
>>> 3 or {}
3
>>> [] or {2}
{2}
```

ord("character")

Modul: Built-in 2nd [catalog]

Syntax: ord("character")

Beskrivning: Returnerar tecknets unicode-värde. Se Python-dokumentationen för mer information.

Exempel:

```
>>> ord("#")
35
>>> ord("/")
47
```

pass

Nyckelord 2nd [catalog]

Beskrivning: Använd pass i en tom funktion- eller klass-definition som platshållare för framtida kod medan du bygger ditt program. Tomma definitioner orsakar inga fel när programmet exekveras.

pen("size","style")

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.pen("size","style")

Beskrivning: Ställer in utseendet på alla följande rader tills nästa pen() exekveras.

[Fns...]>Modul eller math 5:ti_plotlib...> Draw 9:pen()

Argument:

Standardvärden för pen() är "thin" och "solid".

"size"	"thin"
	"medium"
	"thick"
"style"	"solid"
	"dot"
	"dash"

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

Exempel:

Se exempelprogram: COLORLIN eller GRAPH.

pi

Modul: math

Syntax: math.pi eller pi om math module importerats.

Beskrivning: Konstanten pi visas enligt nedan.

Exempel:

```
>>>from math import *
>>>pi
3.141592653589793
```

Alternativt exempel:

```
>>>import math
>>>math.pi
3.141592653589793
```

2nd [π] (ovanför sin)

[Fns…] > Modul 1:math… > Const 2:pi

plot(xlist,ylist,"mark")

Modul: ti_plotlib

Syntax: plt.plot(xlist,ylist,"mark")

Beskrivning: Ett linjediagram som använder ordnade talpar från specificerad xlist och ylist. Stil och storlek på linje ställs in med plt.pen().

xlist och ylist måste vara reella flyttal och listorna måste ha samma dimension.

Argument:

"mark" är det markerande tecknet enligt följande:

o	filled dot (default)
+	cross
x	x
.	pixel

Exempel:

Se exempelprogram: LINREGR.

2nd [catalog]

[Fns...]>Modul eller math 5:ti_plotlib...> Draw 5:Connected Plot with Lists

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

plot(x,y,"mark")

Modul: ti_plotlib

Syntax: plt.plot(x,y,"mark")

Beskrivning: Ett punktdiagram, (x,y) visas med specificerade x och y.

xlist och ylist måste vara reella flyttal och listorna måste ha samma dimension.

Argument:

"mark" är det markerande tecknet enligt följande:

o	filled dot (default)
+	cross
x	x
.	pixel

Exempel:

Se exempelprogram: LINREGR.

2nd [catalog]

[Fns...]>Modul eller math
5:ti_plotlib...>
Draw
6:plot a Point

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

pow(x,y)

Modul: math

Syntax: pow(x,y)

Beskrivning: Returnerar x upphöjt till potensen y. Konverterar både x och y till flyttal. Se Python-dokumentationen för mer information.

Använd funktionen built-in pow(x,y) eller ** för att beräkna exakta heltalspotenser.

Exempel:

```
>>>from math import *
>>>pow(2,3)
>>>8.0
```

Exempel på användning: Built-in:

[Tools] > 6:New Shell

```
>>>pow(2,3)
8
>>>2**3
8
```

math Modul
1:math
5:pow(x,y)

2nd [catalog]

[Fns…] > Modul
1:math
5:pow(x,y)

import-kommandon finns i
2nd [catalog]

print()

Modul: Built-in

Syntax: print(argument)

Beskrivning: Visar argument som en sträng.

Exempel:

```
>>>x=57.4
>>>print("my number is =", x)
my number is= 57.4
```

2nd [catalog]

[Fns…] > I/O
1:print()

radians()) degree ▶radians

Modul: math

Syntax: radians(x)

Beskrivning: Konverterar en vinkel x i degrees till radians.

Exempel:

```
>>>from math import *
>>>radians(180.0)
3.141592653589793
>>>radians(90.0)
1.570796326794897
```

sin Trig
1:radians()

2nd [catalog]

[Fns…] > Modul
1:math… > Trig
1:radians()

raise

Nyckelord

Syntax: raise exception

Beskrivning: Använd raise för att ta fram en specificerad exception och stoppa ditt program.

2nd [catalog]

randint(min,max)

Modul: random

Syntax: randint(min,max)

Beskrivning: Returnerar ett slumpmässigt heltal mellan min och max.

[math] Modul
2:random
4:randint
(min,max)

Exempel:

```
>>>from random import *
>>>randint(10,20)
>>>15
```

[Fns…] > Modul
2:random…
4:randint
(min,max)

Alternativt exempel:

