Correction_TP2

September 15, 2021

1 Boucles bornées

1.1 Exercice 1

(Q1) Faire afficher les entiers de 10 à 0 de manière décroissante.

[1]:
```
for k in range(10, -1, -1):
    print(k)
```

```
10
9
8
7
6
5
4
3
2
1
0
```

(Q2) Faire afficher les entiers pairs compris entre 0 et 50 dans l'ordre croissant.

[2]:
```
for k in range(0, 51, 2):
    print(k)
```

```
0
2
4
6
8
10
12
14
16
18
20
22
24
```

```
26
28
30
32
34
36
38
40
42
44
46
48
50
```

(Q3) Faire afficher les entiers pairs compris entre 0 et 50 dans l'ordre décroissants.

[4]:
```
for k in range(50, -1, -2):
    print(k)
```

```
50
48
46
44
42
40
38
36
34
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
0
```

1.2 Exercice 2

(Q1)

[5]:
```
for i in range(5):
    for j in range(i, 5):
        print(j, end="")
    print()
```

```
01234
1234
234
34
4
```

(Q2) Afichage a

[6]:
```
for i in range(5):
    for j in range(0, i + 1):
        print(j, end="")
    print()
```

```
0
01
012
0123
01234
```

[ ]:
```
(Q2) Afichage b
```

[7]:
```
for i in range(5):
    for j in range(i, i + 5):
        print(j, end="")
    print()
```

```
01234
12345
23456
34567
45678
```

1.3 Exercice 3

[8]:
```
trace = True
f = 0
g = 1
for i in range(12):
    if trace:
        print(f"f = {f} et g={g}")
    f = f + g
    g = f - g
```

```
f = 0 et g=1
f = 1 et g=0
f = 1 et g=1
f = 2 et g=1
f = 3 et g=2
f = 5 et g=3
f = 8 et g=5
f = 13 et g=8
f = 21 et g=13
f = 34 et g=21
f = 55 et g=34
f = 89 et g=55
```

2 Conditionnelles

2.1 Exercice 4

Pour transporter des poteaux sur une île, on dispose d'une barge dont la masse utile est de 1800 kg. Écrire un programme qui demande à l'utilisateur la masse d'un poteau puis le nombre de poteaux et qui affiche le message « Risque de surcharge ! » si la masse utile est dépassée.

[9]:
```
masse = float(input("Masse du poteau ?"))
nombre = int(input("Nombre de poteaux ?"))
masse_utile = masse * nombre
if masse_utile > 1800:
    print("Risque de surcharge !")
```

```
Masse du poteau ?10
Nombre de poteaux ?190
Risque de surcharge !
```

2.2 Exercice 5

[ ]:
```
n = int(input("Entrez le nombre de l'année : "))
if n % 4 != 0 :  #année non divisible par 4  =>  non bissextile
    print('Année non bissextile')
elif n % 100 != 0:  #année divisible par 4 mais pas par 100 => bissextile
    print('Année bissextile')
elif  n % 400 != 0:  #année divisible par 100 mais pas par 400 => non bissextile
    print('Année non bissextile')
else:   #année divisible par 100 et par 400 => bissextile
    print('Année bissextile')
```

3 Boucles non bornées

3.1 Exercice 6

Écrire un programme qui permet de saisir un mot de passe sous la forme d'une chaîne de caractères et qui ensuite redemande ce mot de passe jusqu'à ce qu'il soit correct.

[ ]:
```
secret = "sesame"
while input('Mot de passe ?') != secret:
    print("Mot de passe incorrect")
print("Mot de passe correct")
```

3.2 Exercice 7

On appelle logarithme entier d'un nombre réel x > 1, le nombre de fois qu'il faut le diviser par 2 pour obtenir un nombre inférieur à 1. Écrire un programme qui permet de saisir un nombre x et qui affiche son logarithme entier.

[12]:
```
def logarithme_entier(x):
    """
    Paramètre : x un réel
    Précondition : x > 1
    Valeur renvoyée : un entier naturel
    Postcondition : renvoie le plus petit entier n tel que  x / 2 ** n < 1
    """
    #précondition
    assert x > 1
    n = 0
    p = 1
    #invariant  p = 2 ** n
    while p <= x:
        n = n + 1
        p = p * 2
    return n

#test unitaires
assert logarithme_entier(1.5) == 1
assert logarithme_entier(2) == 2
assert logarithme_entier(7.9) == 3
assert logarithme_entier(8) == 4
assert logarithme_entier(8.2) == 4
```

3.3 Exercice 8

[11]:
```
def jour_semaine(m, d, y):
    """
    Paramètres :
        m un entier représentant le mois dans l'année 1<=m<=12
```

```
        d un entier représentant le jour  dans le mois 1<=d<=31
        y un entier représentant l'année
    Valeur renvoyée :
        un entier d0 représentant le rang du jour de la semaine entre 1 et 7
    """
    y0 = y - (14 - m)//12
    x = y0 + y0//4 - y0//100 + y0//400
    m0 = m + 12*((14-m)//12) - 2
    d0 = (d + x + (31 * m0)//12)%7
    return d0

#tests unitaires
assert jour_semaine(2,14,2000) == 1
assert jour_semaine(2,14,1900) == 3
assert jour_semaine(7,14,1789) == 2
```

3.4 Exercice 9

[ ]:
```
def bissextile(a):
    """Signature : bissextile(a:int)->bool
    Postcondition : Détermine si l'année a est bissextile"""
    return (a % 100 != 0 and a % 4 == 0) or (a % 400 == 0)

#tests unitaires
assert bissextile(2020)
assert not bissextile(2019)
assert bissextile(2000)
assert not bissextile(19000)
```