Corrige_Cours_Chapitre5_Fonctions

September 30, 2020

0.1 Exercice 1

[ ]:
```
def imc(m, t):
    """Retourne la classification de l'IMC pour une taille t
    et une masse m:
    0 pour sous-poids
    1 pour normal
    2 pour surpoids
    """
    mesure = m / t ** 2
    if mesure < 18.5:
        return 0
    elif mesure <= 25:
        return 1
    else:
        return 2
```

[1]:
```
def max2(a, b):
    """Renvoie le maximum de deux nombres
    passés en paramètres"""
    if a > b:
        return a
    else:
        return b
```

L'exécution d'une instruction **return** provoque une sortie de la fonction donc on peut omettre le **else**

[2]:
```
def max2(a, b):
    """Renvoie le maximum de deux nombres
    passés en paramètres"""
    if a > b:
        return a
    return b
```

Une première version

[3]:
```
def max3V1(a, b, c):
    """Renvoie le maximum de trois nombres
```

```
    passés en paramètres"""
    if a >= b:
        if a >= c:
            return a
        return c
    elif b >= c:
        return b
    else:
        return c
```

On peut précalculer le maximum de a et b avec `max2` pour éliminer un test

[4]:
```
def max3V2(a, b, c):
    """Renvoie le maximum de trois nombres
    passés en paramètres"""
    m = max2(a, b)
    if m >= c:
        return m
    return c
```

Dans ce cas on enchaîne deux appels de `max2`

[6]:
```
def max3V3(a, b, c):
    """Renvoie le maximum de trois nombres
    passés en paramètres"""
    m = max2(a, b)
    return max2(m, c)
```

Mais alors on peut composer

[8]:
```
def max3V4(a, b, c):
    """Renvoie le maximum de trois nombres
    passés en paramètres"""
    return max2(max2(a, b), c)
```

0.2 Exercice 2

[ ]:
```
def aumoinsun(a, b, c):
    """Retourne un booléen indiquant si au moins des trois réels
    a,b ou c est positif"""
    return a >= 0 or b >= 0 or c >= 0
```

[9]:
```
def tous(a, b, c):
    """Retourne un booléen indiquant si tous les réels
    a, b et c sont positifs"""
    return a >= 0 and b >= 0 and c >= 0
```

[ ]:
```python
def croissant(a, b, c):
    """Retourne un booléen indiquant si a,b et c
    sont dans l'ordre croissant"""
    return a <= b and b <= c
```

[ ]:
```python
def croissant2(a, b, c):
    """Retourne un booléen indiquant si a,b et c
    sont dans l'ordre croissant.
    Python permet de chainer les opérateurs de comparaison"""
    return a <= b <= c
```

[ ]:
```python
def bissextile(a):
    """Une année est bissextile si elle est divisible par 400
    ou pas divisible par 100 et divisible par 4"""
    return a % 400 == 0 or (a % 100 != 0 and a % 4 == 0)
```

0.3 Entrainement 1

[34]:
```python
def mention(note):
    """Prend en paramètre une note et renvoie
    une chaîne de caractères :
    - 'R' si note < 10
    - 'A' si 10 <= note < 12
    - 'AB' si 12 <= note < 14
    - 'B' si 14 <= note < 16
    - 'TB' si 16 <= note
    """
    if note < 10:
        return 'R'
    elif note < 12:
        return 'A'
    elif note < 14:
        return 'AB'
    elif note < 16:
        return 'B'
    else:
        return 'TB'
```

0.4 Exercice 3

[10]:
```python
from random import randint

def sommeDe(n):
    """Retourne la somme des résultats obtenus en lançant n
    dés à 6 faces"""
    s = 0
    for k in range(n):
```

```
        de = randint(1, 6)
        s = s + de
    return s


def urne():
    """Retourne le numéro d'une boule choisie dans une urne
    contenant 5 boules de numéro 1, 3 boules de numéro 2,
    et deux boules de numéro3"""
    choix = randint(1, 10)
    if choix <= 5:
        return 1
    elif choix <= 8:
        return 2
    else:
        return 3
```

