Samevatting Python:

# Ingebouwde functies:

## Type casting:

* **int()** geeft de parameter waarde terug als integer (indien noodzakelijk afgerond naar beneden)
* **float()** geeft de parameter waarde terug als float (waarbij .0 indien noodzakelijk wordt toegevoegd)
* **str()** geeft de parameter waarde terug als string

## Rekenkundige operatoren:

* Optelling: +
* Aftrekking: -
* Vermenigvuldiging: \*
* Machtsverheffing: \*\*
* Deling: /
* Geheeltallige (integer) deling: //
* Modulo (rest): %

## Ingebouwde wiskundige functies:

* Abs() krijgt numerieke parameter, als positief is wordt hij geretourneerd. Als negatief is, wordt waarde vermenigvuldigd met -1.
* Max() krijgt twee of meer numerieke parameters en retourneert de hoogste.
* Min() retourneert laagste parameter.
* Pow() krijgt twee numerieke parameters en retourneert de eerste verheven tot de macht weergegeven door de tweede.
* Round() krijgt numerieke parameter die wiskundig wordt afgerond.

# Modules:

## Module math:

Module met nog meer wiskundige functies.

* vb.  
  *Import math  
  Math.sqrt(4)*
* vb.  
  *From math import sqrt  
  Sqrt(4)*
* vb.  
  *From math import sqrt as squareroot  
  Squareroot(100)*(from <modulenaam> import <functienaam> as <nieuwenaam>, laat toe functie onder andere naam te gebruiken)
* vb.  
  *from math import ceil, floor  
  print(ceil(2.33)) # naar boven afronden  
  print(floor(2.33)) # naar beneden afronden*

## Module datetime:

* vb.  
  *from datetime import date  
  print(date.today())*
* vb.  
  *print(datetime.now())*
* vb. datum volgens bepaald format weergeven  
  *geboortedag = date(1996, 8, 14)  
  print(“Ik ben geboren op “ + geboortedag.strftime(%d/%m/%y))*

## Module time:

* vb. uitvoering code pauseren  
  *import time  
  Print(“ik ga nu 2 sec slapen”)  
  Time.sleep(2) # pauseert uitvoering van code met 2 seconden  
  Print(“ik ben weer wakker”)*
* vb. uitvoering chronometreren  
  *print(“ik ga drie seconden slapen”)  
  start = time.time() # retourneert huidige tijdstip  
  time.sleep(3)  
  einde = time.time()  
  duur = einde – start  
  print(“ik heb “ + str(round(duur)) + “ seconden geslapen”)*

## Module random:

Willekeurige getallen genereren

* vb.  
  *from random import random  
  random()*
* vb. willekeurig getal tussen 0 en 10  
  *random() \* 10*
* vb. willekeurig getal in opgegeven interval  
  *from random import randint  
  randint(0, 100)*

# I/O:

## Input():

Vraagt gebruiker om tekst in te geven. Geeft altijd string weer. Wanneer berekening wil doen met input kan gebruik maken van type casting.

## Print():

Als geen parameters meegeeft, geeft een lege regel weer. Kan twee speciale parameters meekrijgen:

* sep: geeft aan wat er moet getoond worden tussen twee parameters (default is spatie)  
  vb.  
  *print(“x”, “x”, “x”, sep = “c”)*
* end: geeft aan wat print moet tonen nadat alle parameters getoond zijn (default \n)  
  vb.  
  *print(“Y”, end = “ “)*

## format():

Staat toe om een string te bouwen waarin bepaalde waardes specifiek geformatteerd zijn.

* vb.  
  *print(“{:.3f}”.format(7 / 11)) # afronden op drie cijfers na comma*
* vb.  
  *print(“de eerste drie getallen zijn {}, {} en {}.”.format(“een”, “twee”, “drie”))*

Als je in een andere volgorde wil zet je getal tussen de accolades, begint bij 0, 1,…

String formuleren met een bepaalde precisie:

* vb.  
  *print(“de eerste drie getallen zijn {:7}, {:7} en {:7}.”.format(“een”, “twee”, “drie”))*bovenstaande code gebruikt een precisie van 7 tekens. Als de precisie te kort is voor de lengte van een string zal format gewoon meer ruimte nemen voor de string. Kan precisie dus niet gebruiken om een string voortijdig af te breken!

Als je precisie gebruikt kan je parameter links of rechts aanlijnen of centreren. Kan door een allignment teken tussen dubbele punt en precisie te plaatsen.  
< = links aanlijnen

> = rechts aanlijnen  
^ = centreren

* vb.  
  *print(“de eerste drie getallen zijn {:<7}, {:^7} en {:>7}.”.format(“een”, “twee”, “drie”))*

Kan ook formatteringsinstructies gebruiken op getallen. Als je getal als int wil laten interpreteren plaats letter ‘d’ rechts van het dubbele punt. Als je getal als float wil laten interpreteren plaats letter ‘f’.

* vb.  
  *print(“de eerste drie getallen zijn {:d}, {:d} en {:f}.”.format(1, 2, 1 / 2))*

Kan aangeven hoeveel decimalen een float moet tonen door een punt te plaatsen links van de letter ‘f’.

* vb.  
  *print(“de eerste drie getallen zijn {:.2f}, {:.2f} en {:.2f}.”.format(1.4353, 2.2324, 3.55666))*

Combinatie van precisie, aanlijning en decimalen laat toe overzichtelijke tabel te maken.

* vb.  
  *s = “{:.5d} keer {:>5.2f} is {:>5.2f}”  
  print(s.format(1, 3.75, 1 \* 3.75))  
  print(s.format(2,3.75, 1 \* 3.75))*

# Zelf functies schrijven:

## Functies definiëren:

Kan zelf functies definiëren door gebruik te maken van keyword ‘def’. Alles ingesprongen onder def statement behoort tot functie.

* vb.  
  *def my\_function():  
   print(“Hello”)  
  my\_function()*

Wanneer functie output moet retourneren, gebruik maken van keyword ‘return’.

* vb.  
  *def my\_function():  
   z = x + y  
   print(f“{x} + {y} = {z}”)  
   return z  
  my\_function()*

## Functies als variabele:

Kan een functie toewijzen aan een variabele.

* vb.  
  *fnc = my\_function():  
  fnc()*

Kan functie doorgeven als argument aan andere functie.

* vb.  
  *def minmax(x, y, z, func):  
   return func(x, y, z)  
    
  print(minmax(1, 2, 3, min))  
  print(minmax(1, 2, 3, max))*

## Anonieme functies:

Kan anonieme functies definiëren met ‘lambda’.

* vb.  
  *som = lambda x, y : x + y  
  print(som(1, 2))*

Kan kleine, zelf gecreëerde functies doorgeven aan andere functies via keyword ‘lambda’.