```
>>>import random
>>>random.randint(200,450)
306
```

[2nd] [catalog]

Resultaten kommer att variera, eftersom output är slumpmässigt.

import-kommandon finns i
[2nd] [catalog]

random()

Modul: random

Syntax: random()

Beskrivning: Returnerar ett flyttal från 0 till 1.0. Den här funktionen tar inga argument.

Exempel:

```
>>>from random import *
>>>random()
0.5381466990230621
```

Alternativt exempel:

```
>>>import random
>>>random.random()
0.2695098437037318
```

Resultaten kommer att variera, eftersom output är slumpmässigt.

[math] Modul 2:random… Random 2:random()

[Fns…] > Modul 2:random… Random 2:random()

[2nd] [catalog]

import-kommandon finns i [2nd] [catalog]

random.funktion

Modul: random

Syntax: random.funktion

Beskrivning: Använd efter import random för att använda en function i modulen random module.

Exempel:

```
>>>import random
>>>random.randint(1,15)
2
```

Resultaten kommer att variera, eftersom output är slumpmässigt.

[2nd] [catalog]

randrange(start,stop,step)

Modul: random

Syntax: randrange(start,stop,step)

Beskrivning: Returnerar ett slumpmässigt tal från start till stopp med steg.

Exempel:

```
>>>from random import *
>>>randrange(10,50,2)
12
```

Alternativt exempel:

```
>>>import random
>>>random.randrange(10,50,2)
48
```

Resultaten kommer att variera, eftersom output är slumpmässigt.

[math] Modul 2:random… Random 6:randrange (start,stop,step)

[math] Modul 2:random… Random 6:randrange (start,stop,step)

[2nd] [catalog]

import-kommandon finns i [2nd] [catalog]

range(start,stop,step)

Modul: Built in

[2nd] [catalog]

Syntax: range(start,stop,step)

Beskrivning: Använd funktionen range för att returnera en sekvens av tal. Alla argument är valfria. Standard för start är 0, standard för steg är 1 och sekvensen slutar vid stopp.

Exempel:

```
>>> x = range(2,10,3)
>>> for i in x
… print(i)
…
…
2
5
8
```

.real

Modul: Built-in 2nd [catalog]

Syntax:var.real

Beskrivning: Returnerar realdelen av en specificerad variabel som är av taltypen komplext tal.

Exempel:

```
>>>a=complex(4,5)
>>>a.real
4
>>>a.imag
5
```

var=recall_list("name") 1-6

Modul: ti_system

Syntax:var=recall_list("name") 1-6

Beskrivning: Återkalla en fördefinierad OS-lista. Listans längd måste vara mindre än eller lika med 100.

Argument: "**name**"

För OS L1-L6

1 - 6
"1" - "6"
'1' - '6'

För OS anpassad lista "name"

----- Max 5 tecken, siffror eller bokstäver. Börja med bokstäver och bokstäver måste vara versaler.

Exempel:

"ABCDE"

"R12"

"L1" blir anpassad L1 och inte OS L1

Kom ihåg: Python är dubbel precision. Python har support för fler siffror än i OS.

Exempel:

Exempelprogram:

Skapa en lista i OS.
LIST={1,2,3}

Kör Python App.
Skapa ett nytt program AA.

```
import ti_system as *
xlist=recall_list("LIST")
print xlist
```

Kör program AA.
Shell visar output.

```
[1.0, 2.0, 3.0]
```

2nd [catalog]

2nd [rcl]
ti_system
4:var=recall_
list()

[Fns...]>Modul
eller math
4:ti_system
4:var=recall_
list()

import-
kommandon
finns i 2nd
[catalog] eller i
ti_system
Modul-menyn.

var=recall_RegEQ()

Modul: ti_system

Syntax:var=recall_RegEQ()

Beskrivning: Återkallar variabeln RegEQ från CE OS. Regressionsekvationen måste vara beräknad i OS innan RegEQ återkallas i Python App.

Exempel:

Se exempelprogram: REGEQ1.

2nd [catalog]

2nd [rcl]
ti_system
4:var=recall_
REGEQ()

[Fns…]>Modul
eller math
4:ti_system
4:var=recall_
REGEQ()

import-
kommandon
finns i 2nd
[catalog] eller i
ti_system
Modul-menyn.

.remove(x)

Modul: Built-in

Syntax: listname.remove(item)

Beskrivning: Method remove() tar bort första instansen av ett objekt från en sekvens.

Exempel:

```
>>>listA = [2,4,6,8,6]
>>>listA.remove(6)
>>>print(listA)
[2,4,8,6]
```

2nd [list]
List
7:.remove(x)

2nd [catalog]

[Fns…] > List
7:.remove(x)

return

Modul: Built-in

2nd [catalog]

Syntax: return expression

Beskrivning: Ett return-kommando definierar värdet producerat av en funktion. Python-funktioner returnerar None som standard. Se även: def function ():

[Fns…] > Func 1:def function():

Exempel:

[Fns…] > Func 2:return

