ACTIVITÉ 5: PROCÉDURES ET FONCTIONS EN PYTHON

Problématique: Comment organiser un programme en blocs d'instructions réutilisables et récupérer les résultats d'un calcul ?

Objectifs: *À la fin de l'activité, vous serez capable de :*

- *Comprendre l'intérêt d'une procédure ou d'une fonction ;*
- *Définir et appeler une procédure avec des paramètres ;*
- *Distinguer une procédure d'une fonction qui renvoie une valeur ;*
- *Utiliser l'instruction return ;*
- *Créer et utiliser des fonctions dans un programme Python*

Dans un programme, il arrive que l'on doive exécuter plusieurs fois les mêmes instructions. Pour éviter de les recopier, on peut les regrouper dans un **bloc d'instructions réutilisable**. En Python, on utilise le mot-clé `def` pour définir ce bloc. Il peut recevoir des informations appelées **paramètres** et éventuellement renvoyer un résultat.

Nous allons découvrir comment utiliser ces blocs pour simplifier nos programmes.

PARTIE 1 : DÉCOUVRIR LES PROCÉDURES

A. Pourquoi utiliser une procédure ?

On considère le programme Python suivant :

```
print("Lundi est un jour de la semaine. Bonne journée !")
print("Mardi est un jour de la semaine. Bonne journée !")
print("Mercredi est un jour de la semaine. Bonne journée !")
print("Jeudi est un jour de la semaine. Bonne journée !")
print("Vendredi est un jour de la semaine. Bonne journée !")
print("Samedi est un jour de la semaine. Bonne journée !")
print("Dimanche est un jour de la semaine. Bonne journée !")
```

1. Combien d'instructions contient ce programme ? 7
2. Quelle partie des instructions est répétée ? print("… est un jour de la semaine. Bonne journée !")
3. Comment pourrait-on organiser ce programme pour éviter de recopier plusieurs fois les mêmes instructions ? Utiliser une variable pour le jour de la semaine

   On propose maintenant le programme suivant :

```
def phrase(jour):
    print(jour, "est un jour de la semaine. Bonne journée !")

phrase("Lundi")
phrase("Mardi")
phrase("Mercredi")
phrase("Jeudi")
phrase("Vendredi")
phrase("Samedi")
phrase("Dimanche")
```

4. *Partie sur ordinateur:* Saisir le programme dans Trinket. L'exécuter et vérifier l'affichage obtenu.
5. Quel est le rôle de la variable jour ? Elle contient le caractère du jour de la semaine
6. Que se passe-t-il lorsqu'on exécute `phrase("Lundi")` ? L'affichage est print("Lundi est un jour de la semaine. Bonne journée !")

7. Expliquer l'intérêt d'utiliser une procédure..... Minimiser le nombre de lignes répétées dans le programme.....

A RETENIR: LES PROCÉDURES

Une procédure est un bloc d'instructions qui peut être exécuté plusieurs fois dans un programme.
Une procédure peut recevoir des informations grâce à des paramètres, dont les valeurs sont précisées lors de son appel. En Python, une procédure se définit avec le mot-clé `def`.

```
def nom_procedure(parametre1, parametre2):
    instruction1
    instruction2
```

Pour exécuter une procédure, il faut l'appeler en indiquant les arguments nécessaires.
Attention : les instructions appartenant à la procédure doivent être indentées (décalées vers la droite).

B. Utiliser une procédure avec plusieurs paramètres

On considère le programme Python suivant :

```
print("Bonjour Alice")
print("Quel temps fait-il à Paris ?")

print("Bonjour Thomas")
print("Quel temps fait-il à Rome ?")

print("Bonjour Camille")
print("Quel temps fait-il à Londres ?")
```

On souhaite remplacer ces instructions par une procédure `salutations(prenom, ville)`.

1. Compléter le programme :

```
def salutations(prenom, ville):
    print("Bonjour ", prenom)
    print("Quel temps fait-il à ", ville)

salutations("Alice", "Paris")
salutations("Thomas", "Rome")
salutations("Camille", "Londres")
```

2. *Partie sur ordinateur*
   a. Saisir le programme dans Trinket. L'exécuter et vérifier l'affichage obtenu.
   b. Ajouter un appel permettant d'afficher le contenu ci-dessous lors de l'exécution.

```
Bonjour Lucas
Quel temps fait-il à Milan ?
```

3. Modifier la procédure pour qu'elle affiche également « Bonne journée ! «

PARTIE 2 : DÉCOUVRIR LES FONCTIONS

On souhaite calculer le périmètre d'un rectangle de longueur L et de largeur l.
On considère les deux programmes suivants:

```
#FONCTIONS DU PROGRAMME A
def perimetre1(L, l):
    p = 2*(L+l)
    print(p)

perimetre1(5, 3)
```

```
#FONCTIONS DU PROGRAMME B
def perimetre2(L, l):
    p = 2*(L+l)
    return p

print(perimetre2(5, 3))
```

À RETENIR : LES COMMENTAIRES EN PYTHON: En Python, le symbole # permet d'ajouter un commentaire pour expliquer le programme. Les commentaires ne sont pas exécutés.