* vb.  
  *print(minmax(x = 1, y = 2, z = 3,  
   func=lambda x, y, x: (x + y + z)))*

## Oefeningen functies:

### Prijs boeken berekenen:

def bereken\_prijs\_aantal\_boeken(aantal\_boeken, boek\_prijs, korting\_percentage, verzendkosten\_eerste, verzendkosten\_volgende):  
 # Stap 1: Bereken de prijs van de boeken met korting  
 prijs\_met\_korting = aantal\_boeken \* boek\_prijs \* (1 - korting\_percentage / 100)  
  
 # Stap 2: Bereken de totale verzendkosten  
 if aantal\_boeken <= 1:  
 verzendkosten = verzendkosten\_eerste  
 else:  
 verzendkosten = verzendkosten\_eerste + (aantal\_boeken - 1) \* verzendkosten\_volgende  
  
 # Totale prijs is de prijs met korting plus de verzendkosten  
 totale\_prijs = prijs\_met\_korting + verzendkosten  
 return totale\_prijs  
  
# Gegeven gegevens  
boek\_prijs = 24.95  
korting\_percentage = 40  
verzendkosten\_eerste = 3  
verzendkosten\_volgende = 0.75  
aantal\_boeken = 60  
  
# Bereken de prijs voor 60 boeken  
totale\_prijs\_60\_boeken = bereken\_prijs\_aantal\_boeken(aantal\_boeken, boek\_prijs, korting\_percentage, verzendkosten\_eerste, verzendkosten\_volgende)  
print("De winkel betaalt €{:.2f} voor 60 boeken.".format(totale\_prijs\_60\_boeken))

### Celsius naar Fahrenheit/Kelvin:

def Celsius2Fahrenheit(C):  
 return C \* 1.8 + 32  
  
  
def Celsius2Kelvin(C):  
 return C + 273.15  
  
  
C1 = 33  
C2 = 3  
C3 = 46  
C4 = 22  
C5 = 15  
  
print("Celsius | Fahrenheit | Kelvin")  
print("-" \* (7 + 3 + 10 + 3 + 6))  
  
s = "{:<7.2f} | {:^10.2f} | {:>6.2f}"  
print(s.format(C1, Celsius2Fahrenheit(C1), Celsius2Kelvin(C1)))  
print(s.format(C2, Celsius2Fahrenheit(C2), Celsius2Kelvin(C2)))  
print(s.format(C3, Celsius2Fahrenheit(C3), Celsius2Kelvin(C3)))  
print(s.format(C4, Celsius2Fahrenheit(C4), Celsius2Kelvin(C4)))  
print(s.format(C5, Celsius2Fahrenheit(C5), Celsius2Kelvin(C5)))

### Anonieme functie om kwadraat en vierkantswortel van input te bepalen + controle:

square = lambda x: pow(x, 2) # aanmaken anonieme functie \*\*2  
sqrt = lambda x: pow(x, 0.5) # aanmaken anonieme functie vkw  
  
x1 = float(input("Geef een getal: \n")) # user input vragen  
  
s = "Het kwadraad van {:.4f} is {:.4f}, de vierkantswortel is {:.4f}" # formaat bepalen  
  
print(s.format(x1, square(x1), sqrt(x1))) # print resultaat  
print()  
  
import math # controle  
print("Check:\n")  
print(f"kwadraat = {math.pow(x1,2):.4f}")  
print(f"vierkantswortel = {math.sqrt(x1):.4f}")

# Condities:

## Vergelijkingsoperatoren:

* ﻿﻿Geliik aan: a == b
* ﻿﻿Niet gelijk aan: a != b
* ﻿﻿Kleiner dan: a < b
* ﻿﻿Kleiner dan of gelijk aan: a <= b
* ﻿﻿Groter dan: a > b
* Groter dan of gelijk aan: a >= b

## Volgorde van vergelijkingsoperatoren:

* ﻿﻿eerst de operatoren in, not in, is, is not en de vergelijkingsoperatoren ==, !=, <, <=, > en >=;
* ﻿﻿dan de logische operator not
* ﻿﻿dan de logische operator and
* ﻿﻿tenslotte de logische operator or

## Oefeningen vergelijkingsoperatoren:

### Programma dat uitrekent hoeveel inkom je moet betalen:

leeftijd = int(input("Wat is je leeftijd? \n"))  
  
if leeftijd <= 3:  
 print("Gratis inkom")  
elif leeftijd >= 4 and leeftijd <= 12 or leeftijd >=64:  
 print("Inkom = 20 euro")  
elif leeftijd >= 13 and leeftijd <= 18:  
 print("Inkomt = 25 euro")  
else:  
 print("Inkom = 28 euro")

### Pincode controleren:

pin = 1234  
  
userinput = int(input("Voer je pincode in: \n"))  
  
if pin == userinput:  
 print("Pin correct!")  
else:  
 print("Pin geweigerd!")

## Selecteren van karakters en substrings:

### Karakter in string wijzigen, hoofdletter W naar kleine w:

boodschap = "Hello World!"  
print(boodschap[:6] + 'w' + boodschap[7:])

Kan ook met .replace():

boodschap = "Hello World!"  
print(boodschap)  
boodschap = boodschap.replace('W', 'w')  
print(boodschap)

### Controleren of een substring voorkomt in string:

boodschap = "Hello World!"  
print('!' in boodschap)

Output= true.

boodschap = "Hello World!"  
print("Hello" not in boodschap)

Output=false.

*.startswith()/.endswith()* om te controleren of string begint/eindigt met bepaalde substring.

.index() geeft index van substring in string terug.

.count() geeft weer hoe vaak een sub string in een string voorkomt.

.split() staat toe om op basis van een separator een string te splitsen in meerdere strings.

