DÉPARTEMENT
Sciences Informatiques

DU Développeur II

Algorithmique et Python

Jean-Luc.Bourdon@u-cergy.fr

TP n°03 - correction type
Fonctions

Première partie

Pour commencer

Exercice 1 : Fonction sans paramètre ni sortie

Écrivez une fonction appelée `affiche_bonjour()` qui demande à l'utilisateur son prénom et affiche ensuite « `Bonjour` » suivi du prénom de l'utilisateur.

Algorithme attendu

```
1  #
2  # DU II : TP n°3, exercice 1, algorithme
3  #
4
5  # (1) prototype (encore appelé signature) de la fonction
6  fonction affiche_bonjour( / ) -> /
7
8  # (2) définition de la fonction
9  fonction affiche_bonjour()
10 variables
11     paramètres
12         /
13     internes
14         chaîne prenom           # prénom de l'utilisateur
15     en sortie
16         /
17 début
18     # initialisation et traitement
19     écrire("Votre prénom ? ")
20     lire(prenom)
21     # restitution des résultats
22     écrire(f"Bonjour {prenom} !!!")
23 fin
24
25 ----------------------------------------
26
27 programme principal
28 variables
29     en entrée
30         /
31     en sortie
32         /
33 début
34     # (3) appel de la fonction
35     affiche_bonjour()
36 fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 1, Python
#

## Déclaration de la fonction affiche_bonjour()
def affiche_bonjour():
    prenom = input("Votre prénom ? ")
    print(f"Bonjour {prenom} !!!")


##
## Début du programme principal
##

## Appel de la fonction affiche_bonjour()
affiche_bonjour()
```

Exercice 2 : Fonction avec un paramètre et sans sortie

Écrivez une fonction appelée `affiche_salutation(formule)` qui dépend d'un paramètre `formule`. Cette fonction demande le prénom de l'utilisateur et affiche une formule de salutation suivi du prénom. Par exemple `affiche_salutation("Coucou")` afficherait « `Coucou` » suivi du prénom donné par l'utilisateur.

Algorithme attendu

```
1  #
2  # DU II : TP n°3, exercice 2, algorithme
3  #
4
5  # (1) prototype (encore appelé signature) de la fonction
6  fonction affiche_salutation(formule: chaîne) -> /
7
8  # (2) définition de la fonction
9  fonction affiche_salutation(formule: chaîne) -> /
10 variables
11     paramètres
12         chaîne formule          # message
13     internes
14         chaîne prenom           # prénom de l'utilisateur
15     en sortie
16         /
17 début
18     # initialisation et traitement
19     écrire("Votre prénom ? ")
20     lire(prenom)
21     # restitution des résultats
22     écrire(f"{formule} {prenom} !!!")
23 fin
24
25 ----------------------------------------
26
27 programme principal
28 variables
29     en entrée
30         /
31     en sortie
32         /
33 début
34     # (3) appel de la fonction
35     affiche_salutation("Bonjour")
36     # (3) appel de la fonction
37     affiche_salutation("Cao")
38 fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 2, Python
#

## Déclaration de la fonction affiche_salutation()
def affiche_salutation(formule: str):
    prenom = input("Votre prénom ? ")
    print(f"{formule} {prenom} !!!")


##
## Début du programme principal
##

## Appel de la fonction affiche_salutation()
affiche_salutation("Bonjour")
affiche_salutation("Cao")
```

Exercice 3 : Fonction sans paramètre et avec sortie

Écrivez une fonction appelée `demande_prenom_nom()` qui demande d'abord le prénom de l'utilisateur, puis son nom et renvoie comme résultat l'identité complète avec le nom en majuscule.

Par exemple, si l'utilisateur saisi « Dark » puis « Vador », la fonction renvoie la chaîne "Dark VADOR" (la fonction n'affiche rien).

Indications :

- Si `chaine` est une chaîne de caractères, alors `chaine.upper()` est la chaîne transformée avec les caractères en majuscules. Exemple : si `chaine = "Vador"` alors `chaine.upper()` renvoie `"VADOR"`.
- On peut fusionner deux chaînes en utilisant le signe « + ». Exemple : `"Dark" + "Vador"` vaut `"DarkVador"`. Autre exemple : si `chaine1 = "Dark"` et `chaine2 = "Vador"` alors `chaine1 + " " + chaine2` vaut `"Dark Vador"`.

