20211111_Supplier_Management_MBW7

November 11, 2021

[2]:

```
# Functions in Python

# Definition. Is a codeblock that shall be run, only when called.

# Functions in Python are called with the word "def"

# In order to call a function we call the Function name followed by "()"

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

my_function() #shift+enter
```

```
Hello from my function
```

[5]:

```
# Information cann be brought to the function within the "()".
# this information is called "Arguments" or "Parameters"

# We can define many Arguments that should be separated with "Kommas"

def my_function(firstname, lastname): # firstname and lastname are the Arguments
    print(firstname + ' ' + lastname)

my_function('Anna', 'Krutsch')
```

```
Anna Krutsch
```

[6]:

```
# In order to get a value from a function, we use the order "return"

def my_function(x):
    return 5*x

print(my_function(3))
```

```
15
```

[7]:

```
# I can use many arguemnts in the function
```

```
def my_function(x,y):
    return x**2+y**2 # the double ** means x^2 and reads "x square"

print(my_function(2,3))
```

```
13
```

[11]:

```
# Example. Please write a function that delivers
# the absolute value of a number given in the argument.

def anna_absolute_value(x):
    if x>0:
        return x
    else:
        return -x

anna_absolute_value(-14)
```

[11]:

```
14
```

[12]:

```
# Example. Please write a function that greets a person in the morning

def greet(name):
    print('Hello, ' + name + '. Good morning!')

greet('Christian')
```

```
Hello, Christian. Good morning!
```

[17]:

```
# We can use a parametrized function
# A Function can have an unknown number of arguments. By putting * before the␣
 ↪parameter
# if you dont know the number of parameters, the user can pass the arguments.

def greet(*names):
    print('Hello, ', names[0], ', ', names[1], ' ,and ', names[2])

greet('Anna', 'Christian', 'Yannik')
```

```
Hello,  Anna ,  Christian  ,and  Yannik
```

[18]:

```
# We can return the value of a function with "return"

def sum(a,b):
    return a+b

sum(3,5)
```

[18]: 8

[22]:

```
# Function that delivers the values of the EOQ Model I.

import numpy as np

# Define the function EOQ I

def EOQ_I(A, D, H):
    # we use three Arguments for the function
    # A : Setup Cost
    # D : Yearly demand
    # H : Inventory Holding Cost
    # The function delivers:
    ## Q : Economic Order Quantity
    ## Y : Optimal Cost @ Q
    ## D/Q : Optimal number of production / supplier runs
    ## 12 / Number of production runs: Time between runs
    ## AOC : Annual Ordering Cost
    ## AHC : Annual Inventory Holding Cost
    ## ATC : Annual Total Cost (ATC=AOC+AHC)

    ### First Step. validate that all the data are positive

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

        ### Second Step. Calculate the EOQ I

        Q = (np.sqrt(2*A*D/H)) # Q=square_root(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 # D/Q (number of orders) * A (cost of an order)
        AHC = Q/2*H # Q/2 (average inventory) * H (cost of inventory)
        ATC = AOC+AHC

        ### Third Step. Return a list with the values

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

    ### Fourth Step. Give me an "Error" in case any parameter is not positive

    else:
        print('Error. All Parameters must be positive.')

EOQ_I(10, 2400, 0.3)
```

[22]:

```
[400.0, 120.0, 6.0, 2.0, 60.0, 60.0, 120.0]
```

[23]:

```
EOQ_I(-10, 2400, 0.3)
```

```
Error. All Parameters must be positive.
```

[24]:

```
# Example. Please define a function that calculates the square root of␣
 ↪x**2+y**3+z**3
# of all positive numbers in R^3. If a negative number is given, then deliver␣
 ↪an "Error".

# Condition is x,y,z > 0

def Kateryna_function(x,y,z):
    if (x>0 and y>0 and z>0):
        return np.sqrt(x**2+y**3+z**3)
    else:
        print('Error. All Parameters must be positive.')

Kateryna_function(12,23,24)
```

[24]:

```
161.6632302040263
```

[28]:

```
# Define a function to calculate the EOQ Model II (Supplier Model)

# S = Q*_I*sqrt(p/p+h)
# Q = Q*_I*sqrt(p+h/p)

def EOQ_II(A, D, H, p): # p is the backlog cost
    if (A>0 and D>0 and H>0 and p>0):
        S = (np.sqrt(2*A*D/H))*(np.sqrt(p/(p+H)))
        Q = (np.sqrt(2*A*D/H))*(np.sqrt((p+H)/p))
        Backlog = Q-S
        return [S,Q,Backlog]
    else:
        print('Error. All Parameters must be positive.')

EOQ_II(10, 2400, 0.3, 0.1)
```

[28]:

```
[200.0, 800.0, 600.0]
```

[31]:

```
# Define a function to calculate the EOQ Model III (Production Model)

# Q_III = sqrt((2*A*D)/H*(1-(D/K)))
# Y = sqrt(2*A*D*H*(1-(D/K)))

def EOQ_III(A, D, H, K): # K is the production rate
```

```
    if (A>0 and D>0 and H>0 and K>0):
        Q = np.sqrt((2*A*D)/H*(1-(D/K)))
        Y = np.sqrt(2*A*D*H*(1-(D/K)))
        return [Q,Y]
    else:
        print('Error. All Parameters must be positive.')

EOQ_III(10, 2400, 0.3, 3000)
```

```
[31]: [178.88543819998316, 53.66563145999495]
```

```
[ ]:
```