## Oefeningen:

### Functie om spaties te vervangen met tabs:

def uitrekken(s):  
 s = s.replace(" ", "\t\t\t")  
 return s  
  
strng = str(input("Geef de gewenste string in: "))  
print("Jij gaf onderstaande string in: \n" + strng)  
  
print("Uitgerokken versie: \n" + uitrekken(strng))

### Functie om spaties te vervangen met teken gekozen door gebruiker:

def separatorveranderen(s, se):  
 s = s.replace(" ", se)  
 return s  
  
strng = str(input("Geef de gewenste string in: "))  
print("Jij gaf onderstaande string in: \n" + strng)  
  
sep = input("Welke separator wil je gebruiken om de spatie te vervangen?")  
print("Jij gaf onderstaande separator in: \n" + sep)  
  
print("Aangepaste versie: \n" + separatorveranderen(strng, sep))

### Functie die nagaat of er backslash of slash voorkomt in folder naam en toevoegt op einde van de string indien backslash of slash niet reeds aanwezig:

def check\_folder\_name(folder\_naam):  
 if not (folder\_naam[-1] == '/' or folder\_naam[-1] == "\\"):  
 if "/" in folder\_naam and "\\" not in folder\_naam:  
 folder\_naam += "/"  
 elif "\\" in folder\_naam and "/" not in folder\_naam:  
 folder\_naam += "\\"  
 else:  
 folder\_naam += "/"  
 return folder\_naam  
  
dir1 = "C:\\temp\\"  
dir2 = r"C:\temp"  
dir3 = "C:/local"  
dir4 = "C:/temp"  
dir5 = "temp"  
dir6 = r"C:\temp/opl"  
  
print(check\_folder\_name(dir1))  
print(check\_folder\_name(dir2))  
print(check\_folder\_name(dir3))  
print(check\_folder\_name(dir4))  
print(check\_folder\_name(dir5))  
print(check\_folder\_name(dir6))

# Lists:

## Elementen opvragen:

In:

*myNewList[plane, bike, car]*

*print(myNewList[0:4])*

Out:

Plane, bike, car

Kan ook in omgekeerde volgorde, of een enkel element opvragen.

## Elementen toevoegen en verwijderen:

Element toevoegen aan het einde van de lijst:

*myNewList.append(“boat”)*

Element invoegen op specifieke index:

*myNewList.insert(2, “trike”)*

Element verwijderen:

*myNewList.remove(“boat”)*

Element op specifieke index verwijderen: (als geen index geeft laatste element verwijderd)

*myNewList.pop(2)*

Alternatief kan ook del operator gebruiken:

*Del myNewlist[2]*

Hele lijst verwijderen ook mogelijk met del:

*Del myNewList*

Lijst aanmaken met x\*aantal gelijke elementen:  
*lijst = 69 \* [“nice”]*

## Kopiëren:

Hele lijst kopiëren:

*CopyOfmyNewList = myNewList.copy()*

Alternatief is lijst meegeven aan constructor voor maken van nieuwe lijst:

*CopyOfmyNewList = list(myNewList)*

## Samenvoegen:

Kan met + operator twee lijsten samenvoegen:

*all0fTheData = firstPart0fTheData + secondPart0fTheData*

## Nuttige methodes:

Nagaan hoeveel keer een bepaalde waarde in een list voorkomt:

*Print(myNewList.count(“plane”))*

Index van een waarde opvragen:

*myNewList.index(“boat”)*

Lijst sorteren:

*myNewList.sort() #wordt alfabetisch gesorteerd*

## Oefeningen:

### Lijst aanmaken, controleren of bepaald element aanwezig is, reeks elementen toevoegen aan lijst, tellen hoe vaak bepaald element voorkomt in nieuwe lijst:

dranken = ["water", "cola", "bier", "monster", "fristi"]  
print("Lijst van dranken: \n", dranken, "\n")  
print(("Bevat de lijst het element 'water'? "), "water" in dranken, "\n")  
  
nieuwedranken = ["chocomelk", "thee", "sojamelk", "fruitsap"]  
  
dranken += nieuwedranken  
print("Bijgewerkte lijst van dranken: \n", dranken)  
  
print("Element 'fruitsap' komt " ,dranken.count("fruitsap"), " keer voor.")

# Lussen:

## For-lus:

Alle karakters in een string overlopen en aanpassen:

tekst = "Python"  
for ch in tekst: # voor elk karakter in de string tekst geldt onderstaande  
 print(ch, end = " ") # vervangt het einde van elk karakter met spatie

Elk item in een lijst afdrukken:

Lijst = ["car","Bike","plane"]  
for item in Lijst:  
 print(item)

Elk item in lijst met hoofdletter laten weergeven:

Lijst = ["jan", "Dirk", "wimmezwam"]  
for item in Lijst:  
 print(item.capitalize())

Vast aantal iteraties uitvoeren:

for i in range(10): #range is 0-10  
 print(i\*2) #elk item in range \*10

Range kent begin- eind- en startwaarde: start(3), stop(7), step(2):

idk = range(10)  
for i in range(3,7,2):   
 print(i)  
# 3 is de waarde waar de lus start  
# 7 is de waarde waarbij de lus stopt (tot aan deze waarde, niet tot en met)  
# 2 is het increment waarmee de lus telkens herhaalt

## While-lus:

Zolang i kleiner is dan 5 worden iteraties uitgevoerd:

i = 0;  
while i < 5:  
 i += 1  
 print("i has value: " + str(i))

Gebruiker moet cijfer geven tussen 1 en 20. Zolang dat niet gebeurt zal er input gevraagd worden binnen dat bereik. Als de gebruiker cijfer ingeeft groter dan 0 en kleiner dan 20 voert lus even veel iteraties uit als door gebruiker ingegeven getal:

n = 0  
while n == 0 or n > 20:  
 n = int(input("Geef een cijfer tussen 1 en 20:"))  
  
count = 0  
while n > count:  
 count += 1  
 print("Iteration " + str(count) + " of " + str(n))

### Iteraties onderbreken en overslaan:

For-lus onderbreken aan de hand van break statement:

for i in range(10):  
 if i == 6:  
 break  
 print(i)  
else:  
 print("For-loop finished")

Bepaalde waarde in while-lus overslaan:

i = 0;  
while i < 5:  
 i += 1 # zolang i kleiner is dan 5 wordt de waarde van i bij elke iteratie verhoogd met 1  
 if i == 2: # als i gelijk is aan 2 wordt de huidige iteratie overgeslaan  
 continue  
 print("i has the value: " + str(i))  
else:  
 print("The loop is finished")

While-lus onderbreken:

i = 0;  
while i < 5: # zolang de waarde van i onder 5 blijft zal de lus blijven itereren  
 i += 1  
 print("i has the value of: " + str(i))  
 if i == 3: # als de waarde van i gelijk is aan 3 wordt de lus afgebroken  
 break  
else:  
 print("The loop has ended.")

