CX II does Python!

John Hanna, johnehanna@gmail.com
Steve Phelps, sphelps31415@gmail.com
Prgmers rule the world.

Overview: Included in CX II OS version 5.2.xxx

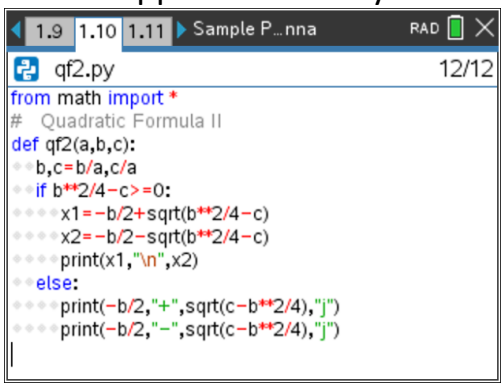
Why Python?

- popular, fast, and powerful
- *easy* to learn
- Procedural, Object-Oriented *and* Functional
- robust - mature (>30 years)
- expandable (Libraries/Packages galore)
- many platforms
- “low floor, high ceiling”
- “batteries included”
  https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020

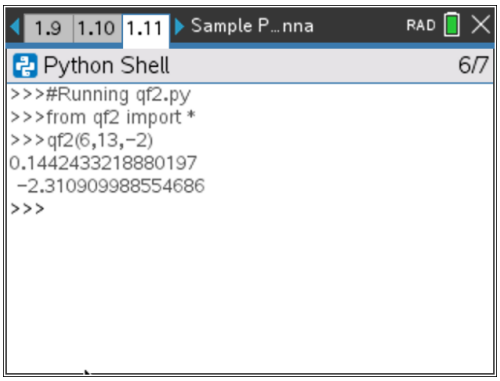
TI-NsPython

There is a new item on the 'New Page' menu: **A: Add Python**

There are **Two** new apps in the family:



Editor



Shell

The Python Experience on TI-Nspire CX II

- Designed for Middle and High School Mathematics and Science classes
- Robust set of menu items for easy 'pick-from-the-menu' programming
- Special inline prompts and tooltips when selecting commands from the menu
- MicroPython modules: math, random, cmath, time, and TI-specific modules for graphics, TI-Innovator/Rover, system
- Familiar TI-Basic shortcuts (ctrl-T, ctrl-B, ctrl-R, ...)
- Beautiful, colorful **Editor**
- **Shell**: like the Calculator app; can test expressions and write code chunks
- **Editor** and **Shell** both have special key mappings (see ctrl-=, Var).
- Full TI-Nspire integration including 'physical computing' with *simple* Innovator and Rover coding using an *Object-Oriented* approach

New to Python?

Resources:

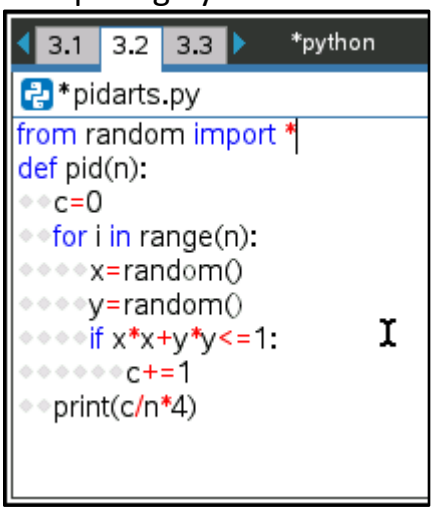
www.python.org - The home of Python
https://docs.micropython.org/ - MicroPython documentation
www.w3schools.com/python/ - Python lessons
https://trinket.io/ - online coding platform & lessons
https://www.pythonforbeginners.com/ - more lessons
https://codingbat.com/python - exercises checked online
https://en.wikipedia.org/wiki/Python_(programming_language) - info
New: TI Codes CX Python course: TI Codes
CX Python Innovator/Rover 'coming soon'

- The **Editor** is for writing Python code
- Use the **[menu]** items for inline prompts, tooltips and proper syntax
- Python is CaSe SeNsItIvE! *not* If *but* **if**
- Use **==** ("equals") for conditions (as in **if a == b:**); use **=** ("gets") for assignment (as in **a = a+1**)
- Indentation (the "off-side rule")**:** *(note the colon)*
  determines blocks (**if**, **while**, **for**, **def**, etc.). The Editor assists
- The **Shell** is the Python 'run' environment. **>>>** is the Shell "command prompt"
- Have a question? – 'google' it!

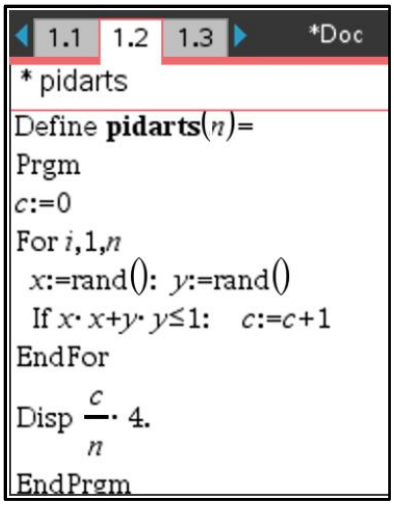
Experienced with Python?