Algorithme attendu

```
#
# DU II : TP n°3, exercice 3, algorithme
#

# (1) prototype (encore appelé signature) de la fonction
fonction demande_prenom_nom() -> chaîne

# (2) définition de la fonction
fonction demande_prenom_nom() -> chaîne
variables
    paramètres
        /
    internes
        chaîne prenom
        chaîne nom
    en sortie
        chaîne identite
début
    # initialisation
    écrire("Votre prénom ? ")
    lire(prenom)
    écrire("Votre nom ? ")
    lire(nom)
    # traitement
    # (3) appel de la fonction
    identite <- prenom + " " + nom.majuscules()
    # restitution des résultats
    renvoyer identite
fin

-----------------------------------------

programme principal
variables
    en entrée
        /
    en sortie
        /
début
    # (3) appel de la fonction
    demande_prenom_nom()
fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 3, Python
#

## Déclaration de la fonction demande_prenom_nom()
def demande_prenom_nom() -> str :
    prenom = input("Votre prénom ? ")
    nom = input("Votre nom ? ")
    identite = prenom + " " + nom.upper()
    return identite


##
## Début du programme principal
##

## Appel de la fonction affiche_bonjour()
print(f"{demande_prenom_nom()}")
```

Exercice 4 : Moyenne de 2 valeurs

1. Écrivez une fonction `moyenne()` qui prend en paramètres 2 valeurs entières et retourne la moyenne de ces 2 valeurs (le type de retour est un réel)
2. Écrivez un programme principal qui demande à l'utilisateur de saisir 2 valeurs entières, qui appelle la fonction précédente en passant en paramètres les valeurs de l'utilisateur et qui affiche le résultat de la fonction après l'avoir récupéré dans une variable

Algorithme attendu

```
#
# DU II : TP n°3, exercice 4, algorithme
#

# (1) prototype (encore appelé signature) de la fonction
fonction moyenne(a: entier,b: entier) -> réel

# (2) définition de la fonction
fonction moyenne(a: entier,b: entier) -> réel
variables
    paramètres
        entier a        # première valeur
        entier b        # seconde valeur
    en sortie
        réel moyenne    # valeur de la moyenne
début
    moyenne <- (a+b)/2
    retourne moyenne
fin

-----------------------------------------

programme principal
variables
    entree
        entier v1
        entier v2
    sortie
        réel v3
début
    # initialisation
    écrire("Entrez une valeur entière v1 : ")
    lire(v1)
    écrire("Entrez une valeur entière v2 : ")
    lire(v2)
    # traitement
    # (3) appel de la fonction
    v3 <- moyenne(v1,v2)
    # restitution des résultats
    écrire("La moyenne de {v1} et {v2} est égale à {v3}")
fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 4, Python
#

## Déclaration de la fonction moyenne()
def moyenne(a: int,b: int) -> float :
    moyenne = (a+b)/2
    return moyenne


##
## Début du programme principal
##

## Récupération des valeurs de l'utilisateur
print()
v1 = int(input("Entrez une valeur entière v1 : "))
print()
v2 = int(input("Entrez une valeur entière v2 : "))

## Appel de la fonction moyenne() et récupération de sa valeur de retour
v3 = moyenne(v1,v2)

## Affichage final
print()
print(f"La moyenne de {v1} et {v2} est égale à {v3}")
```

Exercice 5 : Trinômes

1. Écrivez une fonction `trinome_1(x)` qui dépend d'un paramètre `x` et qui renvoie la valeur du trinôme $3x^2 - 7x + 4$. Par exemple `trinome_1(7)` renvoie `102`.
2. Écrivez une fonction `trinome_2(a,b,c,x)` qui dépend de quatre paramètres `a`, `b`, `c` et `x` et qui renvoie la valeur du trinôme $ax^2 + bx + c$. Par exemple `trinome_2(2,-1,0,6)` renvoie `66`.

