Extramaterial till
Matematik Z

NIVÅ
TVÅ
TRE

Med sikte på framtiden – Sannolikhet

ELEV

I den här uppgiften kommer du att få arbeta med programmeringsspråket Python när du skriver program som slumpar fram tal och beräknar den relativa frekvensen för olika utfall.

Fler programmeringsuppdrag i Python finner du bland uppgifterna i ”Programmering och digital kompetens” som hör till Matematik X, Y respektive Z. I ”Lathund Python” finns mer om programmering i Python

SYFTE

Syftet med övningen är att du ska

- få erfarenhet av att skriva kod i programmeringsspråket Python.
- få erfarenhet av att använda en editor.
- kunna slumpa fram tal med hjälp av Python.
- använda Python för att presentera tal du slumpat fram.
- beräkna den relativa frekvensen utifrån tal du slumpat fram.
- diskutera skillnaden mellan beräknad sannolikhet och det verkliga utfallet.
- lösa problem med hjälp av kod.

REDOVISNING/BEDÖMNING

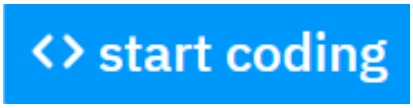
Du redovisar genom att lämna in svar på uppgifterna och koden till ditt/dina program till din lärare.

DEL 1: Python och slumptal

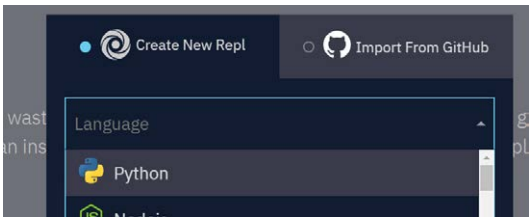
SLUMPA FRAM ETT TAL - UPPGIFT A

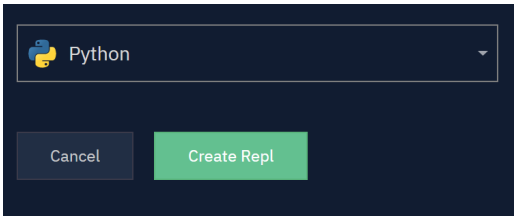
1. Gå in på sidan repl.it

   Längst upp till höger på sidan klickar du på ”<> start coding”

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

2. För att kunna hantera slumpade heltal behöver du börja med att importera ”randint” från *biblioteket* random. Det gör du genom att skriva ”from random import randint” (”-int” står för integer som betyder heltal).

   ```
   main.py   saved
   1  from random import randint
   2
   ```

3. I nästa steg ska du skapa en variabel och beskriva den. För tydlighetens skull kallar vi variabeln ”slumpat_tal” eftersom det handlar om en variabel som vi vill ge ett värde som är ett slumpat heltal mellan 1 och 3.

   Skriv följande på nästa rad:

   ```
   slumpat_tal = randint(1,3)
   ```

4. Klicka på ”run” för att köra ditt program. Vad händer?

5. Om du inte får ett felmeddelande i rutan till höger, fungerar din kod och ett tal har slumpats fram. Problemet är att du inte kan se vilket tal datorn slumpat fram.

   För att kunna göra det behöver du ”säga till” datorn att visa det slumpade talet. För att det ska ske, behövs kommandot ”print”.

   Skriv följande på nästa rad:

   ```
   print(slumpat_tal)
   ```

   ```
   1  from random import randint
   2
   3  slumpat_tal = randint(1,3)
   4
   5  print(slumpat_tal)
   ```

Resultatet blir samma, om man använder varje rad eller som här till vänster, lämnar mellanrum. Fördelen med att lämna mellanrum är att koden blir tydligare och enklare att debugga.

6. Kör programmet igen. Vad händer?
7. Kör programmet flera gånger och notera vad som händer. Verkar programmet fungera? Slumpas ett tal mellan 1 och 3 fram?
8. Ändra programmet så att ett heltal mellan 1 och 5 slumpas fram.
9. Testa programmet så det fungerar.
10. Om du kör programmet 10 gånger, hur många femmor borde du få? Kör programmet 10 gånger och jämför utfallet med den beräknade sannolikheten för att få ”5”.