- **Editor**: use the [menu] for easy code entry, including indent
- **Shell**: see **[menu] > Rerun last program (ctrl-R)** (unique feature)
- Can add modules that are compatible with CXII OS; can write modules
- New **Pylib** folder for 'public' modules
- See 'Under the Hood' later in this doc

For Beginners: Comparing Python and TI-Basic:



Python



TI-Basic

Your first Python program (for those needing step-by-step):

[home] > New document

Add Python > New… type **hello** for the program name [enter]

first line of code:

type **name=**

[menu] > Built-ins > I/O > input( )

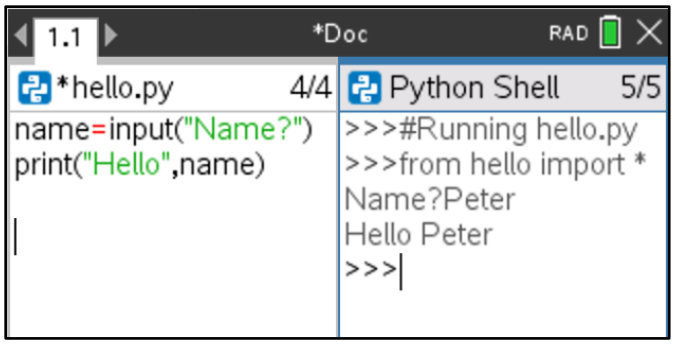
inside the ( ), type **"Name? "** after the) press [enter]

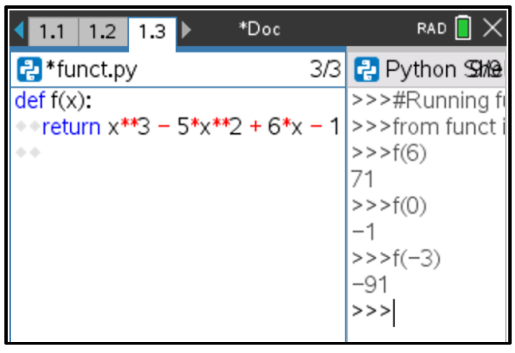
second line of code:

[menu] > Built-ins > I/O > print( )

inside the parens type **"Hello", name**

press **ctrl-r** to run…… then try this:

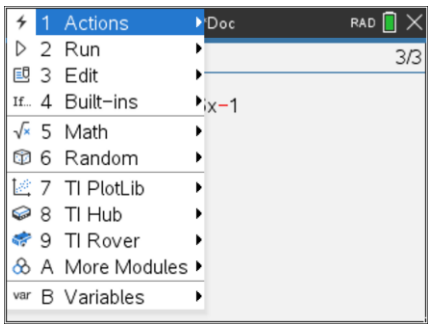




(split screens are for demo purposes showing code and result of execution)

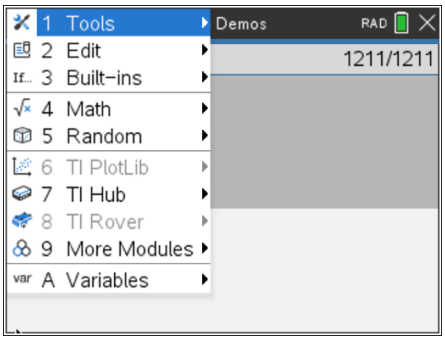
Python Editor [menu]

Not all available modules/functions are on the **Built-ins**, **Math**, and **Random** menus. The most common tools programming are available.



Python Shell [menu]

'**Run…**' lets you select the filename to run. Use this feature if/when **ctrl-r** fails (it's Python's fault!)
dir() returns the list of *names* in the current scope (Shell)
[var] shows defined identifiers (vars & functions)
Coding menus duplicated here for trying things out.
From PROGRAM import * lets you select a Python file to import



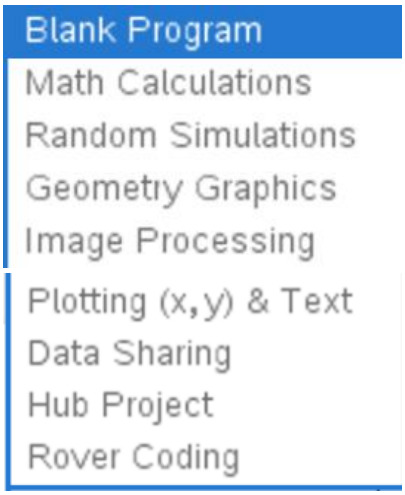
MicroPython documentation
https://docs.micropython.org/en/latest/index.html

MicroPython built-ins documentation can be found at
https://docs.micropython.org/en/latest/library/builtins.html
but not all MicroPython modules are included in CX II Python.

