INFO-H-100 - Programmation
TP 3 - Tests et fonctions

Lorsque l'exercice demande d'écrire une fonction, écrivez-la fonction demandée et testez-la avec plusieurs valeurs pertinentes.

Ex. 1. Qu'affiche ce programme ? Expliquez avec quelques valeurs représentatives. Pouvez-vous le simplifier (le remplacer par un programme équivalent mais plus simple) ?

```
def f(a,b):
    if a > 0:
        if b > 1:
            print(a)
        else:
            print(b)
    else:
        if b > 1:
            print(a+b)
        else:
            print(b)
```

Ex. 2. Quelles sont les valeurs des variables au fur et à mesure de l'exécution de ces instructions ?

```
a = 2
b = 3
c = 4
test1 = True
test2 = (b >= a) and (c >= b)
test3 = test1 or test2
arret = test3 and (not test2)
a += 1
b -= 1
c -= 2
test1 = True
test2 = (b >= a) and (c >= b)
test3 = test1 or test2
arret = arret or test2
```

Ex. 3. Écrire une fonction qui vérifie si un nombre entier est valide pour le Lotto, c'est-à-dire qu'il est compris entre 1 et 42 inclus.

Ex. 4. Écrire une fonction qui vérifie qu'une année est bissextile. Sont bissextiles les années divisibles par 4 mais non divisibles par 100 ou les années divisibles par 400.

Ex. 5. Écrire une fonction qui vérifie qu'un point est dans un rectangle horizontal. Un point est représenté par un tuple `(x,y)` et un rectangle est représenté par deux points : le coin supérieur gauche et le coin inférieur droit. Nous supposons que les x croissent vers la droite et que les y croissent vers le haut.

Ex. 6. Écrire une fonction qui reçoit 4 points et qui vérifie si ces 4 points forment un carré horizontal. Ces 4 points étant respectivement le coin supérieur gauche, le point supérieur droit, le coin inférieur gauche et le coin inférieur droit.

Ex. 7. Écrire une fonction qui reçoit les valeurs de trois dés à six faces et qui vérifie que ces dés forment un 421.

Ex. 8. Écrire une fonction qui reçoit trois nombres et qui renvoie un tuple composé de ces trois nombres dans l'ordre croissant.

Ex. 9. Écrire une fonction qui reçoit trois nombres et qui renvoie les deux plus grands.

Ex. 10. Soit l'équation du second degré $ax^2 + bx + c = 0$. Écrivez une fonction qui reçoit a, b et c et qui calcule et affiche les solutions réelles (s'il y en a) de cette équation. Pour rappel, $\Delta = b^2 - 4ac$.
- Si Δ est strictement positif, l'équation admet deux solutions $x_1 = \frac{-b-\sqrt{\Delta}}{2a}$ et $x_2 = \frac{-b+\sqrt{\Delta}}{2a}$.
- Si Δ est nul, l'équation admet une solution $\frac{-b}{2a}$.

– Si Δ est négatif, l'équation n'admet pas de solution réelle.

Ex. 11. On représente un instant comme un triplet (heure, minutes, secondes). Écrire un fonction qui reçoit un instant et qui renvoie l'instant de la seconde suivante. On considère que l'instant suivant de 23 heures 59 minutes et 59 secondes est 0 heure 0 minute et 0 seconde.

Ex. 12. On représente un instant comme un triplet (heure, minutes, secondes). Écrire une fonction qui additionne deux instants. On considère que l'instant suivant de 23 heures 59 minutes et 59 secondes est 0 heure 0 minute et 0 seconde.

INFO-H-100 - Programmation
TP 3 - Tests et fonctions
Corrections

Solution de l'exercice 1:

Si b est inférieur ou égal à 1, ce programme affiche b. Sinon, si a est strictement supérieur à 0, ce programme affiche a. Sinon, c'est-à-dire que b est strictement supérieur à 1 et a est inférieur ou égal à 0, ce programme affiche la somme de a et de b.

```
f(10,0)  #affiche 0
f(10,1)  #affiche 1
f(1,2)   #affiche 1
f(10,3)  #affiche 10
f(0,10)  #affiche 10
f(-10,3) #affiche -7
```

Voici une version simplifiée :

```
def f(a,b):
    if b<=1:
        print(b)
    elif a>0:
        print(a)
    else:
        print(a+b)
```

