Kapitel 3: Starta programmering på riktigt — Övning 1: Tester och loopar

I denna övning ska vi undersöka s.k. primtal men innan dess så gör vi först en liten genomgång av *faktorisering*. När man förenklar ett bråk så förkortar man bort gemensamma faktorer i täljaren och nämnaren:

$$\frac{65}{20}=\frac{\cancel{5}\cdot 13}{\cancel{5}\cdot 4}=\frac{13}{4}$$

Men hur kommer man på att 65 = 5 · 13 och hur vet man att 13 och 5 inte har någon gemensam faktor?

Att skriva 65 som 5 · 13 kallas att faktorisera 65. Vissa tal kan faktoriseras på flera sätt, till exempel

$$24=6\cdot 4=2\cdot 3\cdot 2\cdot 2$$

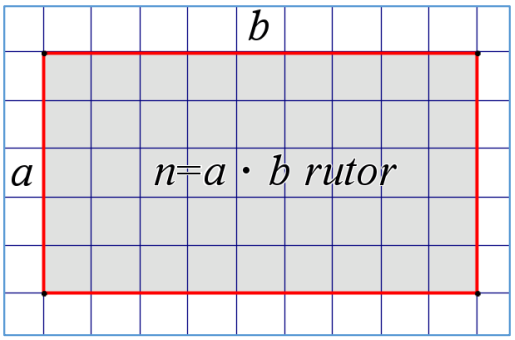
De *minsta* tänkbara faktorerna (större än ett) kallas *primtal*. Talen 2 och 3 är alltså primtal.

Definition: Ett primtal är ett heltal större än ett som inte kan faktoriseras i mindre positiva heltal. Om man kan bryta ner ett tal i mindre faktorer är det alltså *inte* ett primtal. De första primtalen är 2, 3, 5 och 7. Jämna tal (utom 2) är inte primtal eftersom de kan divideras med 2. I marginalen visar vi alla primtal mindre än 100.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

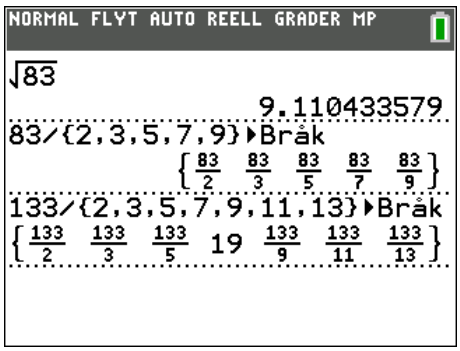
Vi ska nu titta på två program som gör faktoruppdelning. Det ena lite mer omfattande programmet arbetar med listor. Vi behöver här förklara betydelsen av symbolen %. Du hittar den genom att trycka f2 (a A #)

I konsolen (Shell) pröva nu att skriva **24%2** resp. **25%2**. I det första fallet får du resultatet 0 och i det andra fallet 1. Det som beräknas är *resten* vid division av två heltal. 24 dividerat med 2 ger ju svaret 12 medan 25 dividerat med 2 ger 12 och resten 1.



En annan viktig sak när man ska kontrollera om ett tal ***n*** är ett primtal är att man bara behöver dividera med tal som är mindre än $\sqrt{n}$. Varför då?

En viktig sats om primtal:
Kortsidan på en rektangel med *n* rutor måste ha längd som är högst $\sqrt{n}$. Med andra ord, om ett tal *n* inte är ett primtal så måste någon faktor vara högst $\sqrt{n}$. Vi prövar nu denna sats med talen 83 och 133.



Vi prövar nu denna sats på några tal. Roten ur 83 är ungefär 9 och roten ur 133 är ungefär 11. Vi gör nu beräkningarna utanför Python-appen och vi har lagt in talen i nämnaren som listor. Observera att vi bara behöver kontrollera med primtal i nämnaren. För att omvandla decimaltalen i

svaret till bråk så lägger vi till kommandot *Bråk* som du når om du trycker på tangenten [math].

Vi ser att 83 är ett primtal men att 133 inte är det. Vi kan dela upp 133 i faktorerna 7 och 19: $7 \cdot 19 = 133$.