```
>>> def f(a,b):
…return a*b
…
…
…
>>> f(2,3)
6
```

.reverse()

Modul: Built-in

2nd [catalog]

Syntax: listname.reverse()

Beskrivning: Vänder på ordningen av objekt i en sekvens.

Exempel:

```
>>>list1=[15,-32,4]
>>>list1.reverse()
>>>print(list1)
[4,-32,15]
```

round()

Modul: Built in

2nd [catalog]

Syntax: round(number, digits)

Beskrivning: Använd funktionen round för att returnera ett flyttal avrundat till specificerat antal siffror. Standard för siffror är 0 och returnerar närmaste heltal.

Exempel:

```
>>>round(23.12456)
23
>>>round(23.12456,3)
```

round()

```
23.125
```

S

scatter(xlist,ylist,"mark")

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.scatter(xlist,ylist,"mark")

[Fns...]>Modul eller math 5:ti_plotlib...> Draw 4:scatter()

Beskrivning: En sekvens av ordnade talpar från (xlist,ylist) ritas med den stil på markering som har specificerats. Stil och storlek på linje ställs in med plt.pen().

xlist och ylist måste vara reella flyttal och listorna måste ha samma dimension.

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

Argument:

"mark" är det markerande tecknet enligt följande:

o	filled dot (default)
+	cross
x	x
.	pixel

Exempel:

Se exempelprogram: LINREGR.

seed()

Modul: random

Syntax: seed() eller seed(x) där x är ett heltal

Beskrivning: Initialiserar slumptalsgeneratorn.

Exempel:

```
>>>from random import *
>>>seed(12)
>>>random()
0.9079708720366826
>>>seed(10)
>>>random()
0.9063990882481896
>>>seed(12)
>>>random()
0.9079708720366826
```

Resultaten kommer att variera, eftersom output är slumpmässigt.

math Modul
2:random…
Random
7:seed()

[Fns…] > Modul
2:random…
Random
7:seed()

2nd [catalog]

import-kommandon finns i 2nd [catalog]

set(sequence)

Modul: Built-in

2nd [catalog]

Syntax: set(sequence)

Beskrivning: Returnerar en sekvens som ett set. Se Python-dokumentationen för mer information.

Exempel:

```
>>> print(set("84CE"))
{'E', '8', '4', 'C'}
```

show_plot() display > [clear]

Modul: ti_plotlib

Syntax: plt.show_plot() display > [clear]

Beskrivning: Utför visningen av diagrammet enligt inställning i programmet.

show_plot() måste placeras efter alla inställningsobjekt för diagrammet. Programmets ordning för diagramobjekt föreslås av ordningen i Setup-menyn.

För hjälp med diagrammallar, från File Manager, välj [New] ([zoom]) och sedan [Types] ([zoom]) för att välja programtypen "Plotting (x,y) & Text".

Efter att ha kört programmet, rensa det visade diagrammet genom att trycka på [clear] för att återgå till Shell-prompten.

Exempel:

Se exempelprogram: COLORLIN eller GRAPH.

[2nd] [catalog]

[Fns...]>Modul eller [math] 5:ti_plotlib...> Setup 9:show_plot

[Fns...]>Modul eller [math] 5:ti_plotlib... > Draw 9:show_plot()

import-kommandon finns i [2nd] [catalog] eller i ti_plotlib Setup-menyn.

sin()

Modul: math

Syntax: sin()

Beskrivning: Returnerar sinus av x. Argumentets format är i radianer.

Exempel:

```
>>>from math import *
>>>sin(pi/2)
1.0
```

sin 3:sin()

2nd [catalog]

[Fns…] > Modul
1:math… > Trig
3:sin()

import-kommandon finns i 2nd [catalog]

sleep(seconds)

Modul: ti_system; time

Syntax: sleep(seconds)

Beskrivning: Sover i ett givet antal sekunder. Argumentet sekunder är ett flyttal.

Exempel:

Exempelprogram:

```
from time import *
a=monotonic()
sleep(15)
b=monotonic()
print(b-a)
```

Kör programmet TIME.

```
>>>15.0
```

2nd [catalog]

2nd [rcl]
ti_system
A:sleep()

[Fns…]>Modul
eller math
4:ti_system
A:sleep()

[Fns…]>Modul
eller math
3:time
2:sleep()

import-kommandon finns i 2nd [catalog] eller i ti_system Modul-menyn.

.sort()

Modul: Built-in

Syntax: listnamn.sort()

Beskrivning: Metoden sorterar en lista på platsen. Se Python-dokumentationen för mer information.

Exempel:

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>listA.sort()
>>>print(listA) #listA updated to a sorted list
[2,3,3,4,4,4,5,6,6,7,8,9]
```

2nd [list] (ovanför stat) List A:.sort()

2nd [catalog]

[Fns…] > List A:sort()

sorted()

Modul: Built-in

Syntax: sorted(sequence)

Beskrivning: Returnerar en sorterad lista från en sekvens.

