Kapitel 2: Programmera i Python	Övning 1: Hitta primtal
Här kommer vi titta lite mer på programmering i Python.	**Mål:** • Veta mer om villkorssatser • True och False. • Delbarhet

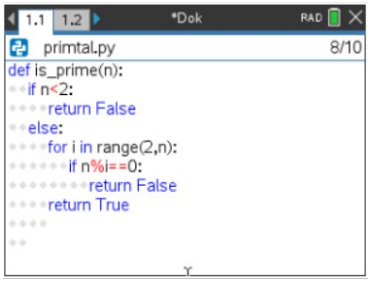
Ett primtal är ett positivt heltal som bara har två delare (1 och sig själv).
Till exempel är 7 ett primtal eftersom talet bara har två delare (1 och 7), men 10 är inte primtal eftersom 10 är delbart med 1, 2, 5 och 10. Den minsta primtalet är 2.
Vi ska skapa en funktion som kan avgöra om ett tal är ett primtal.
Resultatet av denna funktion är sant om talet är ett primtal och falskt om talet inte är ett primtal.

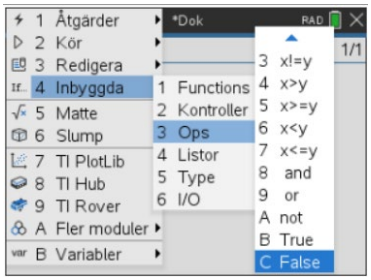
Vi skall sedan hitta det hundradade primtalet med denna funktion.

Vi kommer här att använda en operation som skrivs med %-tecknet. I Python ger % resten efter division. Tillexempel är 20%3 lika med 2 eftersom 20/3 = 6 med en rest 2 (3 går in i 20, 6 gånger och du får 2 över).

Detta betyder att om a%b = 0 så är b en delare av a eftersom b går ett helt antal in i a, med rest 0.

Börja med att definiera is_prime(n) -funktionen.
Om n <2 kan den inte vara ett primtal. Då är resultatet falskt (False).
Om n> = 2 kontrollerar vi om det finns en delare. Vi gör detta genom att testa om n är delbart med något mindre nummer (man behöver egentligen inte testa med alla mindre nummer. Vi kommer titta på det lite längre fram).
Om så är fallet är resultatet falskt, annars är resultatet sant.
(Du kan hitta **True** och **False** i menyn under inbyggda>Ops. True och False är så kallade Booelska värden och de kan användas i villkorssatser.)

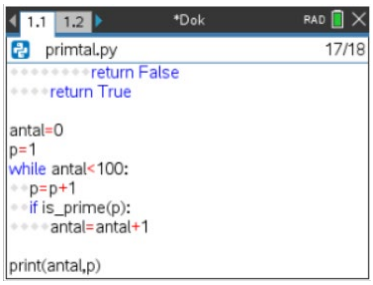




Testa med lite olika tal. Fallet 2 är lite speciellt, eftersom For-loopen skall gå från 2 till 2 (inte inklusive). Detta innebär att loopen inte körs alls och att programmet istället fortsätter med raden efter, där programmet returnerar True

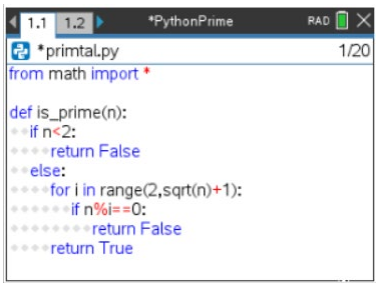
För att hitta det hundrade primtalet så skall vi skapa en while-loop.
I loopen ska vi använda en räknare, antal, som ska räkna antalet funna primtal. Vi skapar en andra variabel, p, som skall innehålla det tal som skall testas. Vi sätter den till 1.

I while-loopen skall vi testa om i<100. Om så är fallet så ökar vi p med 1.
Sedan skall vi testa om p är ett primtal. Om p är ett primtal så ökar vi antal med 1.
Sedan loopar vi igen.



Funktionen is_prime (n) i programmet är mycket långsam.
Om du till exempel vill beräkna 1000:e primtalet så tar det mycket tid, speciellt på handenheten (prova det inte ens).

Vi kan göra funktionen betydligt snabbare eftersom vi inte behöver testa alla tal upp till n inne i funktionen. Om vi testar med ett litet tal så testar vi automatiskt med ett större. Vi har tillexempel att eftersom 3 delar 27 så delar 9 talet 27 med eftersom $3 \cdot 9 = 27$. Detta innebär att vi bara behöver testa upp till roten ur n eftersom om vi delar med något större än roten ur n så blir resultatet mindre än roten ur n, som är ett tal vi redan testat.



Ändra detta i funktionen.

För att använda kvadratroten behöver vi lägga till matematikmodulen i början av programmet. Infoga denna modul med meny>matte>from math import *.

Kontrollera att programmet är snabbare nu.

Vi kan göra ytterligare ändringar i andra delen av programmet (den med while-loopen) med eftersom vi till exempel bara behöver testa vartannat tal efter p=3.