```
"""
Yooo eerste jaars is ye boi Marten met geweldige uitwerkingen :)

Hou er rekening mee dat deze uitwerkingen puur mogelijke oplossingen zijn
van de opdrachten en er niet één goed antwoord is!
"""

import numpy as np
import matplotlib.pyplot as plt

"""
Opdracht 1
==============================
"""
# np.arange() neemt de laatste waarde niet mee dus daarom 10.01
x = np.arange(-10, 10.01, 0.01)
y = np.sin(x)

"""
Opdracht 2
==============================
"""
# Defineer de aray
A = np.array([[1,2,3,4,5],
[2,6,9,12,13],
[-23,45,12,35,36],
[12,-34,56,1,2],
[2,-3,5,2,23]])

# Slice A voor alleen rij met index 1 en alle kolommen
B = A[1,:]

# Slice A voor rijen en kolommen met index 2 of hoger
C = A[2:,2:]

# Slice A voor rijen en kolommen met stapgrootte 2
D = A[::2,::2]
```

```
"""
Opdracht 3
==============================
"""
# a
k = [i**2 for i in range(2,102,2)]

# b
for i in range(2,len(k)+2):
    print(f'kwadraat van {2*i-2} is {k[i-2]}')

# c
for i,j in enumerate(k):
    print(f'kwadraat van {2*i+2} is {j}')

"""
Opdracht 4
==============================
"""
#a
def parabool(x,a,b=0,c=0):
    y = a*x**2 + b*x + c
    return y
print(parabool(1,2,c=3))

# b,c,d en e
plt.figure()
plt.plot(np.linspace(-10,10,1000),parabool(x=np.linspace(-10,10,1000),a=1,b=2,c=3)
         ,label='Parabool')
plt.scatter(np.linspace(-10,10,21), parabool(x=np.linspace(-10,10,21),a=1,b=2,c=3),
         marker='*',color='r',label='Gehele getallen')
plt.xlabel('x')
plt.ylabel('y = x**2 + 2x + 3')
plt.title('Parabool met gehele getallen gemarkeerd')
plt.legend()
plt.grid()
```

```
"""
Opdracht 5
==============================
"""
def passwordcheck(password):
    letters = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p'
               'q','r','s','t','u','v','w','x','y','z'] # letters om te checken
    cijfers = ['1','2','3','4','5','6','7','8','9','0'] # cijfer om te checken
    tekens = ['#','@','&']                               # tekens om the checken
    lettercount = 0 # beginwaarde om te checken of er minimaal 1 letter in zit
    cijfercount = 0 # beginwaarde om te checken of er minimaal 1 cijfer in zit
    tekencount = 0  # beginwaarde om te checken of er minimaal 1 teken in zit

    for letter in letters:
        if letter  in password:
            lettercount += 1    # Er zit minimaal 1 letter in!
    for cijfer in cijfers:
        if cijfer in cijfers:
            cijfercount += 1    # Er zit minimaal 1 cijfer in!
    for teken in tekens:
        if teken in password:
            tekencount += 1     # Er zit minimaal 1 teken in!
    if len(password)>7 and lettercount > 0 and cijfercount > 0 and tekencount > 0:
        return True    # return True als er ten minste 1 letter, 1 cijfer, 1 teken in zit en minimaal 8 lang is
    return False # In elk ander geval geeft het False, password werkt niet met hoofdletters!
```

```
"""
Opdracht 6
==============================
"""
# a
def pi_benadering(N):
    pi = 0
    for n in range(N+1):
        pi += 4 * ( (-1)**n / (2*n + 1) )
    return pi
# b
N=0
while abs(pi_benadering(N) - 3.141) > 0.001:
    N +=1
print(f'de waarde voor N om pi op 3 decimalen nauwkeurig te bapelen is: {N}')

"""
Opdracht 7
==============================
"""
btc_euro = np.array([102, 95, 110, 75, 43, 98, 101, 88]) # Array met waarden

def optimize_profit(btc_euro):
    bestprofit = [] # Array die gevuld gaat worden met alle mogelijke winsten
    for i in range(len(btc_euro)):
        for j in range(len(btc_euro)):
            if j > i:                   # Je kunt alleen btc verkopen die je eerder hebt gekocht
                profit = btc_euro[j] - btc_euro[i] # De mogelijke winst/verlies
                if profit > 0:                     # Alleen winst meenemen
                    bestprofit.append(profit)      # Mogelijk winst toevoegen aan array
    if len(bestprofit)==0:   # Als er geen mogelijkheid is tot winst, dus waarden in array
        return 0             # nemen alleen af of blijven gelijk, dan is winst 0
    maximalprofit = max(bestprofit) # Neem alleen de hoogst mogelijke winst
    return maximalprofit
```