I - PLANCHE 3

Exercices

1 Dans chacun des cas ci-dessous, déterminer le contenu des variables `a` et `b` à la fin de l'exécution du programme.

1.

Code python
```
a = 1
b = a + 3
a = 3
```

2.

Code python
```
a = 7
a = 8
a = a + 2
```

3.

Code python
```
a = 5
b = a + 4
a = a + 1
b = a - 4
```

4.

Code python
```
a = 4
b = 6
a = b
b = a
```

5.

Code python
```
a = 4
b = 6
temp = b
b = a
a = b
```

6.

Code python
```
a = 4
b = 6
a = a + b
b = a - b
a = a - b
```

2 **1.** Déterminer le contenu des variables `a` et `b` à l'issue de l'exécution du programme ci-dessous.

Code python
```
a = 3
b = 6
if a > 5 or b != 3:
    b = 4
else:
    b = 2
```

2. On considère les trois fonctions ci-dessous, dont on a omis la spécification complète.

Code python
```
def f1(val):
    """int->str"""
    if val > 50:
        return "B"
    elif val > 100:
        return "A"
    else:
        return "C"
```

Code python
```
def f2(val):
    """int->str"""
    if val > 100:
        return "A"
    elif val > 50:
        return "B"
    else:
        return "C"
```

Code python
```
def f3(val):
    """int->str"""
    if val < 0:
        return 0
    return val
```

Déterminer le résultat de l'évaluation de :

a. `f1(0)`
b. `f1(25)`
c. `f1(110)`
d. `f2(0)`
e. `f2(25)`
f. `f2(110)`
g. `f3(-5)`
h. `f3(10)`
i. `f3(0)`

3 Quelle est la valeur affichée par la console après la séquence d'instructions suivantes :

Résultat

```
>>> largeur, longueur = 5, 7
>>> aire = largeur  longueur
>>> largeur, longueur = 10, 21
>>> aire
```

4 **1.** Écrire le corps de la fonction `valeur_absolue` dont on donne la spécification et les tests ci-dessous. Écrire 2 tests supplémentaires pour la fonction `valeur_absolue`

Code python

```
def valeur_absolue(x):
    """ float -> float
    Retourne la valeur absolue de x, c'est à dire x si x est négatif, -x
    ↪ sinon """
    # ... à compléter

assert valeur_absolue(3) == 3
assert valeur_absolue(-3) == 3
assert valeur_absolue(0) == 0
# ... à compléter
```

2. a. Écrire une fonction `max2` qui prend en argument deux entiers `a` et `b` et qui renvoie le plus grand des deux.

b. Écrire une fonction `max3` qui prend en argument trois entiers `a`, `b`, et `c` et qui renvoie le plus grand des trois.

c. En utilisant un appel à la fonction `max3`, écrire une fonction `max4` qui prend en argument quatre entiers `a`, `b`, `c` et `d` et qui renvoie le plus grand des quatre.

5 C'est bientôt les soldes, et un commerçant camerounais décide d'écrire un code python afin de l'aider à calculer le prix de ses produits. Il possède une toute petite boutique, qui contient :

Objet	Tasses	Assiettes	Vases
Nombre	20	15	5
Prix à l'unité en euros	10	30	70

1. Initialement, il se dit qu'il va faire très simple : tous les objets sont à moins 30%.

a. Écrire une fonction python `tout_en_solde` qui prend en argument un prix `p` en euros et qui renvoie le nouveau prix, calculé en appliquant une réduction de 30%.

b. Écrire les instructions python qui affectent aux variables `prix_tasses`, `prix_assiettes` et `prix_vases` leur nouveau prix.

Quelle(s) instruction(s) doit-on écrire pour faire afficher ces prix ?

2. Après mûre réflexion, le commerçant se dit qu'il faut être un peu plus agressif dans ses réductions : certains items sont encore trop chers et il faut leur appliquer une remise plus importante. Il utilise pour cela la méthode suivante :

- si le prix `p` de l'objet est supérieur ou égal à 50€, alors on applique une réduction de 60%.
- sinon on applique une réduction de 30% comme avant.

a. Écrire une fonction python `grosses_soldes` qui prend en argument un prix `p` et qui renvoie le nouveau prix, calculé en appliquant la nouvelle méthode.

b. Écrire un jeu de test qui vérifie que les nouveaux prix des tasses, assiettes, et vases sont correctement calculés par la fonction `grosses_soldes`.

