Programmering i Python

	F1	Grunder i Python
⇨	**F2**	Loopar, moduler
	F3	Funktioner, läsa data, dictionaries
	F4	Dictionaries, API:er/requests
	F5	?

Repetition

- int, float, str, list
- print()
- input()
- variabler
- If-satser
- Frågor på detta
- Lösningar på uppgifterna från förra lektionen + frågor

Mer om if-satser

```
if age < 12:
    print(“Du är för ung för att åka Jetline”)
elif age > 80:
    print(“Du är för gammal för att åka Jetline”)
else:
    print(“Välkommen in till Jetline”)
```

Kan lägga till hur många elif som helst, men bara en else för varje if-sats.

Kan ha en if-sats i en if-sats:

```
if age == 25:
    if name == “Elvira”:
        print(“Du är Elvira och 25 år gammal”)
    else:
        print(“Du är 25 år gammal men heter inte Elvira”)
```

For-loopar på listor

Låt säga att vi vill utföra samma operation på alla element i en lista

Exempel:
Vi vill printa alla siffror i en lista på en varsin rad

```
lista = [1, 2, 3, 5, 6, 3, 2, 6]
```

Jobbigt att göra:

```
print(lista[0])
print(lista[1])
print(lista[2])
```

Osv… om det är en lång lista eller om vi inte vet längden på listan innan

Istället använder vi en “for”-loop:

```
for siffra in lista:
    print(siffra)
```

For-loopar på en lista med strängar

```
städer = [“Göteborg”, “Malmö”, “Stockholm”, “Uppsala”, “Sundsvall”]

for stad in städer:
    print(“Jag har bott i”, stad)
```

For-loopar på ett spann (range)

Låt säga att vi vill upprepa något ett visst antal gånger gånger, t.ex. Printa “Hej” 1000 gånger. Jobbigt om vi skulle behöva en 1000 element lång lista.
range(n) skapar en ‘lista’ som är n element lång
Ex. `range(1000) = [0, 1, 2, …, …, 998, 999]`

```
for i in range(1000):
    print(“Hej nummer “, str(i) )
```

Kommer printa

```
Hej nummer 0
Hej nummer 1
…
Hej nummer 999
```

For-loopar som hoppar flera steg i taget

`range()` kan också ta in fler siffror: `range(start, stop, step)`.
`start:` startvärdet vår variabel (i) börjar på
`stop:` värdet vår variabel (i) slutar innan
`step:` antalet steg vår variabel (i) ökar med varje gång vi upprepar

```
for i in range(0, 1000, 2):
    print(“Hej nummer“, str(i) )
```

Kommer printa:

```
Hej nummer 0
Hej nummer 2
Hej nummer 4
…… osv
Hej nummer 998
```

For-loopar som räknar baklänges

`range()` kan också gå baklänges:
`range(start, stop, step).`

Ex: loopa från 100 till 1

```
for i in range(100, 0, -1):
    print(“Hej nummer“, str(i) )
```

Kommer printa:

```
Hej nummer 100
Hej nummer 99
Hej nummer 98
…… osv
Hej nummer 1
```

Fråga

Hur ska loopen se ut om vi vill räkna nedåt från 10 till 1?

Summera (plussa ihop) alla tal från 0 till 10

```
resultat = 0
for i in range(11): # i kommer gå från 0 till och med 10
    resultat = resultat + i

print(“resultatet blev”, resultat)
```

Varför skriver vi range(11 och inte range(10)?

If-satser i for-loopar

Låt säga att vi har en lista med de 5 största städerna i Sverige (strängar)
och att vi vill printa alla städer som har mer än 8 bokstäver

Idé:

1. Gör en for-loop som går igenom hela listan
    a. Printa staden om dess ordlängd är >8

```
cities = [“Norrköping”, “Stockholm”, “Malmö”, “Göteborg”, “Jönköping”]

for city in cities:
    if len(city) > 8:
        print(city)
```

For-loopar i for-loopar?

Två for loopar efter varandra:

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

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

En for-loop i en annan for-loop:

```
for i in range(5):
    for j in range(5): # observera j och inte i
        print(i)
```

Skillnad? Testa och försök förstå.

While-loopar

Används om vi vill upprepa något medan ett villkor stämmer.
T.ex. Vi vill addera 1 till ett tal fram tills att det talet är större än 10.

```
tal = 1
while tal <= 10:
    tal = tal + 1
```

Vad hade hänt om vi hade skrivit `while tal > 0` ?
Oändlig loop!

`break` kan användas för att komma ut ur en loop. Ex:

```
while True:
    if input(“Skriv något”) == “quit”:
        break
    else:
        print(“Jag loopar på!”)
```

While-loopar

Vi skapar ett program där vi ber användaren om sina familjemedlemmar som vi lägger i en lista,
Fram tills att användaren skriver “done”

```
lista = []
namn = input(“Skriv namn på en familjemedlem. Skriv ‘done’ när du är klar: ”)

while namn != “done”:
    lista.append(namn)
    namn = input(“Skriv ett till namn (avsluta med ‘done’): ”)


# Printa alla namn som användaren gav oss, efter while-loopen är klar
print(lista)
```

När stoppas vår while-loop? Varför utförs print bara en gång?

Fråga

Är detta en oändlig loop?

```
count = 0

while count < 10:
    print(“Hej!”)
```

Moduler

Python har många inbyggda funktioner

Bra - man behöver inte uppfinna något som redan någon annan uppfunnit

Det finns även en hel del funktioner som man lätt kan importera till sitt pythonprogram

Random

Ibland vill man generera slumptal. För att göra det i python så måste man importera modulen random.

```
import random
```

Slumpvis tal mellan 0 tom 9:

```
random.randrange(10)
```

Slumpvis tal mellan 2 tom 10:

```
random.randint(2, 10)
```

Slumpvis värde från en lista:

```
random.choice([“en”, “lista”, “här”])
```

Timer delay

Om man vill “pausa” sitt program, eller göra något annat med tid så finns det en modul för det med: time

```
import time
```

Vänta i en sekund:

```
time.sleep(1)
```

Få current time (unix time):

```
time.time()
```

En lösenordsgenerator

Vi bygger tillsammans ett program som genererar säkra lösenord.

1. Vi behöver skapa en sträng med tillåtna tecken
2. Vi frågar användaren hur långt lösenordet ska vara o sparar i en variabel
3. Vi använder random-modulen för att välja rätt antal tecken slumpvis
4. Vi printar ut lösenordet till användaren

Uppgifter, Presentationen finns på github.com/mrkickling/pythonkurs

1. Skapa en lista med namn (strängar) och printa ut alla namn på en varsin rad

2. Skriv ut alla jämna tal från 0 till och med 100

3. Summera (plussa ihop) alla tal från 0 till och med 1000 och printa resultatet

4. Gör en nedräknare till en raketuppskjutning, som räknar/printar ned från 10 till 1 och väntar en sekund mellan varje siffra och sedan printar “LIFT OFF!” (använd sleep)

5. Skapa ett gissningsspel:
   a. Programmet genererar ett hemligt slumptal och sparar i en variabel
   b. Användaren gissar på talet
   c. Programmet svarar ifall det hemliga talet är större eller mindre än gissningen
   d. Upprepa steg b och c tills användaren gissat rätt

6. Skapa sten sax påse där programmet väljer ett alternativ och användaren väljer ett. Vinnaren skrivs sedan ut på skärmen. Printa tydliga instruktioner till användaren. EXTRA: gör sten sax påse med, men bäst av tre