## Oefeningen:

### Programma dat het kwadraat van de getallen 1 t.e.m. 20 afdrukt:

square = lambda x: pow(x,2) # anonieme functie voor machtsverheffing  
  
i = 0;  
for i in range(21): # bereik bepalen  
 if i == 0: # kwadraat van 0 overslaan  
 continue  
 print("Het kwadraat van " + str(i) + " is " + str(square(i)))

### Programma dat gebruiker vraagt 5 soorten fruit op te geven en deze in lijst plaatst. Controle of er geen dubbele waarden zitten, lijst wordt op 1 lijn weergegeven in omgekeerde alfabetische volgorde. Vervolgens wordt lijst leeggemaakt:

listFruits = [] # lijst initialiseren  
n = 1;  
while n < 6: # lus die gebruiker vraagt om precies 5 fruitsoorten in te geven en controleert of er geen dubbele waarden zijn  
 fruit = input("Geef een fruitsoort in: ").upper()  
 n += 1  
 if fruit not in listFruits:  
 listFruits.append(fruit) # fruit toevoegen aan de lijst  
 else:  
 print("Dit fruit zit reeds in de lijst, geef een andere fruitsoort in")  
 n -= 1  
  
listFruits.sort(reverse=True) #lijst omgekeerd alfabetische volgorde plaatsen  
  
for fruit in listFruits: # alle items in lijst afdrukken op 1 lijn  
 print(fruit, end = "\t")  
  
print()  
  
listFruits.clear() # lijst leegmaken zonder de lijst te verwijderen  
print(listFruits)

### Programma om het verschil tussen twee lijsten te visualiseren. Print de elementen van de eerste lijst uit die niet in de twee lijst zitten en omgekeerd:

lijst1 = ["Appel", "Peer", "Mango", "Banaan"]  
lijst2 = ["Appel", "Kiwi", "Lichee", "Passievrucht"]  
  
print("Volgende items uit de eerste lijst komen niet voor in de tweede lijst:")  
i = 0;  
for i in lijst1:  
 if i not in lijst2:  
 print(i, end = ", ")  
print()  
print()  
print("Volgende items uit de tweede lijst komen niet voor in de eerste lijst:")  
i = 0  
for i in lijst2:  
 if i not in lijst1:  
 print(i, end = ", ")

### Programma voor blad steen schaar tegen pc:

from random import randint  
opties = ["blad", "steen", "schaar"]  
  
gebruiker= int(input("0(blad), 1(steen) of 2(schaar) \n"))  
pc = randint(0,2)  
  
print("\nJij koos '{}'\nDe computer koos: '{}'\n".format(opties[gebruiker],opties[pc]))  
  
if gebruiker == pc:  
 print("Gelijkstand")  
elif (pc == 0) and (gebruiker == 2) or \  
 (pc == 1) and (gebruiker == 0) or \  
 (pc == 2) and (gebruiker == 1):  
 print("Gewonnen!")  
else:  
 print("Verloren!")

### Oefening tabel + studenten onderverdelen in groepen:

# data  
header = ["Voornaam", "Familienaam", "U-nummer", "Richting"]  
studenten = [  
 ["Guido", "van Rossum", "u0465", "APP"],  
 ["Steve", "Jobs", "u0145", "APP"],  
 ["John", "von Neumann", "u0421", "NET"],  
 ["Bill", "Gates", "u0260", "APP"],  
 ["Linus", "Torvalds", "u0912", "APP"],  
 ["Sundar", "Pichai", "u0123", "AI"],  
 ["Larry", "Page", "u0456", "AI"],  
 ["Robin", "Li", "u0613", "BIT"],  
 ["Grace", "Hopper", "u0435", "APP"],  
 ["Jeffrey", "Bezos", "u0885", "BIT"],  
 ["Alan", "Turing", "u0102", "AI"],  
 ["Mark", "Zuckerberg", "u0542", "NET"],  
 ["Charles", "Robins", "u0648", "NET"],  
 ["Adele", "Goldberg", "u0893", "APP"],  
 ["Lawrence", "Ellison", "u0247", "BIT"],  
 ["Ada", "Lovelace", "u0111", "AI"],  
 ["George", "Boole", "u0234", "BIT"],  
]  
  
# function to print table  
def print\_table(table, header=None, column\_width=15):  
 # print header if given  
 if header is not None:  
 print\_row(header, column\_width)  
 # print horizontal line below header  
 ncol = len(header)  
 print("-" + "-" \* (column\_width + 1) \* ncol)  
 # loop through table rows  
 for row in table:  
 print\_row(row, column\_width)  
  
# function to print table row  
def print\_row(row, column\_width=15):  
 # format function  
 fmt = lambda column: "{:^{}}|".format(column, column\_width)  
 # initialize row string  
 txt = "|"  
 # loop through columns in row  
 for column in row:  
 txt += fmt(column)  
 print(txt)  
  
print(print\_table(studenten, header))  
  
# function to check if u-number is even  
def is\_even(u\_num):  
 # select number and convert to int  
 num = int(u\_num[1:])  
 # number is even if % 2 is zero  
 return(num % 2) == 0

# function to determine

student's group  
def get\_group(course):  
 if course in ["AI", "BIT"]:  
 group = "AI\_BIT"  
 elif course in ["NET", "APP"]:  
 group = "NET\_APP"  
 else:  
 group = "unknown"  
 return group

Nog af te werken.

# Tuples:

Is onveranderlijk, kan geen elementen toevoegen/verwijderen/wijzigen na creatie.

Kan tuple unpacken:

abc = 'a', 'b', 'c' # variabele abc bevat tuple met 3 elementen  
print(abc)  
  
a, b, c = abc # ken de 3 elementen van abc toe aan 3 verschillende variabelen (= unpacken naar var)  
print(a, b, c)

Functie kan slechts één waarde retourneren, met tuple mogelijk om meerdere waarden te retourneren:

def stats(values):  
 m = min(values)  
 mx = max(values)  
 mn = sum(values) / len(values)  
 return m, mx, mn  
  
out = stats([1, 3, 5, 10]) # list als input  
print(out)  
print(type(out))  
print()  
  
out = stats((10, 30, 50, 100)) # tuple als input  
print(out)  
print(type(out))  
  
m, mx, mn = stats((1, 2, 3)) # output toekennen aan 3 var  
print(m, mx, mn)

Duck typing: functie stats kan als inputparameter een list of tuple van waarden krijgen omdat de functies min(), max() en sum() voor de beide datatypes gedefinieerd zijn.