3. En étudiant les horaires d'affluence de son magasin, il se rend compte du phénomène suivant : lors de la période janvier-juin (inclus), il y a vraiment beaucoup de clients dans son magasin. Par contre, dans la période juillet-décembre, il n'y a pas grand monde. Il pense à la troisième méthode :

- si le numéro du mois en cours est compris entre 1 et 6 (inclus), alors il ne fait aucune réduction ! Il y a bien assez de clients, il y aura bien quelqu'un qui achètera ses objets.
- si le numéro du mois en cours est compris entre 7 et 12 (inclus), alors il applique la méthode des `grosses_soldes` pour calculer le nouveu prix.

a. Calculer :

- le prix d'une tasse en mai ;
- le prix d'une assiette en octobre ;
- le prix d'un vase en novembre ;
- le prix d'un vase en janvier.

b. Écrire une fonction `soldes_malines` qui prend en argument un prix `p` et un entier `numero_mois` et qui renvoie le nouveau prix de l'objet au mois `numero_mois`. Vous pourrez faire appel aux fonctions précédentes.

c. En avril 2026, lors d'une sortie à Yaoundé, M. Picard tombe sous le charme désuets des céramiques que vend le commerçant. Il achète l'intégralité de son stock ! Écrire un code python qui permet de calculer puis d'afficher la somme gagnée par le commerçant en vendant tous ses objets en avril.

6 Déterminer la valeur de la variable `d` après exécution du programme dans chacun des cas ci-dessous.

1.

Code python

```
a, b, c, d = 1, 2, 3, 4
for k in range(2):
    a = a + b
    b = b + a
    c = c + b
    d = d + c
```

2.

Code python

```
a, b, c, d = 1, 2, 3, 4
 for k in range(2):
     a = a + b
     b = b + a
     c = c + b
 d = d + c
```

3.

Code python

```
a, b, c, d = 1, 2, 3, 4
for k in range(2):
    a = a + b
    b = b + a
c = c + b
d = d + c
```

4.

Code python

```
a, b, c, d = 1, 2, 3, 4
for k in range(2):
    a = a + b
b = b + a
c = c + b
d = d + c
```

7 Le tableau ci-contre permet de dresser la trace d'exécution d'un programme contenant une boucle `for`. À la fin de chaque exécution du bloc répété, on note l'état des variables dans la ligne correspondante.

	i	x
Avant le début de la boucle		
Fin de la 1er itération		
Fin de la 2e itération		
...		
...		
...		

1.

Code python
```
u = 15
for i in range(3):
    u = 2*u + 1
print(u)
```

2.

Code python
```
s = 0
for i in range(5):
    s = s + i
print(s)
```

3.

Code python
```
p = 1
for i in range(4):
    p = p * i
print(p)
```

4.

Code python
```
s = 3
for i in range(2, 9):
    s = s + i
print(s)
```

5.

Code python
```
chaine = ""
for c in "tr", "uc", "de", "fo"
 ↪  "u!":
    chaine = chaine + c
print(chaine)
```

6.

Code python
```
for k in range(4):
    print(k**2)
```

7.

Code python
```
for i in range(9):
    if i % 3 == 0:
        print(i)
```

Pour chacun des codes ci-dessus.

1. Dresser la traces d'exécution du programme.

2. Déterminer l'affichage réalisé par le programme.

8 **1.** Écrire le corps de la fonction ci-dessous.

Code python
```
def somme_carres(n):
    """int -> int
    precondition: n >= 1
    retourne la somme des n premiers carrés."""
    ...

assert somme_carres(3) == 1 + 2 ** 2 + 3 ** 2
assert somme_carres(4) == 1 + 4 + 9 + 16
assert somme_carres(1) == 1
assert somme_carres(10) == 385
```

2. On souhaite écrire une fonction python `somme_pairs` qui prend en argument un entier `n` et qui renvoie la somme de tous les entiers pairs compris entre 0 et n strictement.

a. Écrire un jeu de test que doit satisfaire cette fonction. Vous écrirez au moins 5 tests différents.

b. Écrire le code de la fonction `somme_pairs`. Attention à la rédaction de la spécification de la fonction.