SLUMPA FRAM FLERA TAL - UPPGIFT B

1. Tänk dig att du slumpar fram tal mellan 1 och 5. Vad är sannolikheten att få ”4”?
2. Om du slumpar fram tio tal mellan 1 och 5, hur många av dem *borde* vara fyror?
3. I uppgift A slumpade du fram ett tal tio gånger, men tänk om du vill slumpa fram 100 eller fler tal. Då tar det både tid och kraft att köra programmet om och om igen.

   Av den anledningen ska du nu ta hjälp av en så kallad ”loop”. Du kommer börja med att upprepa loopen två gånger.

   Efter den inledande raden, skriver du:

   ```
   for i in range(2):
   ```

4. Vad är det som ska upprepas två gånger? Jo, ett tal ska slumpas fram och datorn ska skriva ut talet. Allt som ska ingå i loopen ska ”tabbas in” (se bild).

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

```
1   from random import randint
2
3   for i in range(2):              # loop som upprepas två gånger
4     slumpat_tal = randint(1,5)    # variabeln slumpat_tal får värdet av ett
                                      slumpat tal mellan 1 och 5
5     print(slumpat_tal)            # värdet av slumpat_tal syns på skärmen
6
```

5. Kör programmet. Vad händer?
6. Korrigera programmet så att tio stycken heltal mellan 1 och 5 slumpas fram.
7. Kör programmet och kontrollera att det fungerar.
8. Vad fick du för resultat?
9. Använd ditt resultat för att beräkna andelen fyror i procent. Jämför med dina svar på fråga 1 och 2. Diskutera eventuella skillnader med en kompis.

10. Hur många fyror ”borde” du få om du slumpar fram 20 stycken tal mellan 1 och 5?
11. Ändra programmet och testa.
12. Diskutera och jämför resultatet med uppgift 9. Blir det någon skillnad om du slumpar fram 100 tal? Testa!

DEL 2: Python, slumptal, räknare och relativ frekvens

SLUMPA FRAM TAL MED RÄKNARE - UPPGIFT A

1. Ju fler tal du slumpar fram desto svårare och längre tid tar det att leta efter och räkna alla fyror. Därför är det dags att lägga till lite kod, för att låta programmet räkna åt dig.

   Börja med att skapa en ny variabel, ”antal_fyror”, och ge den värdet noll:

   antal_fyror=0

   Att den får värdet noll är för att antalet fyror är noll innan programmet körs.

```
from random import randint

antal_fyror=0
```

2. Därefter körs loopen som du använde i tidigare uppgift:

```
from random import randint

antal_fyror=0

for i in range(20):
    slumpat_tal = randint(1,5)
    print(slumpat_tal)
```

3. För varje fyra, som slumpas fram, vill du att värdet av variabeln ”antal_fyror” ska öka med 1. Nästa steg blir att lägga till ett villkor som innebär att OM det slumpade värdet är lika med fyra, så ska ”antal_fyror” öka med ett.

   Lägg till villkoret i loopen. Håll reda på indrag, parenteser och kolon.

```
antal_fyror=0

for i in range(n):
    slumpat_tal = randint(1,5)
    print(slumpat_tal)
    if slumpat_tal==4:
        antal_fyror=antal_fyror+1
```

Varför dubbla likhetstecken? I det här fallet blir betydelsen av `if(slumpat_tal==4):` ”Om det slumpade talen är lika med fyra, så…”. Om du använder **ett** likhetstecken så tilldelar du en variabel ett värde, i det här fallet ”4”. Då kommer inte loopen att fungera.