Algorithme attendu

```
#
# DU II : TP n°3, exercice 5, algorithme
#

# (1) prototype (encore appelé signature) des fonctions
fonction trinome_1(x: réel) -> réel
fonction trinome_2(a: réel,b: réel,c: réel,x: réel) -> réel

# (2) définition des fonctions
fonction trinome_1(x: réel) -> réel
variables
    paramètres
        réel x        # valeur de x
    sortie
        réel eval     # valeur de l'évaluation pour x
début
    eval <- (3*(x**2)) - (7*x) + 4
    retourne eval
fin
fonction trinome_2(a: réel,b: réel,c: réel,x: réel) -> réel
variables
    paramètres
        réel a,b,c    # valeur des coefficient a, b et c
        réel x        # valeur de x
    sortie
        réel eval     # valeur de l'évaluation pour a, b, c et x
début
    eval <- (a*(x**2)) + (b*x) + c
    retourne eval
fin

-----------------------------------------

programme principal
variables
    internes
        réel val1, val2
    en sortie
        réel res1, res2
début
    # initialisation
    val1 <- 7
    # traitement - (3) appel de la fonction
    res1 <- trinome_1(val1)
    # restitution des résultats
    écrire("L'évaluation du trinôme (+3,-7,+4) pour x={val1} est égale à {res1}")

    # initialisation
    val2 <- 6
    # traitement - (3) appel de la fonction
    res2 <- trinome_2(2,-1,0,val2)
    # restitution des résultats
    écrire("L'évaluation du trinôme (+2,-1,+0) pour x={val2} est égale à {res2}")
fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 5, Python
#

## Déclaration de la fonction trinome_1()
def trinome_1(x: float) -> float :
    eval = 3*x**2 - 7*x + 4
    return eval

## Déclaration de la fonction trinome_2()
def trinome_2(a: float,b: float,c: float,x: float) -> float :
    eval = a*x**2 + b*x + c
    return eval


##
## Début du programme principal
##

## appel des fonctions
val1 = 7
res1 = trinome_1(val1)
val2 = 6
res2 = trinome_2(2,-1,0,val2)

## Affichage final
print(f"L'évaluation du trinôme (+1,+2,+1) pour x={val1} est égale à {res1}")
print(f"L'évaluation du trinôme (+2,-1,+0) pour x={val2} est égale à {res2}")
```

Exercice 6 : Devises

1. Écrivez une fonction `conversion_euros_vers_dollars(montant)` qui dépend d'un paramètre et qui pour une somme d'argent `montant`, exprimée en euros, renvoie sa valeur en dollars (on prendra comme taux de change 1 euro = 1, 15 dollar).
2. Écrivez une fonction `conversion_euros_vers_dollars2(taux,montant)` qui dépend de deux paramètres et qui pour une somme d'argent `montant`, exprimée en euros, renvoie sa valeur en dollars selon la valeur `taux` du taux de change.

Algorithme attendu

```
#
# DU II : TP n°3, exercice 6, algorithme
#

# (1) prototype (encore appelé signature) des fonctions
fonction conversion_euros_vers_dollars(montant_e: réel) -> réel :

# (2) définition des fonctions
fonction conversion_euros_vers_dollars(montant_e: réel) -> réel :
variables
    paramètres
        réel montant_e
    sortie
        réel montant_d
début
    montant_d <- 1.15 * montant_e
    renvoyer montant_d
fin

-----------------------------------------

programme principal
variables
    entree
        réel montant_en_euro
    sortie
        réel montant_en_dollar
début
    # initialisation
    écrire("Entrez le montant en euros à convertir: ")
    lire(montant_en_euro)

    # traitement
    # (3) appel de la fonction
    montant_en_dollar <- conversion_euros_vers_dollars(montant_en_euro)

    # restitution des résultats
    écrire("{montant_en_euro} en € <-> {montant_en_dollar} en $")
fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 6, Python
#

## Déclaration de la fonction conversion_euros_vers_dollar()
def conversion_euros_vers_dollars(montant_e: float) -> float :
    montant_dollar = 1.15 * montant_e
    return montant_dollar


###
## Début du programme principal
###

print("Entrez le montant en euros à convertir: ")
montant_en_euro = float( input() )

montant_en_dollar = conversion_euros_vers_dollars(montant_en_euro)
print(f"{montant_en_euro} en euros <-> {montant_en_dollar} en dollars")
```

Exercice 7 : Périmètres et aires

1. Écrivez une fonction dont l'usage est `perimetre_rectangle(a,b)` et qui renvoie en sortie le périmètre d'un rectangle de dimensions a et `b`.
2. Écrivez une fonction dont l'usage est `aire_rectangle(a,b)` et qui renvoie en sortie l'aire d'un rectangle de dimensions a et `b`.
3. Même question avec `perimetre_disque(r)` pour le périmètre d'un disque de rayon `r`.
4. Même question avec `aire_disque(r)` pour l'aire d'un disque de rayon `r`.