Kan net als bij list individuele elementen benaderen via indices:

fruit = ("appel, ", "banaan", "Kers")  
print(fruit[2]) # print derde element af

Door elementen in tuple gaan via for/while lus:

fruit = ("appel ", "banaan ", "Kers")  
  
for i in range(len(fruit)): # for lus om door elementen van tuple te gaan  
 print(fruit[i])  
  
print ()  
  
i = 0  
while i < len(fruit): # while lus om door elementen van tuple te gaan  
 print(fruit[i])  
 i += 1

Kan net als bij list min(), max() en sum() functie gebruiken bij tuple die uit cijfers bestaat:

t1 = (327, 419, 101, 667, 925, 225)  
print (max(t1))  
print (min(t1))  
print(sum(t1))

Tuples hebben ook count() en index() methode:

t = (1, 2, 3, 4, 3, 2, 1)  
print (t.count (3)) # geeft aan hoeveel keer het element 3 in de tuple voorkomt  
print(t.index(2)) # geeft de (eerste) index weer van het element 2

Tuples samenvoegen kan via ‘+’ operator:

a = (1, 2, 3)  
b = (4, 5)  
print(a + b)

Kan tuple kopiëren door te concateneren met lege tuple(niet via copy() methode als bij list:

a1 = (1, 2, 3)  
print("a1:", a1)  
  
a2 = a1 + () # creëert nieuwe tuple  
print("a2:", a2)

Tuple omzetten naar list:

list\_fruit = ["mango", "banaan", "appel", "kers"]  
tuple\_fruit = ("mango", "banaan", "appel", "kers")  
  
list\_fruit = list(tuple\_fruit)  
print(list\_fruit)  
tuple\_fruit\_2 = tuple(list\_fruit)  
print(tuple\_fruit\_2)

# Sets:

Zijn ongeordend, heterogeen (niet enkel elementen van zelfde datatype) en veranderlijk (kan elementen toevoegen of verwijderen, maar elementen zelf zijn onveranderlijk). Bevatten enkel unieke elementen (geen dubbele waarden).

Set aanmaken door elementen tussen accolade te plaatsen of door set() functie aan te roepen:

fruitset = {"appel", "banaan", "kers"}  
print (fruitset)  
s = set([1, 2, 3])  
print(s)

Set maken uit verschillende letters van een string:

helloset = set("hello world")  
print (helloset)

Lege set aanmaken:

lege\_set = set() # niet {}!  
print(lege\_set)  
print(type(lege\_set))

Omdat elementen in tuple onveranderlijk (kan niet toevoegen, verwijderen of veranderen) zijn kan je een tuple opnemen in een set, met een list of andere set gaat dit niet.

Kan len() functie ook bij set gebruiken:

fruitset = {"appel", "banaan", "kers"}  
print(len(fruitset))

Kan met de in-operator en for-lus alle elementen in een set doorlopen. Kan echter de volgorde niet bepalen.

fruitset = {"appel", "banaan", "kers"}  
for fruit in fruitset:  
 print(fruit)

Met in-operator kan bepalen of element tot een set behoort:

fruitset = {"appel", "banaan", "kers"}  
if "peer" in fruitset:  
 print("peer")  
else:  
 print("geen peer")

met add() en remove() methoden kan je elementen toevoegen of verwijderen:  
kan ook met pop() als je laatst toegevoegde element wil verwijderen en retourneren.

fruitset = {"appel", "banaan", "kers"}  
fruitset.add("peer")  
fruitset.remove("appel")  
print(fruitset)  
fruit = fruitset.pop()  
print(fruitset)  
print(fruit)

## Set bewerkingen:

Met union() methode krijg je unie van twee sets:

set1 = {"a", "b", "c"}  
set2 = {1, 2, 3}  
  
set3 = set1.union(set2)  
print(set3)

Kan ook update() gebruiken, maar dan worden elementen toegevoegd aan de set die de methode aanroept:

set1 = {"a", "b", "c"}  
set2 = {1, 2, 3}  
  
set1.update(set2)  
print(set1)

Om verschil tussen twee of meer sets te visualiseren kan difference() gebruiken:

computers1 = {"HP", "Dell", "Asus", "Apple"}  
computers2 = {"Lenovo", "HP", "Apple", "Acer"}  
verschil = computers1.difference(computers2)  
print(verschil)

Controleren of twee sets disjoint zijn (geen gemeenschappelijke elementen hebben):

computers1 = {"HP", "Dell", "Asus", "Apple"}  
computers2 = {"Lenovo", "HP", "Apple", "Acer"}  
print(computers1.isdisjoint(computers2)) # false apple komt in beide lijsten voor  
print({1,2,3}.isdisjoint({4,5,6})) # true

Controleren of ene set subset of superset is van andere set:

A = {1, 2, 3}  
B = {1, 2, 3, 4, 5}  
print (A.issubset (B)) # True want A is deelverzameling van B  
print(B.issuperset (A)) # True om dezelfde reden

Kan sets ook kopiëren met copy() methode:

Kan een set ‘listcasten’ om de set om te zetten naar een list, daarna mogelijk om set te sorteren (weet zeker welke volgorde in een lus doorlopen wordt):

A = {"appel", "kiwi", "peer"}  
for element in A:  
 print(element)  
  
print()  
  
fruitlist = list(A) # set omzetten naar list  
fruitlist.sort() # lijst sorteren  
for element in fruitlist:  
 print(element)

## Frozenset:

Kan set creëren waaraan geen elementen kunnen toegevoegd worden of verwijderd kunnen worden. Alle bovenstaande methodes werken ook op frozenset (behalve toev en verw):

fruit1 = frozenset(["appel", "kers", "peer"])  
print(fruit1)

# Dictionaries:

Een manier om ongeordende heterogene data te structureren. Om een waarde te vinden moet de key van het element kennen.

Dictionary = verzameling van keys met geassocieerde waarden. De keys moeten onveranderlijke datatypen zijn.

Kan elementen wijzigen, toevoegen en verwijderen.

Dictionary aanmaken:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
print(fruitmand)

Kan ook dict() constructor gebruiken (= in plaats van :):

auto = dict(merk = "ford", model = "mustang", jaar = 1964)  
print(auto)

Lege dictionary aanmaken:

lege\_dict = {}  
print(lege\_dict)

Kan ook andere dictionaries onderbrengen in dictionary:

auto = dict(merk = "ford", model = "mustang", jaar = 1964)  
print(auto)  
  
persoon = {"naam": "jos", "leeftijd": 58, "auto": auto}  
print(persoon)

## Elementen selecteren en vervangen:

Waarde vinden die bij een specifieke sleutel hoort:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
print(fruitmand["appel"])

Kan ook get() methode gebruiken:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
aantal\_kersen = fruitmand.get("kers")  
print(f"Er zijn {aantal\_kersen} kersen in de fruitmand.")

Element van een dictionary in een dictionary opvragen:

auto = dict(merk = "ford", model = "mustang", jaar = 1964)  
persoon = {"naam": "jos", "leeftijd": 58, "auto": auto}  
print("{} rijdt met een {}".format(persoon["naam"],  
 persoon["auto"]["merk"]))

Wanneer je element wil opvragen dat niet voorkomt in de dictionary krijg je error.