TI Custom Modules

ti_plotlib - facilitates quick graphing and graphical display of data
ti_hub - control TI-Innovator Hub
ti_rover - control TI-Innovator Rover
ti_system - interoperability with TI-Nspire native functions and variables
ti_draw - draw geometric graphics (like Draw in TI-Basic)
ti_image - image processing (manipulation) also good for animation

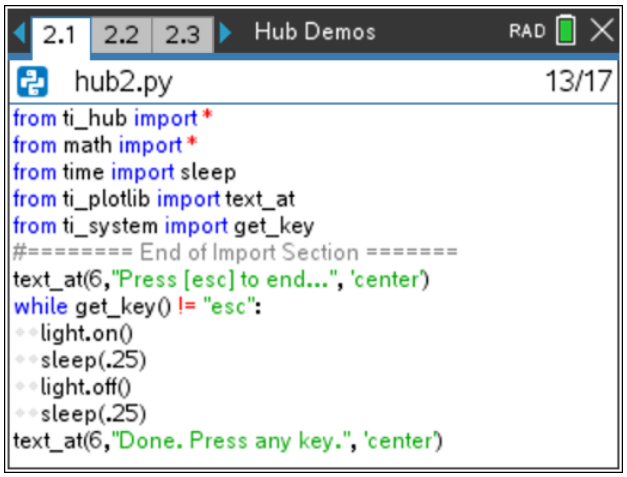
TI-NsPython Templates

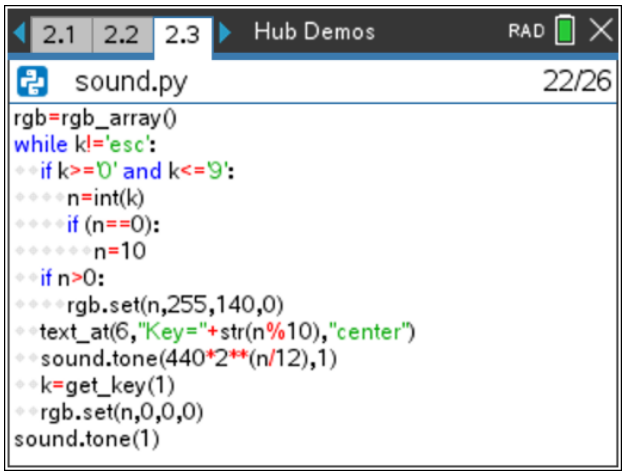


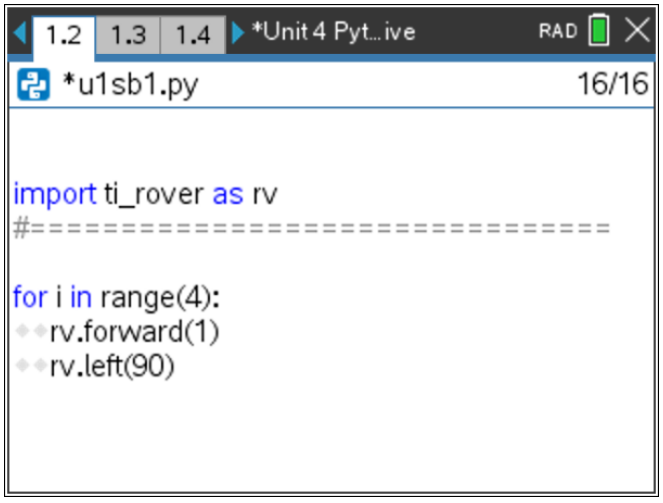
Most templates provide *only* the **import** statements for *commonly* used tools for that type of programming project. Other things can be imported by the programmer.

The Hub and Rover templates check to make sure that the TI-Innovator Hub is attached. If not, the program halts. But… there is a workaround.

Python and the TI-Innovator Hub







Under the Hood: Python Scope in a TI-Nspire Document

- Python programs can run in the Python Shell *only*.
- Python variables can be linked to TI-Nspire variables via **store_var()** and **recall_var()** functions found in **ti_system**.
- *All* Python files in a document are available to open *anywhere* in the document (use Python>Open or Shell>from PROGRAM import *)
- There is only one Shell *environment* per Problem (but with separate text histories). (See demo Canada flags.tns)
- Python programs, functions & vars are *not* TI-Nspire vars.
- Pressing [var] in Editor or Shell exposes only Python vars & functions and, in Editor, cursor position affects visible vars
- Can copy/paste Python code from other sources into/out of Premium software.
- Shell history is *not* saved with the document. Next time you open the document all Shells are there but are empty except for the Shell prompt '>>>'.
- Python *.py files are stored within the document.
- There is a new PyLib folder (similar to MyLib).