Algorithme attendu

```
#
# DU II : TP n°3, exercice 7, algorithme
#

# (1) prototype (encore appelé signature) des fonctions
fonction perimetre_rectangle(a: réel,b: réel) -> réel
fonction perimetre_disque(r: réel) -> réel
fonction aire_rectangle(a: réel,b: réel) -> réel
fonction aire_disque(r: réel) -> réel

# (2) définition des fonctions
fonction perimetre_rectangle(a: réel,b: réel) -> réel
    perimetre <- 2*a + 2*b
    renvoyer perimetre
fonction perimetre_disque(r: réel) -> réel
    perimetre <- 2*math.PI*r          # 2.PI.r
    renvoyer perimetre
fonction aire_rectangle(a: réel,b: réel) -> réel
    aire <- a*b
    renvoyer aire
fonction aire_disque(r: réel) -> réel
    aire <- math.PI * r**2            # PI.r^2
    renvoyer aire

-----------------------------------------

programme principal
variables
    /
début
    # traitement et résultat
    # (3) appel de la fonction
    écrire("perimetre_rectangle(3,4): {perimetre_rectangle(3,4)}")
    écrire("aire_rectangle(3,4): {aire_rectangle(3,4)}")
    écrire("perimetre_disque(5): {perimetre_disque(5)}")
    écrire("aire_disque(5): {aire_disque(5)}")
fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 7, Python
#

## Déclaration de la fonction perimetre_rectangle()
def perimetre_rectangle(a: float,b: float) -> float :
    perimetre = 2*a + 2*b
    return perimetre

## Déclaration de la fonction perimetre_disque()
def perimetre_disque(r: float) -> float :
    perimetre = 2*math.pi*r
    return perimetre

## Déclaration de la fonction aire_rectangle()
def aire_rectangle(a: float,b: float) -> float :
    aire = a*b
    return aire

## Déclaration de la fonction aire_disque()
def aire_disque(r: float) -> float :
    aire = math.pi * r**2
    return aire

##
## Début du programme principal
##

print(f"perimetre_rectangle(3,4): {perimetre_rectangle(3,4)}")
print(f"aire_rectangle(3,4): {aire_rectangle(3,4)}")
print(f"perimetre_disque(5): {perimetre_disque(5)}")
print(f"aire_disque(5): {aire_disque(5)}")
```

Deuxième partie

Pour aller plus loin

Exercice 8 : Égalité expérimentale

1. Écrivez une fonction à deux paramètres `F1(a,b)` qui renvoie $(a+b)^2$.
2. Même chose avec `F2(a,b)` qui renvoie $a^2+2ab+b^2$.
3. On dit que deux fonctions de deux variables F et G sont expérimentalement égales si $F(i,j) = G(i,j)$ pour tout i $= -100, -99, \ldots, 100$ et pour tout j $= -100, -99, \ldots, 100$. Vérifiez que les fonctions définies par $(a+b)^2$ et $a^2+2ab+b^2$ sont expérimentalement égales en testant comme valeurs pour les i et les j égaux à : -100, -75, -50, -25, 0, 25, 50,75,100.