0.5 Entrainement 2

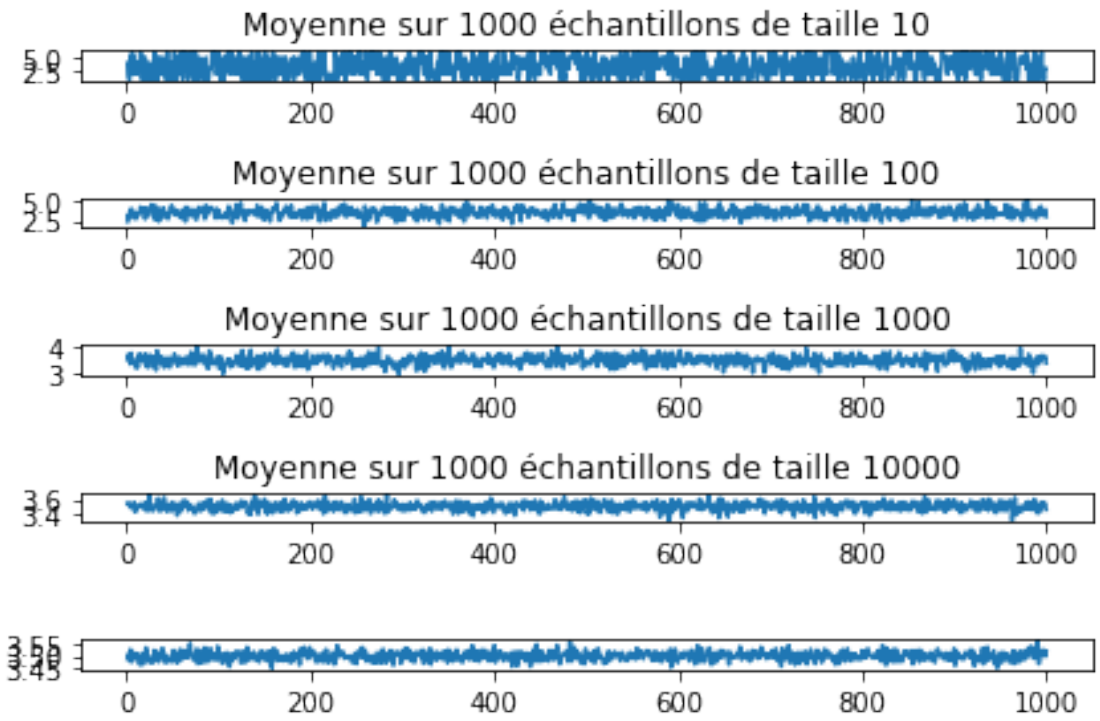
0.5.1 Question 1

[13]:
```
from random import randint

def moyenneDe(n):
    s = 0
    for k in range(n):
        s = s + randint(1, 6)
    return s / n
```

[11]:
```
# Import de la bibliothèque graphique matplotlib
import matplotlib.pyplot as plt
```

