Extramaterial till Matematik Z

LATHUND PYTHON

Python

I den här lathunden kan du läsa om hur du startar upp Repl.it för att sedan kunna skriva program i programmeringsspråket Python. Därefter finns en kort beskrivning av diverse begrepp som har med programmering att göra och lite allmänna tips för att klara uppgifterna. Se även ”Lathund Python with Turtle”.

Kom igång med Python

För att skriva och köra program i Python på din dator kan du ladda ned och installera Python tillsammans med en så kallad editor. Allt detta finns gratis på webbplatsen www.python.org.

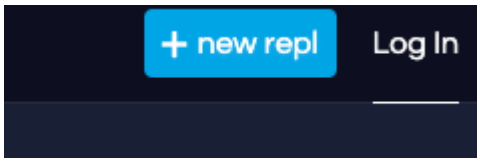
Installera aldrig Python eller andra program om du inte först har tillåtelse från datorns ägare.

Om man inte vill eller kan installera något på sin dator, finns det webbsidor där man kan arbeta med Python direkt i webbläsaren. Exempel på sådana sidor är repl.it och trinket.io. Om man skaffar ett konto kan man spara sina projekt. Fördelen med repl.it gentemot trinket.io är att man kan använda editorn även om man inte skapat något konto.

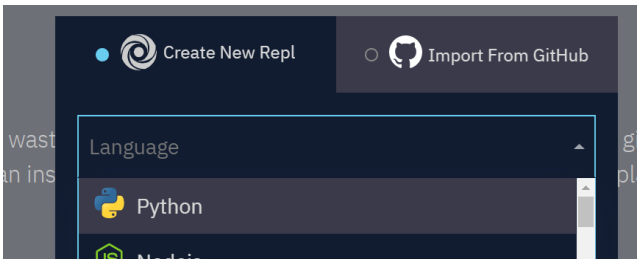
REPL.IT

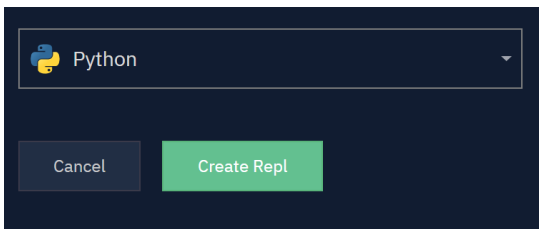
Gå in på repl.it

Längst upp till höger på sidan klickar du på ”+new repl”

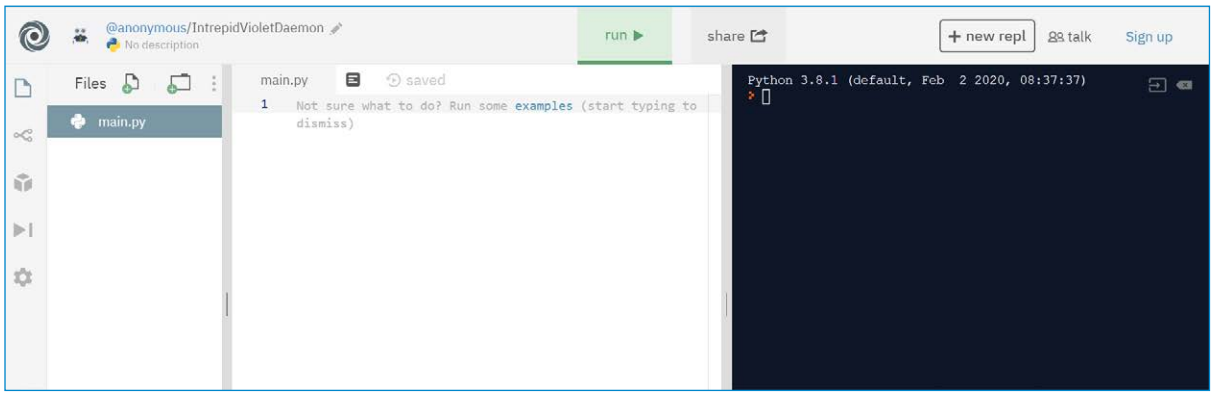


Välj sedan ”Python” och klicka på ”Create repl”.