4. Kör programmet. Vad händer?
5. Om du inte fått något felmeddelande slumpas 20 heltal mellan 1 och 5 fram, men du kan inte se antalet fyror. För att det ska presenteras måste du lägga till att resultatet ska skrivas ut. Det gör du med kommandot ”print(antal_fyror)”. Eftersom det inte tillhör loopen utan ska genomföras när loopen är färdig, ska **inte** ”print(antal_fyror)” tabbas in.
6. Kör programmet igen. Vad händer? Kan du se antalet fyror nu? Stämmer det?
7. För att göra det ännu tydligare, byt ut ”print(antal_fyror)”mot ”print(”antal fyror=”, antal_fyror)” och kör programmet igen.

```
1   from random import randint
2
3   antal_fyror=0
4
5   for i in range(20):
6       slumpat_tal = randint(1,5)
7       print(slumpat_tal)
8       if slumpat_tal==4:
9           antal_fyror=antal_fyror+1
10
11  print("Antal fyror",antal_fyror)
```

8. Vad är skillnaden mellan

```
print(antal_fyror)
```
och
```
print(”Antal fyror”,antal_fyror)?
```

SLUMPA FRAM TAL: RELATIV FREKVENS - UPPGIFT B

1. Använd dig av kunskaperna från uppgift A och skapa ett program där 100 heltal mellan 1 och 5 slumpas fram.

   Antalet ettor, tvåor, treor osv ska redovisas längst ned.
2. När du fått programmet att fungera är det dags att utveckla det genom att lägga till kod så att även den **relativa frekvensen**, dels i decimalform dels i procent, för varje värde beräknas och presenteras.

TIPS:

- Hur räknar du ut den relativa frekvensen om antalet fyror är 5 stycken när du slumpat fram 20 värden?
- Antalet fyror = **`(antal_fyror)`**
- Hur många tal har du slumpat fram?
- För att göra matematiska beräkningar använder man något som kallas för operatorer

addition	+
subtraktion	-
multiplikation	*
division	/

DEL 3: Python med slumptal, räknare och input

Tänk dig att du vill skapa ett program där du ska slumpa fram resultatet av ett visst antal tärningskast. Du vill låta användaren av programmet få välja hur många gånger ”tärningen” ska kastas.

INPUT - UPPGIFT A

Fortsätt att arbeta med koden från Del 2:

```
1   from random import randint
2
3   antal_fyror=0
4
5   for i in range(20):
6       slumpat_tal = randint(1,5)
7       print(slumpat_tal)
8       if slumpat_tal==4:
9           antal_fyror=antal_fyror+1
10
11  print("Antal fyror",antal_fyror)
```

1. Lägg till följande rad, för att användaren ska kunna skriva in hur många tal som ska slumpas fram:

```
n = int(input(”Ange antalet tal som ska slumpas fram: ”))
```

```
1   from random import randint
2
3   n = int(input("Ange antalet tal som ska slumpas fram: "))
```

Bokstaven ”n” är en variabel. Vilket värde som n får beror på vad användaren skriver efter att ha fått instruktionen ”Ange antalet tal som ska slumpas fram:”.

Blir svaret ”10” kommer n att tilldelas värdet 10, om svaret är 100 tilldelas n värdet 100 osv.

2. Kör programmet. Vad händer?
3. Testa programmet några gånger, genom att tilldela n några olika värden. Stämmer resultatet? Om inte, vad kan det bero på?

TIPS:

Titta på den första loopen. Vad står siffran inom parentesen för (i det här fallet 5)?

```
7    for i in range(5):
```

Vad ska siffran bytas ut mot för att rätt antal tal ska slumpas fram?

SIMULERA TÄRNINGSKAST OCH PRESENTERA RESULTATET - UPPGIFT B

Skapa ett program som simulerar tärningskast.

VILLKOR:

- Användaren ska själv få välja hur många gånger tärningen ska kastas.
- Resultatet av de slumpade kasten ska visas.
- Antalet ettor, tvåor, treor osv ska summeras och redovisas.
- Utveckling: Den relativa frekvensen för ettor, tvåor, treor osv ska också redovisas.