Informatique scientifique:

Série 04

Andrin Reber

Exercice 1

a)
```python
import numpy as np
import matplotlib.pyplot as plt


def assign(x):
    if x < 0:
        return 0
    elif 0 <= x < 2:
        return 3 * np.sin(np.pi * x)
    else:
        return 2 * x**2 - 2.5 * x - 3
```

b)
```python
x = np.linspace(-2, 4, 100)
y = np.array([assign(i) for i in x])
# y = []
# for i in x:
#     y.append(assign(i))

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('exercice 1b)')
plt.show()
# >>> figure
```

Exercice 2

a)

```python
def pgdc(x, y):
    while y != 0:
        r = x % y
        x = y
        y = r
    return x
```

b)

```python
def arrow_for(x):
    for i in range(1, 2 * x):
        for j in range(0, min(i, 2 * x - i)):
            print('* ', end='')
        print()

def arrow_while(x):
    i = 1
    while i < 2 * x:
        j = 0
        while j < min(i, 2 * x - i):
            print('* ', end='')
            j += 1
        print()
        i += 1
```

```
c) def operation(x, y):
       if x % 2 == 0:
           z = 0
           for i in range(0, y):
               z += x
           return z
       else:
           z = x
           for i in range(1, y):
               z *= x
           return z
```

Exercice 3

```
a) # >>> n: 3, i: 1, a: 3, b: 2
   # >>> n: 6, i: 1, a: 2, b: 2
   # >>> n: 8, i: 2, a: 4, b: 4
   # >>> n: 11, i: 3, a: 6, b: 16
   # >>> n: 24, i: 3, a: 8, b: 16
   # >>> n: 28, i: 4, a: 15, b: 256
   # >>> n: 33, i: 5, a: 18, b: 70225
   # >>> n: 70244, i: 5, a: 21, b: 70225
   # a >= 20 -> the program is done
```

Exercice 4

a)

```python
def min_in_vector(x):
    y = x[0]
    for i in x:
        if i < y:
            y = i
    return y
```

b)

```python
def min_in_vector_two(x):
    y = 0
    for i in range(0, len(x)):
        if x[i] < x[y]:
            y = i
    return y, x[y]
```

c)

```python
a = [1, 2, 3, -10, 0, 4, 999, 10, 9, 8, 999, -10, 0, 999, 0]

print(min_in_vector(a))
# >>> -10

print(min_in_vector_two(a))
# >>> (3, -10)

# Comme le min_in_vector_two commence à 0, il rencontre le -10 a la 4eme position avant le -10 a la 12eme position.
# Comme il change le current min seulement quand il rencontre un nombre plus petit, il ne change pas de min quand il la meme nombre.
```

```
d) def mean_in_vector(x):
       y = 0
       z = 0
       for i in x:
           y += i
           z += 1
       return y / z

   print(mean_in_vector(a))
   # >>> 200.933
```

Exercice 5

a)
```python
def is_prime(x):
    if x < 2:
        return False
    for y in range(2, int(np.ceil(x))):
        if x % y == 0:
            return False
    return True
```

b)
```python
def find_primes(L):
    M = []
    for i in L:
        if is_prime(i):
            M.append(i)
    return M

print(find_primes([0,1,2,3,4,5,6,7,8]))
# >>> [2, 3, 5, 7]
```

Exercice 6

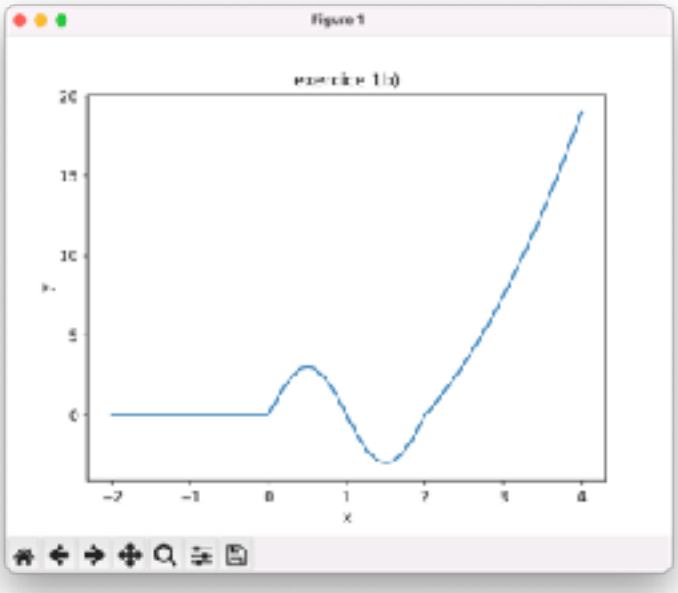
a)

```python
def solution(a, b, c):
    x = b**2 - 4 * a * c
    if x < 0:
        return None
    elif x == 0:
        return -b / (2 * a)
    else:
        return (-b + np.sqrt(x)) / (2 * a), (-b -
np.sqrt(x)) / (2 * a)


print(solution(1, 4, 5))
# >>> None
print(solution(-210, -333, 111))
# >>> (-1.8685866058500555, 0.2828723201357699)
print(solution(4, -12, 9))
# >>> 1.5
print(solution(5, -1, 0))
# >>> (0.2, 0.0)
```

Exercice 7

a)
```python
import turtle, random


turtle.tracer(0, 0) # remove to see drawing
def draw_spiral(x, y):
    turtle.penup()
    turtle.goto(x, y)
    turtle.pendown()
    for i in range(0, 50):
        turtle.forward(i)
        turtle.right(91)


for i in range(0, 100):
    color = (random.randint(0, 100) / 100, random.randint(0, 100) / 100, random.randint(0, 100) / 100)
    turtle.color(color)
    draw_spiral(random.randint(-400, 400), random.randint(-400, 400))
    turtle.update() # remove for more speed


turtle.update()
turtle.done()


# >>> figure 2
# Turtle module isn't compatible with pyplot.plot(), pyplot.hist(), etc. Python crashes if they're not commented out.
```

Figures:

Fig 1	
Fig 2