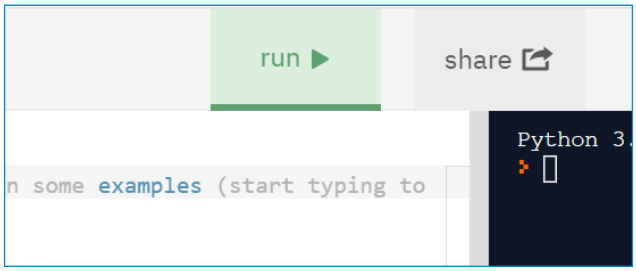


Då öppnas följande fönster:

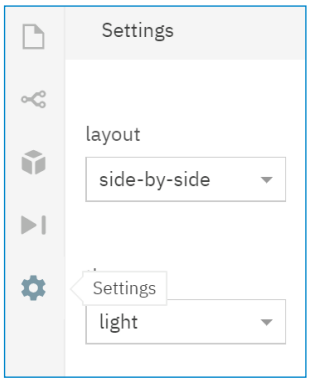


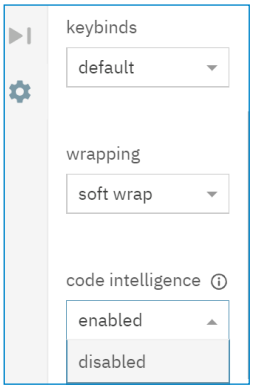
Till vänster finns bland annat verktygsfältet och i mitten själva *editorn*. Där skriver man sitt program.

När man är färdig och vill köra sitt program klickar man på ”run”. Då körs/visas programmet i det högra fönstret.



I editorn finns en inbyggd funktion som innebär att den gissar vad man tänker skriva och ger förslag. När man är ovan som programmerare kan den funktionen vara förvirrande. I så fall kan den stängas av genom att välja settings (kugghjulet) längst till vänster och sedan ”Code intelligence”. Markera ”disabled”.





Programmering i Python – begrepp och tips

VARIABLER

När man programmerar använder man sig ibland av **variabler**. Det kan vara en eller flera bokstäver eller till exempel ett ord som man ger ett speciellt värde:

```
a = 4
MittTal = 4
```

I det första exemplet kallar vi variabeln för **a** och ger den värdet 4. I det andra exemplet kallar vi den istället för **MittTal** och ger den värdet 4.

Tänk på att försöka döpa variablerna till något relevant. Det förenklar när man skriver större program och när man felsöker. Stora program kan innehålla många variabler.

Om man till exempel skapar en variabel som är ett slumpat tal kan det vara bra att döpa variabeln till just **SlumpatTal** eller **Slumpat_tal**.

Observera att man inte kan använda mellanslag, bindestreck eller bokstäverna å, ä eller ö i variabelnamnet.

LOOPAR

En loop används om en del av koden ska upprepas ett visst antal gånger. Vad som är viktigt att tänka på är att alltid avsluta beskrivningen av loopen med kolon och att allt som ingår i loopen måste tabbas in:

```
for i in range(2):
    slumpat_tal = randint(1,5)
    print(slumpat_tal)
```

Ibland behöver man placera en loop inuti i en annan loop. Då kan det se ut så här:

```
for i in range(6):
  for i in range(4):
    forward(100)
    left(90)
  left(60)
```

VILLKOR, IF/ELSE

”If” (om) anger vad som ska utföras om/när ett villkor är uppfyllt. Ofta följs ”if” av ”else” (annars). If/else-satserna avslutas alltid med kolon och det som ingår i villkoret ska tabbas in:

```
if(a%b==0):
    print("delbart med 2")
else:
    print("ej delbart med 2")
```

FUNKTIONER

Om någon del i ett program upprepas flera gånger, kan man skapa en funktion, för att slippa upprepa koden gång på gång. Koden blir då tydligare och inte lika svår att överblicka. Det underlättar även om man vill ändra delar av koden. Då behöver man bara ändra i koden där funktionen definieras.