Element wijzigen:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
print(fruitmand)  
fruitmand["banaan"] = 3 # 3 bananen ipv 5  
fruitmand["appel"] +=1 # 1 appel toevoegen  
print(fruitmand)

Bij dictionary kan in-operator gebruiken om te checken of gegeven key in dictionary voorkomt:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
print("In de fruitmand ligt: ")  
for fruit in fruitmand:  
 print(fruit)

Met aantallen:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
print("In de fruitmand ligt: ")  
for fruit in fruitmand:  
 aantal = fruitmand[fruit]  
 print(aantal, fruit)

Kan efficiënter met methode items():

fruitmand = {"appel":3, "banaan":5, "kers":50}  
print("In de fruitmand ligt: ")  
for fruit, aantal in fruitmand.items():  
 print(aantal, fruit)

Keys en values opvragen:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
fruit = fruitmand.keys()  
aantal = fruitmand.values()  
print(fruit)  
print(aantal)

Waarden zonder sleutels doorlopen:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
for fruit in fruitmand.values():  
 print(fruit, end=" ")

## Elementen toevoegen en verwijderen:

Toevoegen:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
fruitmand["mango"] = 1  
print(fruitmand)

Via pop() methode verwijder je element die bij opgegeven key hoort:

os = dict(naam="Windows", versie=10, opensource=False)  
print(os)  
os.pop("versie")  
print(os)

Kan ook via del-operator:

os = dict(naam="Windows", versie=10, opensource=False)  
print(os)  
del os["versie"]  
print(os)

Via methode clear() kan je volledige dictionary leegmaken:

os = dict(naam="Windows", versie=10, opensource=False)  
print(os)  
os.clear()  
print(os)

## Dictionaries samenvoegen:

Met update() methode kan andere dictionary toevoegen:

fruitmand = {"appel":3, "banaan":5, "kers":50}  
fruitmand.update({"peer" : 5, "kiwi" : 3})  
print(fruitmand)

Samenvoegen naar nieuwe dictionary:

d1 = {"naam" : "Jos", "leeftijd" : 56}  
d2 = {"functie" : "systeembeheerder"}  
d3 = dict (list(d1.items ()) + list(d2.items ()))  
print(d3)

## Dictionaries kopiëren:

Kan dictionaries kopiëren met copy() methode. Zijn gelijk (==) als ze dezelfde keys en values hebben. Dictionaries zijn niet geordend dus volgorde waarin values/keys gedefinieerd zijn maakt niet uit.

# Comprehensions:

Python geeft mogelijkheid om op efficiëntere, kortere manier dan met lussen reeksen zoals lists, sets en dictionaries te maken.

## List comprehensions:

Syntax:

new\_list = [expression for element in iterable if (element satisfies condition)]

Lijst met elementen van 1 tot en met 10:

lst = [i for i in range(1, 11)]  
print(lst)

Bewerking uitvoeren op elementen:

print([i\*\*2 for i in range(1, 10)])

Geneste list comprehension:

print([[j for j in range(0,5)] for i in range(0,5)])

## Dictionary comprehensions:

Syntax:

new\_dict = {key: expression for key, value in iterable if (key, value satisfies condition)}

## Set comprehensions:

Syntax:

new\_set = {expression for element in iterable if (element satisfies condition)}

# Herhalingsoefeningen:

Schrijf een functie die alle karakters van een string die gelijk zijn aan het eerste karakter van die string vervangen door een opgegeven karakter, behalve het eerste karakter zelf. De defaultwaarde van het opgegeven karakter is '$':

def vervang\_karakters(tekst, vervang ='$'):  
 return tekst[0] + tekst[1:].replace(tekst[0], vervang)  
 # geef 0e karakter weer + begin vanaf karakter 1e karakter met vervangen  
 # (vervang tegen gelijk aan teken op pos 0 met variabele vervang)  
  
print(vervang\_karakters("the quick brown fox jumps easily over the fat and lazy dog", ))  
print(vervang\_karakters("the quick brown fox jumps easily over the fat and lazy dog", "%"))

Schrijf een anonieme functie die een string omdraait als de lengte ervan een meervoud van 4 is. Test je functie uit op enkele zelf bedachte strings:

omdraaien = lambda tekst: tekst[::-1] if len(tekst) % 4 == 0 else tekst  
print(omdraaien("jan"))  
print(omdraaien("Dirk"))

Schrijf een programma dat een overzichtelijke tabel produceert met daarin de temperatuur van 15 tot 45 graden Celsius (in stappen van 1 graad) in de eerste kolom, en de overeenkomstige waarden in graden Fahrenheit en Kelvin in de volgende twee kolommen:

def Celsius2Fahrenheit(C): # functie die celsius omzet naar Fahrenheit  
 return C \* 1.8 + 32  
  
  
def Celsius2Kelvin(C): # functie die celsius omzet naar kelvin  
 return C + 273.15  
  
  
header = "Celsius | Fahrenheit | Kelvin\n" # header en onderlijning  
line = len(header) \* "-"  
print(header + line)  
  
s = "{:<7.2f} | {:^10.2f} | {:>6.2f}" # formaat: uitlijning, precisie, cijfers na comma en type bepalen  
Celsius = range(46) # celsius is een lijst van 1-45  
for i in Celsius: # voor elke waarde in de lijst met graden celsius  
 print(s.format(i, Celsius2Fahrenheit(i), Celsius2Kelvin(i)))   
# lijst afdrukken met omzettingen volgens formaat s

Schrijf een programma waarbij twee strings ingegeven worden. Vervolgens worden deze twee strings tot een string samengevoegd met een spatie tussenin. Bovendien worden de eerste 2 karakters van elke string verwisseld met elkaar:

def samenvoegen(s1, s2):  
 samenvoegsel = (s2[0:2] + s1[2:]) + " " + (s1[0:2] + s2[2:])  
 return samenvoegsel  
  
  
print(samenvoegen("Dirk", "Van-Sint-Jan"))

Schrijf een programma dat enkele statistieken van een reeks opgegeven getallen weergeeft: het gemiddelde, het minimum en het maximum. De gebruiker geeft deze getallen een voor een op. De berekening wordt pas uitgevoerd nadat alle getallen zijn ingevoerd. Het einde van de invoer wordt aangegeven met het woord “einde”. Het programma print vervolgens de berekende statistieken uit:

def stats(getallen):  
 m = {}  
 m["gemiddelde"] = sum(getallen) / len(getallen)  
 m["minimum"] = min(getallen)  
 m["maximum"] = max(getallen)  
 return m  
  
def vraag\_getal(i):  
 getal = input(f"Geef getal {i}: ")  
 if getal.lower() == "einde":  
 return None  
 else:  
 return float(getal)  
  
def print\_stats(getallen):  
 print()  
 if len(getallen) > 0:  
 m = stats(getallen)  
 for stat, value in m.items():  
 print(f"{stat}: {value:.3f}")  
 else:  
 print("Geen getallen opgegeven.")  
  