Solution de l'exercice 2:

instruction	a	b	c	test1	test2	test3	arret
`a = 2`	2						
`b = 3`	2	3					
`c = 4`	2	3	4				
`test1 = True`	2	3	4	True			
`test2 = (b >= a) and (c >= b)`	2	3	4	True	True		
`test3 = test1 or test2`	2	3	4	True	True	True	
`arret = test3 and (not test2)`	2	3	4	True	True	True	False
`a += 1`	3	3	4	True	True	True	False
`b -= 1`	3	2	4	True	True	True	False
`c -= 2`	3	2	2	True	True	True	False
`test1 = True`	3	2	2	True	True	True	False
`test2 = (b >= a) and (c >= b)`	3	2	2	True	False	True	False
`test3 = test1 or test2`	3	2	2	True	False	True	False
`arret = arret or test2`	3	2	2	True	False	True	False

Solution de l'exercice 3:

```
def est_valide_lotto(nombre):
    return nombre >= 1 and nombre <= 42

print(est_valide_lotto(-1)) #affiche False
print(est_valide_lotto(1))  #affiche True
print(est_valide_lotto(42)) #affiche True
print(est_valide_lotto(98)) #affiche False
```

Solution de l'exercice 4:

```
def est_bissextile(annee):
    return annee%400==0 or (annee%100 != 0 and annee%4==0)

print(est_bissextile(1998)) #affiche False
print(est_bissextile(2000)) #affiche True
print(est_bissextile(2011)) #affiche False
print(est_bissextile(2012)) #affiche True
```

Solution de l'exercice 5:

```
def est_dans_rectangle(point, superieur_gauche, inferieur_droit):
    """ verifie si le point (tuple (x,y)) est dans le rectangle
        defini par ses coins superieur gauche et inferieur droit"""
    (x,y) = point
    (x1,y1) = superieur_gauche
    (x2,y2) = inferieur_droit
    return (x >= x1 and x <= x2 and y >= y2 and y <= y1)

print(est_dans_rectangle((1,1),(0,2),(3,0)))  #affiche True
print(est_dans_rectangle((0,0),(0,2),(3,0)))  #affiche True
print(est_dans_rectangle((-1,0),(0,2),(3,0))) #affiche False
```

Solution de l'exercice 6:

```
def est_carre(sup_g, sup_d, inf_g, inf_d):
    """verifie que le carre horizontal dont les 4 sommets sont sup_g,
       sup_d, inf_g et inf_d, respectivement les coins superieur gauche,
       superieur droit, inferieur gauche et inferieur droit."""
    (x1, y1) = sup_g
    (x2, y2) = sup_d
    (x3, y3) = inf_g
    (x4, y4) = inf_d
    return(x1 == x3 and y1 == y2 and x2 == x4 and y3 == y4 and x2-x1 == y1-y3)

print(est_carre((0,1),(1,1),(0,0),(1,0))) #affiche True
print(est_carre((0,1),(2,1),(0,0),(2,0))) #affiche False
print(est_carre((0,1),(1,1),(0,0),(2,0))) #affiche False
```

Solution de l'exercice 7:

```
def est_421(x,y,z):
    """verifie que les trois entiers x,y et z forment un 421"""
    return ((x==4 or y==4 or z==4) and
            (x==2 or y==2 or z==2) and
            (x==1 or y==1 or z==1))
```

Solution naïve :

```
def est_421(x,y,z):
    """verifie que les trois entiers x,y et z forment un 421"""
    if x == 4 and y == 2 and z == 1:
        return True
    elif x == 4 and y == 1 and z == 2:
        return True
    elif x == 2 and y == 4 and z == 1:
        return True
    elif x == 2 and y == 1 and z == 4:
        return True
    elif x == 1 and y == 4 and z == 2:
        return True
    elif x == 1 and y == 2 and z == 4:
        return True
    else:
        return False

print(est_421(4,2,1))    #affiche True
print(est_421(4,1,2))    #affiche True
print(est_421(2,4,1))    #affiche True
print(est_421(2,1,4))    #affiche True
print(est_421(1,4,2))    #affiche True
print(est_421(1,2,4))    #affiche True
```

```
print(est_421(4,2,2))     #affiche False
print(est_421(-1,2.0,4)) #affiche False
```

Autre solution identique au niveau du flot d'exécution :

```
def est_421(x,y,z):
  """verifie que les trois entiers x,y et z forment un 421"""
  return (x == 4 and y == 2 and z == 1 or
          x == 4 and y == 1 and z == 2 or
          x == 2 and y == 4 and z == 1 or
          x == 2 and y == 1 and z == 4 or
          x == 1 and y == 4 and z == 2 or
          x == 1 and y == 2 and z == 4)
```