Exempel: Koden för hur man kan rita en stjärna med Turtle ser ut så här:

```
for i in range(5):
        forward(100)
        left(144)
```

Istället för att skriva koden för hur en stjärna ritas varje gång man vill rita en stjärna, kan man skapa och definiera en funktion:

```
def rita_stjarna():
    for i in range(5):
        forward(100)
        left(144)
```

Nästa gång man vill rita en stjärna i programmet, skriver man bara:

```
rita_stjarna()
```

Om man ändrar sig och vill rita mindre stjärnor, ändrar man bara i definitionen:

```
for i in range(5):
        forward(50)
        left(144)
```

TYDLIGHET

Det spelar ingen roll om man använder varje rad eller inte i editorn. Utfallet av programmet blir det samma. Fördelen med att lämna mellanrum är att koden blir mer lättöverskådlig och den blir enklare att debugga (felsöka). Jämför själv:

```
1  from random import randint
2
3  n = int(input("Ange antalet tärningskast: "))
4
5  for i in range(n):
6      Slumpat_tal = randint(1, 6)
7      print(Slumpat_tal)
```

vs

```
1  from random import randint
2  n = int(input("Ange antalet tärningskast: "))
3  for i in range(n):
4      Slumpat_tal = randint(1, 6)
5      print(Slumpat_tal)
```

OPERATORER

Ibland när man programmerar vill man kunna göra beräkningar genom att sätta samman variabler och/eller tal med en *operator*. En operator är själva tecknet som visar vilket räknesätt som ska användas eller vilken beräkning som ska genomföras.

Exempel:

1 + 3
a – 7

Operatorer för de fyra räknesätten:

+	Addition
–	Subtraktion
*	Multiplikation
/	Division

Andra användbara operatorer är:

**	Upphöjt till	**Exempel:** 3**3 = 27
math.sqrt()	Kvadratroten ur	**Exempel:** math.sqrt(25)=5 (mer om math nedan)
//	Kvoten av heltalsdivision	**Exempel:** 10/3 ger svaret 3 Även 11/3 ger svaret 3. Svaret avrundas alltså inte uppåt.
%	Anger resten vid heltalsdivision	**Exempel:** 9%3 ger svaret 0, medan 10%3 ger svaret 1 och 11%3 ger svaret 2.
*math.pi	Multiplicerat med pi.	**Exempel:** 10*math.pi ger svaret 31,41592...

BIBLIOTEK

För att kunna genomföra vissa räkneoperationer kan man behöva importera så kallade bibliotek/delar av bibliotek.

För att kunna hantera slumptal (heltal) behöver man importera ”randint” från biblioteket ”random”:

```
from random import randint
```

För att kunna använda ”roten ur” och ”pi” behöver man importera ”math”:

```
import math
```

För att importera allt från turtle skriver man:

```
from turtle import *
```

PRINT

För att resultatet av ens program ska synas på skärmen, behöver man lägga till ”print”.

Om man till exempel endast skriver:

```
from random import randint
a=randint(1,3)
```

så slumpas ett tal mellan 1 och 3 fram, men man får inte veta vilket tal som slumpats fram. Därför behöver man lägga till ”print”.

```
from random import randint
a=randint(1,3)
print(a)
```

Print används också om man vill skriva ett meddelande eller instruktioner till den som kör programmet, exempelvis:

```
print(”Hej! Nu ska du få spela ett spel”)
```

eller

```
print(”Välj en siffra mellan 1 och 10”)
```