def vraag\_getallen():  
 getallen = []  
 i = 0  
 while True:  
 i += 1  
 getal = vraag\_getal(i)  
 if getal is None:  
 break  
 else:  
 getallen.append(getal)  
 print\_stats(getallen)  
  
vraag\_getallen()

Schrijf een anonieme functie die de eerste en laatste karakter omwisselt van een gegeven string:

tekst = "The quick brown fox jumps easily over the fat and lazy dog."  
  
omwisselen = lambda tekst: tekst[-1] + tekst[1:-1] + tekst[0] #laatste substring + string vanaf 2e karakter tot laatste + eerste substring  
  
print(omwisselen("hond"))  
print(omwisselen(tekst))  
print(omwisselen("j")) # werkt maar geeft gewoon twee keer zelfde string door

Schrijf een programma dat de Lotto trekking simuleert. In de lottotrekking worden 6 getallen tussen 1 en 42 getrokken (1 en 42 inclusief), en een bijkomend nummer. De getallen worden niet teruggelegd, dus je krijgt 7 unieke getallen. Print de uitslag uit: eerst de 6 getallen na elkaar gescheiden door een komma, daarna op de volgende lijn het bijkomend nummer:

from random import randint  
  
  
def trek\_getal(i):  
 getal = randint(1, 43)  
 return int(getal)  
  
  
def lotto():  
 trekking = []  
 i = 0  
 while i < 7:  
 i += 1  
 getal = trek\_getal(i)  
 if getal not in trekking:  
 trekking.append(getal)  
 print("De winnende cijfers zijn: " + str(trekking[0:6]) + "\n" + "Het bijkomende cijfer is: " + str(trekking[-1]))  
  
  
lotto()

Schrijf een functie die de karakters ‘ing’ toevoegt aan het einde van een ingegeven string. Indien de laatste 3 karakters van de string al eindigen op ‘ing’, worden de karakters ‘ly’ toegevoegd. Test je functie uit op de volgende lijst van Engelse werkwoorden, en gebruik hiervoor een lus:

def werk\_woord(text):  
 return text + "ly" if text.endswith("ing") else text + "ing"  
  
for text in ["jump", "play", "minecraft", "dirk", "Janning"]:  
 print(werk\_woord(text))

Schrijf een Python functie die de leeftijd van een persoon checkt. De functie geeft aan of de gebruiker oud genoeg is om alcohol te bestellen. De functie checkt ook of de persoon die 18 of ouder is nog moet rijden:

# Dit is een functie die wordt gebruikt om te controleren of een persoon alcohol mag kopen op basis van de opgegeven gegevens.  
# Als de persoon jonger is dan 18 en ook de "Bob" is, of oud genoeg is, maar geen "Bob" is, wordt een gepaste boodschap afgedrukt.  
def alcoholchecker(persoon):  
 if persoon['leeftijd'] < 18:  
 boodschap = "Sorry, {}, je bent nog geen 18!"  
 elif persoon['is\_bob']:  
 boodschap = "Sorry, {}, je mag dan wel 18 zijn, maar je bent Bob!"  
 else:  
 boodschap = "Ok, {}, je bent oud genoeg om alchohol te kopen en je bent geen Bob."  
 print(boodschap.format(persoon["naam"]))  
  
# Deze functie vraagt de gebruiker om informatie over een persoon in te voeren, zoals naam, leeftijd en of ze "Bob" zijn.  
# De ingevoerde gegevens worden in de lijst personen opgeslagen als een dictionary.  
def voeg\_persoon\_toe(personen):  
 naam = input("naam: ")  
 leeftijd = int(input("Leeftijd: "))  
 is\_bob = input("Ben je Bob? (j/n)").lower()[0] == "j"  
 personen.append({  
 "naam" : naam,  
 "leeftijd" : leeftijd,  
 "is\_bob" : is\_bob  
 })  
  
personen = [] # Hier wordt een lege lijst personen gemaakt om de informatie over verschillende personen op te slaan.  
for i in range(5): # Deze loop voert vijf keer de functie voeg\_persoon\_toe uit om informatie over vijf verschillende personen te verzamelen.  
 voeg\_persoon\_toe(personen)  
 print()  
  
# Deze loop doorloopt de lijst personen en roept vervolgens de alcoholchecker-functie aan  
# voor elke persoon om te bepalen of ze alcohol mogen kopen.  
for persoon in personen:  
 print()  
 alcoholchecker(persoon)

Schrijf code die een gegeven zin omzet in hoofdletters en in de zogenaamde “CamelCasing” variant. Met dit laatste wordt hier bedoeld dat elk woord in de zin met een hoofdletter begint. Het programma vraagt een zin op en twee functies zorgen voor de omzetting van die zin naar het gevraagde. De resultaten worden getoond:

zin = "The quick brown fox jumps easily over the fat and lazy dog"  
def titel(zin):  
 return zin.title()  
  
def capslock(zin):  
 return zin.upper()  
  
print(titel(zin))  
print(capslock(zin))

Schrijf een programma dat aan de gebruiker een reeks getallen vraagt en die getallen rangschikt van klein naar groot. De gerangschikte getallen worden aan de gebruiker getoond. Het opvragen van de getallen gaat door tot de gebruiker het woord “stop” opgeeft:

def vraag\_getal(i):  
 getal = input(f"Geef getal {i}: ")  
 if getal.lower() == "stop":  
 return None  
 else:  
 return float(getal)  
  
  
def vraag\_getallen():  
 getallen = []  
 i = 0  
 while True:  
 i += 1  
 getal = vraag\_getal(i)  
 if getal is None:  
 break  
 else:  
 getallen.append(getal)  
 getallen.sort()  
 print(getallen)  
  
  
vraag\_getallen()

Schrijf een functie die een string als input accepteert en waarbij de eerste 2 en laatste 2 karakters worden samengevoegd tot een nieuwe string. Indien de lengte van de string lager is dan 2, wordt een boodschap weergegeven dat de string te kort is:

def knip(tekst):  
 n = len(tekst)  
 if n < 2:  
 print("De string " + "\'" + str(tekst) + "\'" + " is te kort")  
 else:  
 print(tekst[0:2] + tekst[-2:n])  
  
knip("dirk vansintjan")  
knip("ja")  
knip(("k"))