Någon *enkel* metod för att faktorisera tal finns inte utan man måste dividera med alla mindre faktorer och kontrollera om divisionen går jämnt upp. Jobbigt att göra för hand men nu kan vi skriva ett program som fixar detta.

Här är nu ett enkelt program som löser uppgiften med faktoruppdelning av 133. Pröva gärna med några större men inte alltför stora tal. 133 går bra men när vi försöker faktoruppdela 9737 får vi svaret nedan till höger efter väldigt lång tid (flera minuter!). Hur ska vi tolka det? Efter en del funderande kommer du säkert på att den korrekta primtalsppdelningen är $7 \cdot 13 \cdot 107$. Se bilderna nedan.

```
EDITOR: FAKTOR
PROGRAM LINE 0002
from math import *
talet=133
for faktor1 in range(1,sqrt(tale
    t)):
  for faktor2 in range(faktor1,t
      alet):
    if talet==faktor1*faktor2:
      print(faktor1,faktor2)
Fns… | a A # | Tools | Run | Files
```

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running FAKTOR
>>> from FAKTOR import *
7 19
>>>
```

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running FAKTOR
>>> from FAKTOR import *
7 1391
13 749
91 107
>>>
```

Vi prövar nu ett annat program som är lite mer sofistikerat och där man använder listor och operatorn %. Vi har fuskat lite här och satt ihop två skärmar till en.

Om vi nu kör programmet får vi resultatet nedan. Vi får samma resultat som förut men nu kommer faktorerna i stigande ordning och så får vi förtydligande text också. Dessutom är det betydligt snabbare.

```
EDITOR: FAKTUPPD
PROGRAM LINE 0001
_rom math import *
t = int(input("Skriv ett tal: ")
    )
faktorer = []
n=1
for n in range(2,t):
  if t%(n)==0:
    a = (int(t/n))
    faktorer.append(a)
faktorer.reverse()
if faktorer==[]:
  print(str(t) + " är ett primtal
      ")
else:
  print("Faktorerna till " + str(
      t) + " är:")
  print(faktorer)
Fns… | a A # | Tools | Run | Files
```

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running FAKTUPPD
>>> from FAKTUPPD import *
Skriv ett tal: 9737
Faktorerna till 9737 är:
[7, 13, 91, 107, 749, 1391]
>>>
Fns… | a A # | Tools | Editor | Files
```

Lärarkommentar: Ta upp de nya instruktioner som inte förekommit i tidigare övningar. Det gäller t.ex append och reverse för listor.

Append lägger till en post i en lista. Exempel:

```
listaX=[3,5,7]
listaX.append(11)
print(listaX)
[3,5,7,11]
```

Reverse bytter ordning på posterna i en sekvens. Exempel:

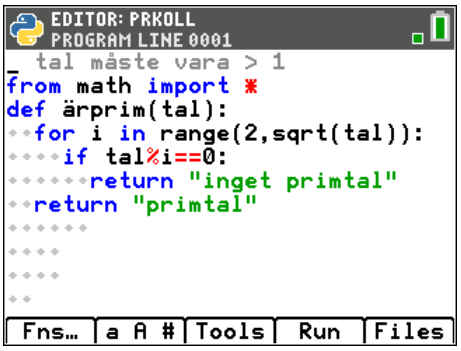
```
listaX=[3,5,7]
listaX.reverse()
print(listaX)
[7,5,3]
```

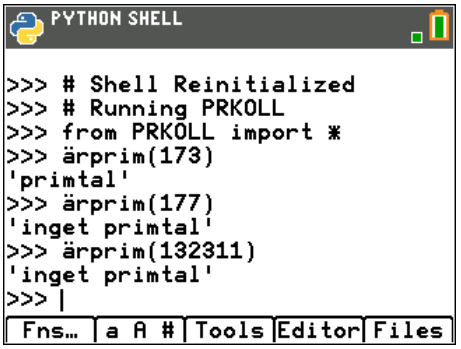
Hitta primtal

Vi övergår nu till att koncentrera oss på tal som är primtal och inte faktoruppdelning. Det första programmet testar bara om det inmatade talet är ett primtal. Enklare än så här kan det knappast bli.Pröva tal av olika storleksordning. Vi har importerat matematikmodulen *from math import*. Den behövs föatt vi vill använda kvadratsrots-funktionen sqrt().

Tänk också på att operatorn % beräknar *resten* vid division av två tal.

Tryck på tangenten [vars] och välj sedan **ärprim()** när du vill göra en körning. Skriv sedan i det tal du vill kontrollera inom parenteserna och tryck på [enter].

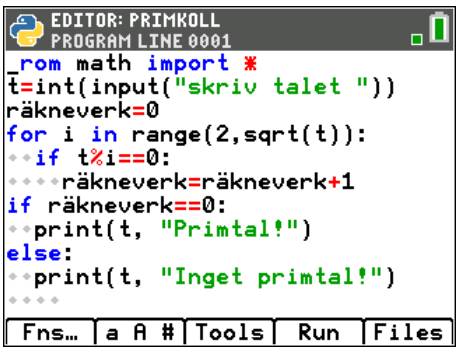




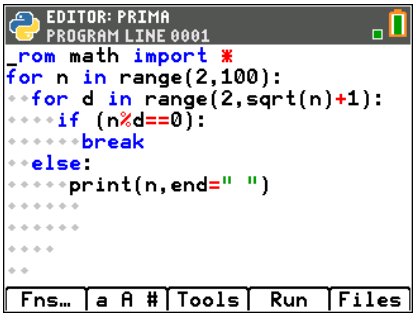
Det andra programmet har en inputsats och användaren uppmanas alltså att mata in ett tal. Vi måste då lägga in int() i satsen för att det ska fungera. Om man inte definierar att det är tal kommer den inte att kunna utföra beräkningar med värdena. Alltså: För att definiera det som ett heltal skrivs det in enligt denna mall:

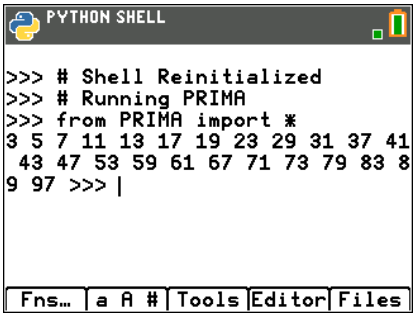
heltal = int(input(”Skriv nu in heltalet”)).

Dessa två program kan bara kontrollera ett tal i taget. Det naturliga när man håller på med primtal är att få en lista med ett antal primtal inom ett intervall t.ex.

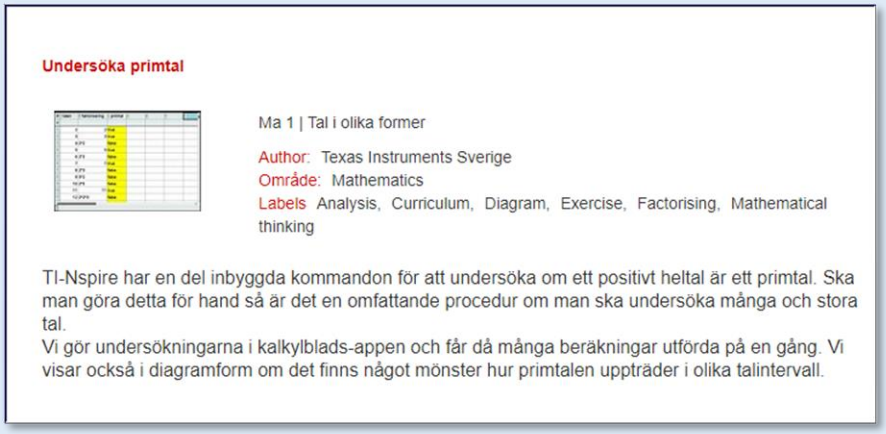


Här är ett sådant kort program som printar ut primtal inom ett intervall. Printsatsen har med instruktionen end=” ” för att primtalen ska fyllas på upp vågrätt istället för radbrytning mellan varje tal. Vi har här





Lärarkommentar: På denna sista sida så visar vi hur man kan arbeta med primtal och faktoruppdelning med TI-Nspire®. Om du har möjlighet så visa gärna detta för eleverna. Aktiviteten finns i TI:s materialdatabas och heter Undersöka primtal.



Här några skärmar från denna aktivitet. Funktioner för faktoruppdelning och undersökning om ett ett tal är ett primtal är alltså inbyggda i TI-Nspire.