Exempel:

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>sorted(listA)
[2,3,3,4,4,4,5,6,6,7,8,9]
>>>print(listA) #listA did not change
[4,3,6,2,7,4,8,9,3,5,4,6]
```

2nd [list] (ovanför stat) List 0:sorted()

2nd [catalog]

[Fns…] > List 0:sorted()

.split(x)

Modul: Built-in

2nd [catalog]

Syntax:var.split(x)

Beskrivning: Metod returnerar en lista enligt specificerad avgränsare. Se Python-dokumentationen för mer information.

Exempel:

```
>>> a="red,blue,green"
>>> a.split(",")
['red', 'blue', 'green']
```

sqrt()

Modul: math

math Modul 1:math 3:sqrt ()

Syntax: sqrt(x)

Beskrivning: Returnerar kvadratroten av x.

Exempel:

2nd [catalog]

```
>>>from math import *
>>>sqrt(25)
5.0
```

[Fns…] > Modul 1:math 3:sqrt ()

import-kommandon finns i 2nd [catalog].

store_list("name",var) 1-6

Modul: ti_system

Syntax: store_list("name",var) 1-6

Beskrivning: Lagrar en lista från exekveringen av ett Python-skript till en OS-listvariabel med namnet "name" där var är en definierad Python-lista. Listans längd måste vara mindre än eller lika med 100.

Argument: "**name**"

För OS L1-L6

1 - 6
"1" - "6"
'1' - '6'

För OS anpassad lista "name"

----- Max 5 tecken, siffror eller bokstäver. Börja med bokstäver och bokstäver måste vara versaler.

Exempel:

"ABCDE"

"R12"

"L1" blir anpassad L1 och inte OS L1

Kom ihåg: Python är dubbel precision vilket ger fler siffror än som stöds av OS.

Exempel:

```
>>>a=[1,2,3]
>>>store_list("1",a)
>>>
```

Avsluta Python App och tryck på 2nd [L1] (ovanför [1]) och entrer på start-skärmen för att se listan [L1] som {1 2 3}.

2nd [catalog]

2nd [rcl]
ti_system
3:var=store_list()

[Fns...]>Modul
eller math
4:ti_system
3:var=store_list()

import-kommandon finns i 2nd [catalog] eller i ti_system Modul-menyn.

str()

Modul: Built-in

2nd [catalog]

Syntax: str(argument)

Beskrivning: Konverterar ett "argument" till en sträng.

[Fns…]
> Type
3 :str()

Exempel:

```
>>>x=2+3
>>>str(x)
'5'
```

sum()

Modul: Built-in

2nd [list] (ovanför stat) List
9:sum()

Syntax: sum(sequence)

Beskrivning: Returnerar summan av objekten i en sekvens.

2nd [catalog]

Exempel:

```
>>>listA=[2,4,6,8,10]
>>>sum(listA)
30
```

[Fns…] > List
9:sum()

tan()

Modul: math

Syntax: tan(x)

Beskrivning: Returnerar tangens av x. Vinkelargumentet är i radianer.

Exempel:

```
>>>from math import *
>>>tan(pi/4)
1.0
```

[sin] 5:tan()

[Fns…] > Modul 1:math… > Trig 5:tan()

[2nd] [catalog]

import-kommandon finns i [2nd] [catalog]

text_at(row,"text","align")

Modul: ti_plotlib

Syntax: plt.text_at(row,"text","align")

Beskrivning: Visar "text" i diagramområdet med angiven "align".

row	integer 1 through 12
"text"	strängen klipps av om den är för lång
"align"	"left" (standard) "center" "right"
optional	1 clears line prior to text (default) 0 line does not clear

Exempel:

Se exempelprogram: DASH1.

[2nd] [catalog]

[Fns…]>Modul eller [math] 5:ti_plotlib…> Draw 0:text_at()

import-kommandon finns i [2nd] [catalog] eller i ti_plotlib Setup-menyn.

time.function

Modul: Built-in

2nd [catalog]

Syntax: time.function

Beskrivning: Använd efter import time för att använda en funktion i time-modulen.

Exempel:

Se:[Fns…]>Modul: time och ti_system modules.

title("title")

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.title("title")

[Fns…]>Modul eller math 5:ti_plotlib…> Setup 8:title()

Beskrivning: "title" visas centrerad på fönstrets översta rad. "title" kapas om den är för lång.

Exempel:

Se exempelprogram: COLORLIN.

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

ti_hub.function

Modul: ti_hub 2nd [catalog]

Syntax: ti_hub.function

Beskrivning: Använd efter import ti_hub för att använda en funktion i ti_hub-modulen.

Exempel:

Se:[Fns…] > Modul: ti_hub module.

ti_system.function

Modul: ti_system 2nd [catalog]

Syntax: ti_system.function

Beskrivning: Använd efter import ti_system för att använda en funktion i ti_system-modulen.

Exempel:

```
>>> # Shell Reinitialized
>>>import ti_system
>>>ti_system.disp_at(6,8,"texte")

texte>>>|

#syns på rad 6, kolumn 8 med Shell prompten.
```

True

Nyckelord

Beskrivning: Returnerar True när exekverat påstående är True. "True" representerar det sanna värdet av objekt av typen booleska objekt.

Exempel:

```
>>>64>=32
True
```

2nd [test] (ovanför math)

2nd [catalog]

[Fns…] > Ops
A:True

[a A #]

trunc()

Modul: math

Syntax: trunc(x)

Beskrivning: Returnerar det reella värdet av x trunkerat till ett heltal.

Exempel:

```
>>>from math import *
>>>trunc(435.867)
435
```

math Modul
1:math…
0:trunc()

2nd [catalog]

[Fns…] > Modul
1:math…
0:trunc()

import-kommandon finns i
2nd [catalog]

try:

Nyckelord

Beskrivning: Använd try kodblock för att testa om kodblocket har fel. Används även med except och finally. Se Python-dokumentationen för mer information.

2nd [catalog]

tuple(sequence)

Modul: Built-in

2nd [catalog]

Syntax: tuple(sequence)

Beskrivning: Konverterar en sekvens till en tupel. Se Python-dokumentationen för mer information.

Exempel:

```
>>>a=[10,20,30]
>>>tuple(a)
(10,20,30)
```

type()

Modul: Built-in

2nd [catalog]

Syntax: type(object)

[Fns…]>Type>6:type ()

Beskrivning: Returnerar objektets typ.

Exempel:

```
>>>a=1,25
>>>print(type(a))
<class 'float'>
>>>b=100
>>>print(type(b))
<class 'int'>
>>>a=10+2j
>>>print(type(c))
<class 'complex'>
```

uniform(min,max)

Modul: random

Syntax: uniform(min,max)

Beskrivning: Returnerar ett slumptal x (float) sådant att min <= x <= max.

Exempel:

```
>>>from random import *
>>>uniform(0,1)
0.476118
>>>uniform(10,20)
16.2787
```

Resultaten kommer att variera, eftersom output är slumpmässigt.

[math] Modul 2:random… Random 3:uniform (min,max)

[2nd] [catalog]

[Fns…] > Modul 2:random… Random 3:uniform (min,max)

import-kommandon finns i [2nd] [catalog]

wait_key()

Modul: ti_system

2nd [catalog]

Syntax: wait_key()

Beskrivning: Returnerar en kombinerad tangentkod som representerar intryckt tangent, sammanslagen med 2nd och/eller alpha. Metoden väntar på att en tangent trycks in innan den återgår till programmet.

Exempel:

Se:[Fns…]>Modul: time och ti_system modules.

while condition:

Nyckelord

[Fns…] Ctl 8:while condition:

Syntax: while condition:

Beskrivning: Exekverar kommandona i det följande kodblocket tills "condition" utvärderas till False.

2nd [catalog]

Exempel:

```
>>> x=5
>>> while x<8:
… x=x+1
… print(x)
…
…
6
7
8
```

window(xmin,xmax,ymin,ymax)

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.window(xmin,xmax,ymin,ymax)

[Fns...]>Modul eller math 5:ti_plotlib...> Setup 4:window()

Beskrivning: Definierar diagramfönstret genom att avbilda det specificerade horisontella intervallet (xmin, xmax) och vertikala intervallet (ymin, ymax) på tilldelat diagramområde (pixlar).

Den här metoden måste exekveras före några andra kommandon i ti_plotlib module exekveras.

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

ti_plotlib Properties vars, xmin, xmax, ymin, ymax uppdateras till argumentvärdena. Standardvärdena är (-10, 10, -6.56, 6.56).

Exempel:

Se exempelprogram: GRAPH.

with

Nyckelord

2nd [catalog]

Beskrivning: Se Python-dokumentationen för mer information.

xmax default 10.00

Modul: ti_plotlib

Syntax: plt.xmax default 10.00

Beskrivning: Specificerad variabel för fönsterargument definierade som plt.xmax.

Standardvärden:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Exempel:

Se exempelprogram: GRAPH.

2nd [catalog]

[Fns...]>Modul eller math 5:ti_plotlib...> Properties 2:xmax

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

xmin default -10.00

Modul: ti_plotlib

Syntax: plt.xmin default -10.00

Beskrivning: Specificerad variabel för fönsterargument definierade som plt.xmin.

Standardvärden:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Exempel:

Se exempelprogram: GRAPH.

[2nd] [catalog]

[Fns...]>Modul eller [math] 5:ti_plotlib...> Properties 1:xmin

import-kommandon finns i [2nd] [catalog] eller i ti_plotlib Setup-menyn.

yield

Nyckelord 2nd [catalog]

Beskrivning: Använd yield för att avsluta en funktion. Returnerar en generator. Se Python-dokumentationen för mer information.

ymax default 6.56

Modul: ti_plotlib

2nd [catalog]

Syntax: plt.ymax default 6.56

[Fns...]>Modul eller math 5:ti_plotlib...> Properties 4:ymax

Beskrivning: Specificerad variabel för fönsterargument definierade som plt.ymax.

Standardvärden:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

Exempel:

Se exempelprogram: GRAPH.

ymin default -6.56

Modul: ti_plotlib

Syntax: plt.ymin default -6.56

Beskrivning: Specificerad variabel för fönsterargument definierade som plt.ymin.

Standardvärden:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Exempel:

Se exempelprogram: GRAPH.

2nd [catalog]

[Fns...]>Modul eller math 5:ti_plotlib...> Properties 3:ymin

import-kommandon finns i 2nd [catalog] eller i ti_plotlib Setup-menyn.

Symboler

@

Operator

[alpha] [θ] (ovanför [3])

Beskrivning: Decorator – Se allmän Python-dokumentation för mer information.

[2nd] [catalog]

<<

Operator

[2nd] [catalog]

Syntax: x<<n

Beskrivning: Bitvis vänsterskift med n bitar.

>>

Operator

[2nd] [catalog]

Syntax: x>>n

Beskrivning: Bitvis högerskift med n bitar.

|

Operator

[2nd] [catalog]

Syntax: x|y

Beskrivning: Bitvis or.

&

Operator

[2nd] [catalog]

Syntax: x&y

Beskrivning: Bitvis and.

^

Operator 2nd [catalog]

Syntax: x^y

Beskrivning: Bitvis exclusive or.

~

Operator 2nd [catalog]

Syntax: ~x

Beskrivning: Bitvis not; bitarna i x inverterade.

x<=y

Operator

Syntax: x<=y

Beskrivning: Jämförelse; x mindre än eller lika med y.

Exempel:

```
>>>2<=5
True
>>>3<=0
False
```

math
1:math > Ops
7:x<=y

2nd [catalog]

[Fns…] > Ops
7:x<=y

[a A #]

x<y

Operator

Syntax: x<y

Beskrivning: Jämförelse; x är strikt mindre än y.

Exempel:

```
>>>6<10
True
>>>12<-15
False
```

math
1:math > Ops
6:x<y

2nd [catalog]

[Fns…] > Ops
6:x<y

[a A #]

x>=y

Operator

Syntax: x>=y

Beskrivning: Jämförelse; x större än eller lika med y.

Exempel:

```
>>>35>=25
True
>>>14>=65
False
```

math
1:math > Ops
5:x>=y

2nd [catalog]

[Fns…] > Ops
5:x>=y

[a A #]

x>y

Operator

Syntax: x>y

Beskrivning: Jämförelse; x är strikt större än y.

Exempel:

```
>>>35>25
True
>>>14>65
False
```

math
1:math > Ops
4:x>y

2nd [catalog]

[Fns…] > Ops
4:x>y

[a A #]

x!=y

Operator

Syntax: x!=y

Beskrivning: Jämförelse; x är inte lika med y.

Exempel:

```
>>>35!=25
True
>>>14!=10+4
False
```

math
1:math > Ops
3:x!=y

2nd [catalog]

[Fns…] > Ops
3:x!=y

[a A #]

x==y

Operator

Syntax: x==y

Beskrivning: Jämförelse; x är lika med y.

Exempel:

```
>>>75==25+50
True
>>>1/3==0.333333
False
>>>1/3==0.3333333 #equal to stored Python value
True
```

math
1:math > Ops
2:x==y

2nd [catalog]

[Fns…] > Ops
2:x==y

[a A #]

x=y

Operator

Syntax: x=y

Beskrivning: y är lagrad i variabel x

Exempel:

```
>>>A=5.0
>>>print(A)
5.0
>>>B=2**3 #Använd [ ^ ] på knappsats för **
>>>print(B)
8
```

sto→

math
1:math > Ops
1:x=y

2nd [catalog]

[Fns…] > Ops
1:x=y

[a A #]

\

Avgränsare

Beskrivning: Backslash-tecken.

2nd [catalog]

[a A #]

\t

Avgränsare

Beskrivning: Tabb-utrymme mellan strängar eller tecken.

2nd [catalog]

\n

Avgränsare

Beskrivning: Ny rad för att visa sträng ordentligt på skärmen.

2nd [catalog]

' '

Avgränsare

Beskrivning: Klistra in två single citattecken.

Exempel:

```
>>>eval('a+10')
17
```

2nd [mem]
(ovanför +)

2nd [catalog]

[a A #]

" "

Avgränsare

Beskrivning: Klistra in två double citattecken.

Exempel:

```
>>>print("Ok")
```

alpha ["]
(ovanför +)

2nd [catalog]

[a A #]

Bilaga

Utvalda TI-Python Built-in, nyckelord och modulinnehåll

Utvalda TI-Python Built-in, nyckelord och modulinnehåll

Built-ins

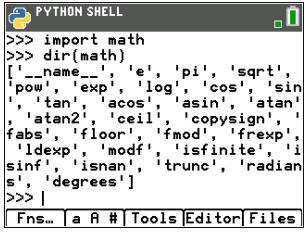
Built-ins	Built-ins	Built-ins
__name__	abs -- <function>	BaseException -- <class 'BaseException'>
__build_class__ -- <function>	all -- <function>	ArithmeticError -- <class 'ArithmeticError'>
__import__ -- <function>	any -- <function>	AssertionError -- <class 'AssertionError'>
__repl_print__ -- <function>	bin -- <function>	AttributeError -- <class 'AttributeError'>
bool -- <class 'bool'>	callable -- <function>	EOFError -- <class 'EOFError'>
bytes -- <class 'bytes'>	chr -- <function>	Exception -- <class 'Exception'>
bytearray -- <class 'bytearray'>	dir -- <function>	GeneratorExit -- <class 'GeneratorExit'>
dict -- <class 'dict'>	divmod -- <function>	ImportError -- <class 'ImportError'>
enumerate -- <class 'enumerate'>	eval -- <function>	IndentationError -- <class 'IndentationError'>
filter -- <class 'filter'>	exec -- <function>	IndexError -- <class 'IndexError'>
float -- <class 'float'>	getattr -- <function>	KeyboardInterrupt -- <class 'KeyboardInterrupt'>
int -- <class 'int'>	setattr -- <function>	ReloadException -- <class

Built-ins	Built-ins	Built-ins
		'ReloadException'>
list -- <class 'list'>	globals -- <function>	KeyError -- <class 'KeyError'>
map -- <class 'map'>	hasattr -- <function>	LookupError -- <class 'LookupError'>
memoryview -- <class 'memoryview'>	hash -- <function>	MemoryError -- <class 'MemoryError'>
object -- <class 'object'>	help -- <function>	NameError -- <class 'NameError'>
property -- <class 'property'>	hex -- <function>	NotImplementedError -- <class 'NotImplementedError'>
range -- <class 'range'>	id -- <function>	OSError -- <class 'OSError'>
set -- <class 'set'>	input -- <function>	OverflowError -- <class 'OverflowError'>
slice -- <class 'slice'>	isinstance -- <function>	RuntimeError -- <class 'RuntimeError'>
str -- <class 'str'>	issubclass -- <function>	StopIteration -- <class 'StopIteration'>
super -- <class 'super'>	iter -- <function>	SyntaxError -- <class 'SyntaxError'>
tuple -- <class 'tuple'>	len -- <function>	SystemExit -- <class 'SystemExit'>
type -- <class 'type'>	locals -- <function>	TypeError -- <class 'TypeError'>
zip -- <class 'zip'>	max -- <function>	UnicodeError -- <class 'UnicodeError'>
classmethod -- <class 'classmethod'>	min -- <function>	ValueError -- <class 'ValueError'>
staticmethod -- <class 'staticmethod'>	next -- <function>	ZeroDivisionError -- <class 'ZeroDivisionError'>

Built-ins	Built-ins	Built-ins