Fluctuation d'échantillonnage

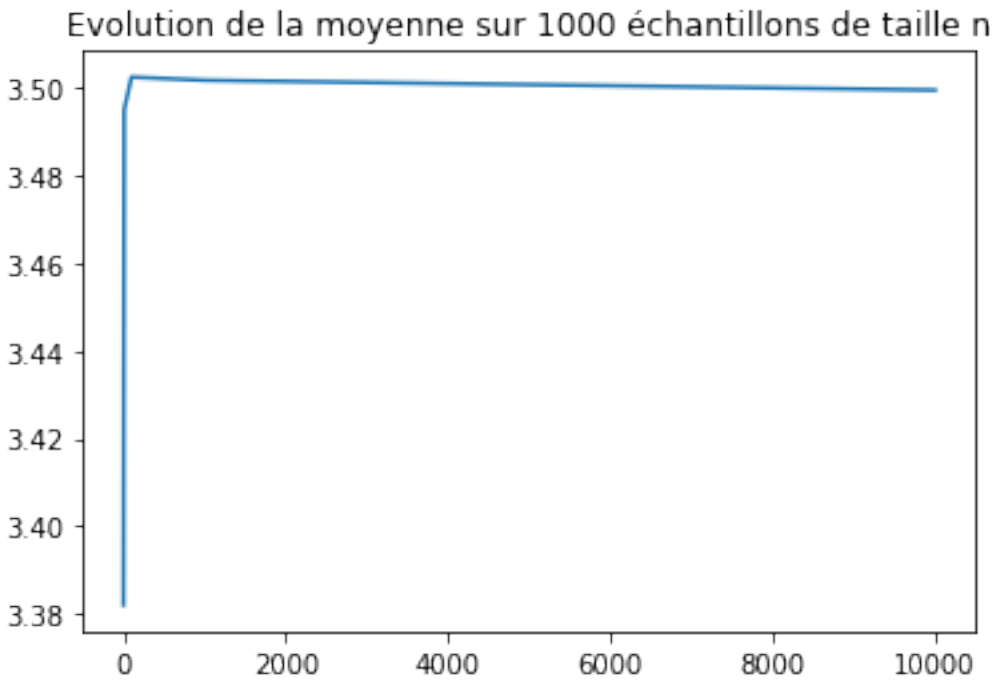
[14]:
```
%matplotlib inline

for k in range(5):
    n = 10 ** k
    numero_echantillon = [k for k in range(1, 1001)]
    liste_moyenne = [moyenneDe(n) for k in range(1000)]
    plt.title('Moyenne sur 1000 échantillons de taille {}'.format(n))
    plt.subplot(51 * 10 + (k+1))
    plt.plot(numero_echantillon, liste_moyenne)
plt.tight_layout()
```



Loi faible des grands nombres

[15]:
```
%matplotlib inline

taille_echantillon = [10 ** k for k in range(5)]
moyenne_moyenne = [ sum(moyenneDe(n) for k in range(1000))/1000 for n in␣
 ↪taille_echantillon]
plt.title('Evolution de la moyenne sur 1000 échantillons de taille n')
plt.plot(taille_echantillon, moyenne_moyenne)
```

[15]:
```
[<matplotlib.lines.Line2D at 0x7f4d1afd9280>]
```



0.5.2 Question 2

[16]:
```
def premier6():
    """Retourne le rang du premier 6"""
    k = 1
    while randint(1, 6) != 6:
        k = k + 1
    return k
```

0.5.3 Question 3

[17]:
```
def tempsAttente(n):
    """Retourne le temps d'attente moyen du premier 6
    sur un échantillon de n lancers"""
    t = 0
    for k in range(n):
        t = t + premier6()
    return t / n
```

0.6 Exercice 4

[18]:
```
from turtle import *

def spirale1(n):
    """Trace une spirale constituée de n carrés déformés"""
    penup()
    goto(0,0)
    pendown()
    c = 5
    for i in range(n):
        for j in range(4):
            forward(c)
            c = c + 10
            left(90)
    exitonclick()

##Permet de capturer l'exception Terminator générée lorsqu'on ferme la fenêtre␣
 ↪Turtle
try:
    reset()
except Terminator:
    pass

spirale1(10)
```

[19]:
```
from turtle import *

def spirale2(n, m):
    """Trace une spirale constituée de n polygones déformés
    à m cotés"""
    penup()
    goto(0,0)
    pendown()
    c = 5
    for i in range(n):
        for j in range(m):
            forward(c)
            c = c + 10
            left(360/m)
    exitonclick()

##Permet de capturer l'exception Terminator générée lorsqu'on ferme la fenêtre␣
 ↪Turtle
try:
    reset()
except Terminator:
```

```
    pass

spirale2(4, 6)
```

0.7 Entrainement 3

[32]:
```
from turtle import *
from math import sin, pi

def spirale3(n, m):
    shape('turtle')
    ecart = 10
    rayon = 20
    cote = 20
    for i in range(n):
        penup()
        goto(0,0)
        setheading(0)
        forward(rayon)
        pendown()
        #la difficulté est de calculer l'orientation de la tortue pour le tracé
 ↪du premier coté
        setheading(90+180/m)
        cote = 2 * rayon * sin(pi/m)
        for j in range(m):
            forward(cote)
            left(360/m)
        rayon = rayon + ecart
    exitonclick()

##Permet de capturer l'exception Terminator générée lorsqu'on ferme la fenêtre
 ↪Turtle
try:
    reset()
except Terminator:
    pass

spirale3(4, 6)
```