Schrijf nu een functie is\_schrikkeljaar(). De functie moet een jaartal als invoer nemen en de waarde True teruggeven als het een schrikkeljaar is en de waarde False als het geen schrikkeljaar is. Maak dan een lijst van 10 willekeurig gegenereerde jaren tussen 1900 en 2100. De jaren moeten uniek zijn. Overloop de jaren in de lijst met een lus en check of het al dan niet schrikkeljaren zijn. Print elk jaar uit en geef aan of het een schrikkeljaar is met de boodschap: "het jaar \\ is \<een/geen> schrikkeljaar!":

from random import randint  
  
def is\_schrikkeljaar(jaar):  
 return jaar % 4 == 0 and (jaar % 100 != 0 or jaar % 400 == 0)  
  
def print\_is\_schrikkeljaar(jaar):  
 print(f”Het jaar {jaar} is “ +  
 (“” if is\_schrikkeljaar(jaar) else “g”) + “een schrikkeljaar”)  
  
jaren = set() # set omdat waarden uniek moeten zijn  
while len(jaren) < 10:  
 jaren.add(randint(1900,2100))  
  
for jaar in jaren:  
 print\_is\_schrikkeljaar(jaar)

Een palindroom (ook keerwoord of spiegelwoord genoemd) is een woord waarin de letters symmetrisch gerangschikt zijn, zodanig dat het woord van achter naar voren gelezen hetzelfde is als van voor naar achter. Schrijf een anonieme functie die True retourneert als een opgegeven string een palindroom is en False als dat niet zo is.   
Er zijn ook palindroomzinnen, waarbij de woorden van achter naar voor worden gelezen, niet de letters. Bijvoorbeeld: "Lollig zijn lui die zeggen die lui zijn lollig". Hoe ga je dit checken?

is\_palindroom = lambda woord: woord[:] == woord[::-1]  
  
print(is\_palindroom("lepel"))  
print(is\_palindroom("soep"))  
  
def is\_palindroomzin(zin):  
 zin = zin.lower().split(" ")  
 return zin == zin[::-1]  
  
print(is\_palindroomzin("Lollig zijn lui die zeggen die lui zijn lollig"))

Schrijf een functie met de naam geef\_wijsheid() die een geheel getal als invoer neemt. Als antwoord geeft je functie een string terug met een wijze spreuk erin. De spreuk wordt gekozen op basis van het opgegeven nummer en hetzelfde nummer moet altijd dezelfde spreuk teruggeven. Zorg dat je functie alleen geldige getallen accepteert (dus alleen getallen waar ook een spreuk bij hoort). Als een ander getal wordt gegeven, geeft je functie een foutmelding terug.  
Schrijf dan een tweede functie fortune\_cookie() die een willekeurige spreuk retourneert, net zoals het gelukskoekje in een Chinees restaurant:

from random import randint  
  
  
def geef\_wijsheid(cijfer):  
 spreuken = {  
 1: "leevn es vo de rappe zeitn boer en jie stook zinn trektur in 2de",  
 2: "Kwil pitten zeitn boer en ie kocht nen bak druvven",  
 3: "Kga voort doen zei den boer en ie dronkt no een pinte",  
 4: "Asse kan veugeln kanse vliegn wok zeitn boer en n smeet zn wuf deur d’rutte",  
 5: "Kit get zeitn boer, en ie gavet were"  
 }  
 if 1 <= cijfer <= len(spreuken):  
 return spreuken[cijfer]  
 else:  
 print("Er bestaat geen overeenkomende spreuk voor het cijfer dat je opgaf")  
  
  
def fortune\_cookie():  
 return geef\_wijsheid(randint(1, 5))  
  
  
for i in range(5):  
 print(fortune\_cookie())

Schrijf een functie waarbij HTML tags worden toegevoegd aan een ingegeven string. Bij de functie worden 2 paramaters meegegeven: Het soort tags en de ingevoerde string. Indien de string newlines "\n" bevat, dan moeten die door een single line break "<br>" vervangen worden:

def add\_tags(tekst, tag):  
 tekst = tekst.replace("\n","<br>")  
 return f"<{tag}>{tekst}</{tag}>"  
  
print(add\_tags("Dirk", "i"))  
print(add\_tags("I like coding in python \n It is a lot of fun", "b"))

Schrijf een functie die controleert of een gegeven getal al dan niet een priemgetal is. De functie geeft als resultaat True indien het een priemgetal is en False in het andere geval. Gebruik vervolgens die functie in een programma dat getallen, opgegeven door de gebruiker, controleert. Het resultaat verschijnt vervolgens op het scherm in een leesbare boodschap. Dit proces wordt herhaald tot de gebruiker de waarde “einde” opgeeft:

def is\_priemgetal(getal):  
 if getal > 1:  
 for deler in range(2, getal):  
 if getal % deler == 0:  
 return False  
 return True  
 else:  
 return False  
  
  
print(is\_priemgetal(3))  
print(is\_priemgetal(4))  
  
def check\_getal():  
 while True:  
 getal = input("Geef een positief geheel getal: ")  
 if getal == "einde": break  
 else: print("\tis " +  
 ("" if is\_priemgetal(int(getal)) else "g") + "een priemgetal")  
  
check\_getal()

# OBJECT-GEORIENTEERD PROGRAMMEREN

## Virtuele omgevingen en externe libraries:

**Pip** is het standaard package managementsysteem om Python packages te installeren en te beheren.

Stel dat we NumPy versie 1.16.1 willen installeren, dan doen we dat als volgt via terminal in pycharm:

pip install numpy==1.16.1

## Modules & packages:

* Script = python code in tekstfile opgeslagen met extensie ‘.py’. Is bedoeld om uitgevoerd te worden.
* Module = python code in tekstfile opgeslagen met extensie ‘.py’. Code in een module is bedoeld om in een ander stuk code gebruikt te worden.
* Package = verzameling van modules. Een folder die de modules als .py files bevat.  
  Een package kan ook sub packages bevatten.

## Een eigen module aanmaken:

Mymodule.py:

# Deze code opslaan onder de bestandsnaam mymodule.py  
# Begin module --------------------------  
# Functie in de module  
def verwelkom(naam):  
 print("Welkom,",naam,"!")  
  
# Variabele in de module  
persoon = {  
 "naam": "Alexander",  
 "leeftijd": 27,  
 "nationaliteit": "Belg"  
}

Mymodule gebruiken in test2.py:

import mymodule  
mymodule.verwelkom("Johnny")  
#output: Welkom, Johnny !  
  
# De module kan ook worden geimporteerd en hernoemd via een alias  
import mymodule as m  
print(m.persoon["leeftijd"])  
#output: 21  
  
# Het is ook omgelijk om slechts een deel van de module te importeren  
from mymodule import persoon  
print(persoon["naam"]) # !! Let op dat hier de 'mymodule.' niet gebruikt wordt

Met functie dir(0 zie je welke attributen een module heeft (paar door python en paar eigen gedefinieerde):

print(dir(mymodule))

*out: ['\_\_builtins\_\_', '\_\_cached\_\_', '\_\_doc\_\_', '\_\_file\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_', 'persoon', 'verwelkom']*