Ellipsis -- Ellipsis	oct -- <function>	
	ord -- <function>	
	pow -- <function>	
	print -- <function>	
	repr -- <function>	
	round -- <function>	
	sorted -- <function>	
	sum -- <function>	

nyckelord	nyckelord	nyckelord
False	elif	lambda
None	else	nonlocal
True	except	not
and	finally	or
as	for	pass
assert	from	raise
break	global	return
class	if	try
continue	import	while
def	in	with
del	is	yield

math



math	math	math
__name__	acos -- <function>	frexp -- <function>
e -- 2.71828	asin -- <function>	ldexp -- <function>
pi -- 3.14159	atan -- <function>	modf -- <function>
sqrt -- <function>	atan2 -- <function>	isfinite -- <function>
pow -- <function>	ceil -- <function>	isinf -- <function>
exp -- <function>	copysign -- <function>	isnan -- <function>
log -- <function>	fabs -- <function>	trunc -- <function>
cos -- <function>	floor -- <function>	radians -- <function>
sin -- <function>	fmod -- <function>	degrees -- <function>
tan -- <function>		

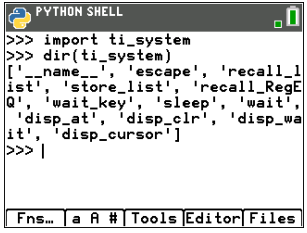
random

random	random	random
__name__	randint -- <function>	
seed -- <function>	choice -- <function>	
getrandbits -- <function>	random -- <function>	
randrange -- <function>	uniform -- <function>	

time

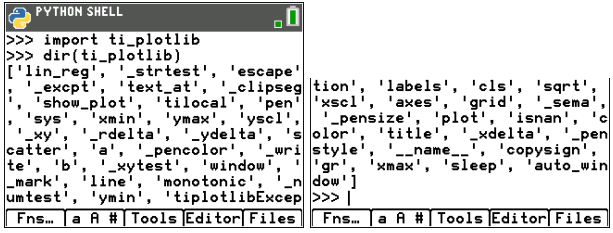
time	time	time
__name__		
monotonic		
sleep		
struc_time		

ti_system



ti_system	ti_system	ti_system
__name__	recall_RegEQ	disp_at
escape	wait_key	disp_clr
recall_list	sleep	disp_wait
store_list	wait	disp_cursor

ti_plotlib



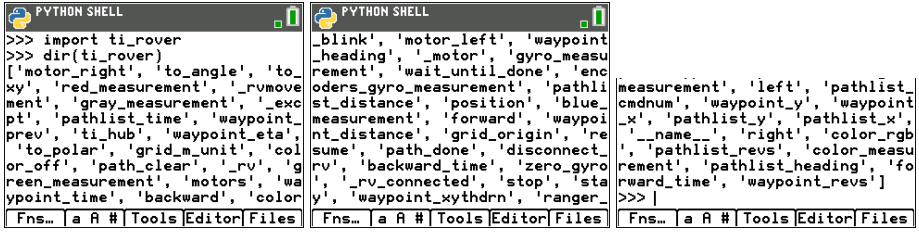
ti_plotlib	ti_plotlib	ti_plotlib
__name__	a	grid
lin_reg	_pencolor	-pensize
_strtest	_write	_sema
escape	b	-pensize
_except	_xytest	plot
text_alt	window	isnan
_clipseg	_mark	color
show-plot	line	title
tilocal	monotonic	_xdelta
pen	_ntest	_penstyle

ti_plotlib	ti_plotlib	ti_plotlib
sys	ymin	copysign
xmin	tiplotlibException	gr
ymax	labels	xmax
yscl	cls	sleep
_xy	sqrt	auto_window
_rdelta	xscl	
_ydelta	axes	
scatter		

ti_hub

ti_hub	ti_hub	ti_hub
__name__	version	last_error
connect	begin	sleep
disconnect	start	tihubException
set	about	wait
read	isti	
calibrate	what	
range	who	

ti_rover



ti_rover	ti_rover	ti_rover
__name__	color_blink	_rv
motor_right	motor_left	stay
to_angle	waypoint_heading	waypoint_xythdrn
to_xy	_motor	ranger_measurement
red_measurment	gyro_measutrment	left
rvmovement	wait_until_done	pathlist_cmdnum
gray_measurment	encoders_gyro_measurement	waypoint_y
_excpt	pathlist_distance	waypoint-x
ti_hub	position	pathlist_y
waypoint_prev	blue_measurement	pathlist_x

ti_rover	ti_rover	ti_rover
pathlist_time	forward	right
waypoint_revs	waypoint_distance	color_rgb
to_polar	grid_origin	pathlist-revs
waypoint_eta	resume	color_measurement
color_off	path_done	tiroverException
grid_m_unit	disconnect_rv	forward_time
path_clear	backward_time	pathlist_heading
green_measurement	zero-gyro	
waypoint_time	_rv_connected	
motors	stop	
backward		

.

Allmän information

Hjälp-funktion online

education.ti.com/eguide

Välj ditt land för ytterligare produktinformation.

Kontakta TI support

education.ti.com/ti-cares

Välj ditt land för teknisk och andra supportresurser.

Service- och garanti-information

education.ti.com/warranty

Välj ditt land för information om garantins längd och villkor eller om produkttjänsten.

Begränsad garanti. Denna garanti påverkar inte dina lagstadgade rättigheter.