

20211109_Informationsmanagement_MV1

November 9, 2021

```
[3]: # Funktionen in Python.

# definition: eine Funktion in Python ist ein Codeblock, der nur ausgeführt
    ↳ wird,
# wenn er aufgerufen wird.

# Funktionen werden mit dem Schlüsselwort "def" definiert.

# Um eine Funktion aufzurufen, verwenden wir den Funktionsnamen gefolgt von
    ↳ Klammern.

def my_function():
    print('Hello from a function')

my_function()
```

Hello from a function

```
[4]: # Informationen können als Argumente an Funktionen übergeben werden.
# Argumente werden nach Funktionsnamen in Klammern angegeben.

# Wir können beliebig viele Argumente hinzufügen. Getrennt allerdings durch
    ↳ Kommas.

def my_function(Vorname): # Vorname ist das Argument
    print(Vorname + ' Nachname')

my_function('Fabian')
```

Fabian Nachname

```
[5]: # Die Begriffe "Argumente" und "Parameter" können für dasselbe verwendet werden:
# Informationen die an eine Funktion übergeben werden.

def my_function(Vorname, Nachname):
    print(Vorname + ' ' + Nachname)

my_function('Fabian', 'Kalmbach')
```

Fabian Kalmbach

[6]: *# Um eine Funktion einen Wert zurueckgeben zu lassen, verwenden wir "return".*

```
def my_function(x):  
    return 5*x  
  
print(my_function(3))
```

15

[8]:

```
def my_function(x,y):  
    return x**2+y**3  
  
print(my_function(3,2))
```

17

[10]: *# Python akzeptiert auch Funktionsrekursion, was bedeutet, dass eine
definierte Funktion sich selbst aufrufen kann.*

*# Dies hat den Vorteil, dass wir die Daten durchlaufen können,
um zu einem Ergebnis zu gelangen.*

*# In dem folgenden Beispiel, wird die Funktion "tri-recursion()" definiert,
welche sich selbst aufruft (recurse).*

*# Wir verwenden die "k-" Variable als Daten,
die bei jeder Wiederholung dekrementiert wird (-1).*

Die rekursion endet, wenn die Bedingung nicht grösser als 0 ist.

```
def tri_recursion(k):  
    if(k>0):  
        result = k + tri_recursion(k-1)  
        print(result)  
    else:  
        result=0  
    return result  
  
tri_recursion(6)
```

*# er addiert 1+0=1, das Ergebnis 1+2=3, das Ergebnis 3+3=6, das Ergebnis
↪ 6+4=10,...*

1

3

6

10
15
21

[10]: 21

```
[17]: # EOQ Model I -- my Python

import numpy as np

# Definieren wir die EOQ I Funktion:

def EOQ_I (A, D, H):
    """
    Drei Argumente werden definiert.
    A : Vorbereitungskosten der Maschine (setup Kosten).
    D : Bedarfe (jährliche Demand).
    H : Bestandshaltekosten.

    Die Funktion liefert dann:

    Q : optimale Bestellmenge.
    Y : optimale Kosten bei der optimalen Bestellmenge.
    D/Q : optimale Anzahl Bestellungen.
    12 Montage / Anzahl Bestellungen : Zeit zw. Bestellungen.
    AOC (annual ordering cost) : gesamte jährliche Bestellkosten.
    AHC (annual holding cost) : gesamte jährliche Bestandskosten.
    ATC (annual total cost) : gesamte jährliche Kosten
    """

    # Schritt 1. validieren, dass die Parameter alle Positiv sind.

    if (A>0 and D>0 and H>0):

        # Schritt 2. EOQ I Modell kalkulieren:

        Q = (np.sqrt(2*A*D/H))
        Y = (np.sqrt(2*A*D*H))
        number_of_orders = D/Q
        time_between_cycles = 12/number_of_orders
        AOC = D/Q*A
        AHC = Q/2*H
        ATC = AOC+AHC

        # Schritt 3. Return eine Liste mit den gewünschten Ergebnissen

        return [Q, Y, number_of_orders, time_between_cycles, AOC, AHC, ATC]
```

```

    else:
        print('Error. Alle Funktionsparameter müssen positiv sein.')

EOQ_I(10, 2400, 0.3)

```

[17]: [400.0, 120.0, 6.0, 2.0, 60.0, 60.0, 120.0]

```

[21]: # Übung. bitte definieren Sie eine Funktion welche die Würzel von
# x**2+y**3+z**3 berechnet für alle Positive Zahlen.
# Die Funktion sollte eine Fehlermeldung geben,
# falls eine negative Zahl berechnet werden soll.

def Laura_funktion(x,y,z):
    if (x>0 and y>0 and z>0):
        return np.sqrt(x**2+y**3+z**3) # 2+ 3+ 3
    else:
        print('Error. Alle Funktionsparameter müssen positiv sein.')

Laura_funktion(4,12,10)

```

[21]: 52.38320341483518

```

[22]: # Übung. ;-)

# Bitte erstelle eine Funktion, welche die EOQ_II und EOQ_III Modelle
↳ kalkuliert.
# Versteht sich nur Q und Y(Q).

```

```

[23]: # Übung. ;-)

# Bitte erstelle eine Funktion, welche die Y(Q')/Y(Q) im EOQ_I Modell
↳ kalkuliert.

```

```

[24]: # Diese Zeile ist nicht Prüfungsrelevant.

# Wir erstellen eine graphische Darstellung des EOQ_I Mdeolls
# Dafür müssen wir zwei Listen generieren für die zwei Achsen: Zeitliche
↳ Periode und Bestand

# Liste für die Perioden:

period = [0,2]

while period[-1]<12:
    period.append(period[-1])
    period.append(period[-1]+2)

```

```

# Liste vom Bestand

inventory = [400,0]

while len(inventory)<len(period):
    inventory.append(400)
    inventory.append(0)

import matplotlib.pyplot as plt

plt.figure(figsize=(15,5))
plt.plot(period,inventory)
plt.xlabel("Month")
plt.ylabel("Inventory Level")
plt.title("Economic Order Quantity I")

```

[24]: Text(0.5, 1.0, 'Economic Order Quantity I')



```

[25]: def absolute_value(num):
        """This function returns the absolute
        value of the entered number"""

        if num >= 0:
            return num
        else:
            return -num

print(absolute_value(2))

print(absolute_value(-4))

```

4

```
[26]: def greet(firstname, lastname):  
        print ('Hello', firstname, lastname)  
  
        greet(lastname='Jobs', firstname='Steve')  
        # passing parameters in any order using keyword argument
```

Hello Steve Jobs

```
[ ]:
```