Algorithme attendu

```
1  #
2  # DU II : TP n°3, exercice 8, algorithme
3  #
4
5  # (1) prototype (encore appelé signature) des fonctions
6  fonction f1(a: réel,b: réel) -> réel
7  fonction f2(a: réel,b: réel) -> réel
8
9  # (2) définition des fonctions
10 fonction f1(a: réel,b: réel) -> réel
11     renvoyer (a+b)**2
12 fonction f2(a: réel,b: réel) -> réel
13     renvoyer a**2 + 2*a*b + b**2
14
15 ----------------------------------------
16
17 programme principal
18 variables
19     /
20 début
21     # traitement et résultat
22     # (3) appel de la fonction
23     écrire("f1(3,4): {f1(3,4)}")
24     écrire("f2(3,4): {f2(3,4)}")
25
26     # traitement et résultat
27     # (3) appel de la fonction
28     écrire("-100: {f1(-100,-100)} / {f2(-100,-100)}")
29     écrire("-75: {f1(-75,-75)} / {f2(-75,-75)}")
30     écrire("-50: {f1(-50,-50)} / {f2(-50,-50)}")
31     écrire("-25: {f1(-25,-25)} / {f2(-25,-25)}")
32     écrire("0: {f1(0,0)} / {f2(0,0)}")
33     écrire("25: {f1(25,25)} / {f2(25,25)}")
34     écrire("50: {f1(50,50)} / {f2(50,50)}")
35     écrire("75: {f1(75,75)} / {f2(75,75)}")
36     écrire("100: {f1(100,100)} / {f2(100,100)}")
37 fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 8, Python
#

## Déclaration de la fonction f1()
def f1(a: float,b: float) -> float :
    return (a+b)**2

## Déclaration de la fonction f2()
def f2(a: float,b: float) -> float :
    return a**2 + 2*a*b + b**2


##
## Début du programme principal
##

print(f"f1(3,4): {f1(3,4)}")
print(f"f2(3,4): {f2(3,4)}")

print(f"-100: {f1(-100,-100)} / {f2(-100,-100)}")
print(f"-75: {f1(-75,-75)} / {f2(-75,-75)}")
print(f"-50: {f1(-50,-50)} / {f2(-50,-50)}")
print(f"-25: {f1(-25,-25)} / {f2(-25,-25)}")
print(f"0: {f1(0,0)} / {f2(0,0)}")
print(f"25: {f1(25,25)} / {f2(25,25)}")
print(f"50: {f1(50,50)} / {f2(50,50)}")
print(f"75: {f1(75,75)} / {f2(75,75)}")
print(f"100: {f1(100,100)} / {f2(100,100)}")
```

Exercice 9 : Volumes

Écrivez des fonctions qui calculent et renvoient des volumes :
— le volume d'un cube en fonction de la longueur d'un côté,
— le volume d'une boule en fonction de son rayon,
— le volume d'un cylindre en fonction du rayon de sa base et de sa hauteur,
— le volume d'une boîte parallélépipède rectangle en fonction de ses trois dimensions.

Pour la valeur de π, vous prendrez soit la valeur approchée 3.14, soit la valeur approchée fournie par la constante pi du module `math`.

Algorithme attendu

```
#
# DU II : TP n°3, exercice 9, algorithme
#

# (1) prototype (encore appelé signature) des fonctions
fonction volume_cube(a: réel) -> réel
fonction volume_sphere(r: réel) -> réel
fonction volume_cylindre(r: réel,h: réel) -> réel
fonction volume_boite(a: réel,b: réel,c: réel) -> réel

# (2) définition des fonctions
fonction volume_cube(a: réel) -> réel
    renvoyer a**3
fonction volume_sphere(r: réel) -> réel
    renvoyer 4/3 * math.PI * r**3
fonction volume_cylindre(r: réel,h: réel) -> réel
    renvoyer math.PI * r**2 * h
fonction volume_boite(a: réel,b: réel,c: réel) -> réel
    renvoyer a * b * c

-----------------------------------------

programme principal
variables
    /
début
    # traitement et résultat
    # (3) appel de la fonction
    écrire("volume_cube(5): {volume_cube(5)}")
    écrire("volume_sphere(5): {volume_sphere(5)}")
    écrire("volume_cylindre(5,5): {volume_cylindre(5,5)}")
    écrire("volume_boite(5,5,5): {volume_boite(5,5,5)}")
fin
```

Code Python attendu

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# DU II : TP n°3, exercice 9, Python
#

## Déclaration de la fonction volume_cube()
def volume_cube(a: float) -> float :
    return a**3

## Déclaration de la fonction volume_sphere()
def volume_sphere(r: float) -> float :
    return 4/3 * math.pi * r**3

## Déclaration de la fonction volume_cylindre()
def volume_cylindre(r: float,h: float) -> float :
    return math.pi * r**2 * h

## Déclaration de la fonction volume_boite()
def volume_boite(a: float,b: float,c: float) -> float :
    return a * b * c


##
## Début du programme principal
##

print(f"volume_cube(5): {volume_cube(5)}")
print(f"volume_sphere(5): {volume_sphere(5)}")
print(f"volume_cylindre(5,5): {volume_cylindre(5,5)}")
print(f"volume_boite(5,5,5): {volume_boite(5,5,5)}")
```