Une autre solution plus rapide (analysez le flot d'éxécution) :

```
def est_421(x,y,z):
    """verifie que les trois entiers x,y et z forment un 421"""
    if x == 4:
        if y == 2:
            if z == 1:
                return True    #ordre 4 2 1
        elif y == 1:
            if z == 2:
                return True    #ordre 4 1 2
    elif x == 2:
        if y == 4:
            if z == 1:
                return True    #ordre 2 4 1
        elif y == 1:
            if z == 4:
                return True    #ordre 2 1 4
    elif x == 1:
        if y == 4:
            if z == 2:
                return True    #ordre 1 4 2
        elif y == 2:
            if z == 4:
                return True    #ordre 1 2 4
    return False
```

Une solution plus élégante est de trier ces trois nombres avant de tester leurs valeurs comme fait dans l'exercice 8 : .

```
def est_421(x,y,z):
    """verifie que les trois entiers x,y et z forment un 421"""
    (x,y,z) = dans_l_ordre(x,y,z)
    return x == 1 and y == 2 and z == 4
```

Solution basée sur le fait que les dés ont 6 faces de 0 à 6 :

```
def est_421(x,y,z):
    """verifie que les trois entiers x,y et z forment un 421"""
    return x+y+z==7 and (x == 4 or y == 4 or z == 4)
```

Solution de l'exercice 8:

```
def dans_l_ordre3(x,y,z):
    if x < y:
        if y < z:
            return (x,y,z)
        elif x < z:
            return (x,z,y)
        else:
            return (z,x,y)
    else:
        if x < z:
            return (y,x,z)
        elif y < z:
            return (y,z,x)
        else:
            return (z,y,x)

print(dans_l_ordre3(1,2,3))  #affiche (1,2,3)
print(dans_l_ordre3(1,1,1))  #affiche (1,1,1)
print(dans_l_ordre3(3,2,1))  #affiche (1,2,3)
print(dans_l_ordre3(1,3,2))  #affiche (1,2,3)
print(dans_l_ordre3(3,1,1))  #affiche (1,1,3)
```

Solution de l'exercice 9:

```
def deux_maxs(a,b,c):
    if a < b and a < c: #a est le minimum
        return(b,c)
    elif b < c:         #b est le minimum
        return(a,c)
    else:               #c est le minimum
        return(a,b)

print(deux_maxs(1,2,3)) #affiche (2,3)
print(deux_maxs(2,2,3)) #affiche (2,3)
print(deux_maxs(1,2,2)) #affiche (2,2)
print(deux_maxs(3,2,1)) #affiche (3,2)
print(deux_maxs(1,3,2)) #affiche (3,2)
```

Explication de l'algorithme : trouver le minimum plutôt que d'essayer de déterminer les deux nombres les plus grands.

Solution de l'exercice 10:

```
import math

def sols_deux_degre(a,b,c):
    d = b**2 - 4*a*c
    if d > 0:
        ds = math.sqrt(d)
        print((-b+ds)/2.0/a, (-b-ds)/2.0/a)
    elif d == 0:
        print(-b/2.0/a)

sols_deux_degre(1,2,3) #x2 + 2b + 3c : pas de solution (n'affiche rien)
sols_deux_degre(1,2,1) #x2 + 2b + c : delta = 0, une solution
sols_deux_degre(1,4,1) #x2 + 4b + c : delta > 0, deux solutions
```

Solution de l'exercice 11:

```
def instant_suivant(instant):
    (h,m,s) = instant
    s += 1
    if(s == 60):
        s = 0
        m += 1
    if(m == 60):
        m = 0
        h += 1
    if(h == 24):
        h = 0
    return (h,m,s)


print(instant_suivant((23,59,59))) #affiche (0, 0, 0)
print(instant_suivant((10,10,59))) #affiche (10, 11, 0)
print(instant_suivant((10,59,59))) #affiche (11, 0, 0)
print(instant_suivant((0,0,0)))    #affiche (0, 0, 1)
```

Solution de l'exercice 12:

```
def somme_instants(instant1,instant2):
    (h1,m1,s1) = instant1
    (h2,m2,s2) = instant2
    s = s1+s2
    m = m1+m2
    h = h1+h2
    if(s >= 60):
        m += s/60
        s = s%60
    if(m >= 60):
        h += m/60
        m = m%60
    if(h >= 24):
        h = h%24
    return (h,m,s)


print(somme_instants((10,10,10),(10,10,10))) #affiche(20, 20, 20)
print(somme_instants((10,10,10),(10,10,50))) #affiche(20, 21, 0)
print(somme_instants((10,10,10),(10,50,50))) #affiche(21, 1, 0)
print(somme_instants((10,10,10),(13,49,50))) #affiche(0, 0, 0)
```