1.

a. Quelle formule mathématique est utilisée dans les deux programmes ?

Perimetre = (Longueur + Largeur) x 2

b. Quel résultat doit-on obtenir pour *L=5* et *l=3* ?

(5+3)x2 = 8x2 = 16

c. Quelle instruction est différente entre les deux programmes ?

print(p) dans le prog A ; return p dans le prog B

`#PROGRAMME A`
`resultat = perimetre1(5, 3)`
`print(resultat)`
	`#PROGRAMME B`
	`resultat = perimetre2(5, 3)`
	`print(resultat)`

2. On ajoute maintenant les instructions ci-dessus :

a. Exécuter les deux programmes dans Trinket. Que constatez-vous ? L'affichage est identique

b. Quelle est la différence entre les instructions `print` et `return` ? print affiche le perimetre ; return envoie le perimetre dans une variable

c. Quel programme permet de récupérer le résultat du calcul pour le réutiliser ? B

A RETENIR: LES FONCTIONS

Une fonction est un bloc d'instructions qui peut recevoir des paramètres et qui renvoie une valeur grâce à l'instruction `return`.
En Python, une fonction se définit ainsi :

```
def nom_fonction(parametre1, parametre2):
    instruction1
    instruction2
    return resultat
```

Contrairement à une procédure qui se contente d'exécuter des instructions, une fonction permet de récupérer un résultat et de le réutiliser dans un programme.
L'instruction `print()` affiche une information à l'écran, tandis que `return` renvoie une valeur au programme qui a appelé la fonction.

PARTIE 3 : CRÉER ET UTILISER DES FONCTIONS

A. Calculer une moyenne

On souhaite calculer la moyenne de trois notes.
On considère l'algorithme suivant et sa traduction en Python.

FONCTION moyenne(a,b,c):
m ← (a+b+c)/3
RETOURNER m
	`def moyenne(a,b,c):`
	`m = (a+b+c)/3`
	`return m`

1. Calculer la moyenne des notes 8, 12 et 19: (8+12+19)/3 = 39/3 = 13
2. Compléter la traduction de l'algorithme en Python.
3. *Partie sur ordinateur :*
   a. Saisir la fonction dans Trinket.
   b. Ajouter l'instruction `print(moyenne(8,12,19))`.
   c. Modifier l'instruction précédente pour afficher la moyenne arrondie à deux chiffres après la virgule.
   d. Vérifier le résultat obtenu.
   e. Utiliser la fonction pour calculer la moyenne de 14, 12 et 13. Résultat: 13
4. Modifier la fonction pour qu'elle calcule la moyenne de quatre notes.

APPEL → **Appeler le professeur pour vérification**

B. Convertir des températures

La relation entre une température exprimée en degrés Fahrenheit (F) et sa valeur en degrés Celsius (C) est:

$$C = \frac{5F - 160}{9}$$

On souhaite créer une fonction permettant d'effectuer cette conversion.

1. Compléter la fonction suivante :

```
def celsius(fahrenheit):
    C = (5*fahrenheit-160)/9
    return C

print(celsius(100))
print(celsius(200))
```

2. *Partie sur ordinateur :*
   a. Saisir et compléter le programme dans Trinket. Exécuter le programme et relever les deux températures obtenues. 100°F ≈ 38°C 200°F ≈ 93°
   b. Modifier le programme pour afficher les températures arrondies à deux chiffres après la virgule.
   c. Ajouter un appel à la fonction permettant de convertir 32 °F en degrés Celsius.

APPEL → **Appeler le professeur pour vérification**

PARTIE 4 : RÉINVESTIR LES FONCTIONS MATHÉMATIQUES

A. Calculer l'hypoténuse d'un triangle rectangle

On considère un triangle rectangle dont les deux côtés de l'angle droit mesurent *a* et *b*.

D'après le théorème de Pythagore, la longueur de l'hypoténuse est : h = $\sqrt{a^2 + b^2}$
On souhaite créer une fonction hypotenuse(a,b) qui renvoie cette longueur.

1. Calculer la longueur de l'hypoténuse d'un triangle rectangle dont les côtés mesurent 3 et 4.

   $h = \sqrt{3^2 + 4^2} = \sqrt{9+16} = \sqrt{25} = 5$

2. Compléter l'algorithme et sa traduction en Python.

FONCTION hypotenuse(a,b):
h ← $\sqrt{a^2 + b^2}$
RETOURNER h
	`from math import *`
	`def hypotenuse(a,b):`
	`  h = sqrt(a**2+b**2)`
	`  return h`

3. *Partie sur ordinateur*
   a. Compléter et saisir le programme dans Trinket.
   b. Calculer l'hypoténuse d'un triangle rectangle dont les côtés mesurent 3 et 4.
   c. Vérifier que le résultat obtenu est cohérent.
   d. Calculer ensuite les hypoténuses des triangles dont les côtés mesurent respectivement 5 et 12, puis 8 et 15.
4. *Problème*: On considère trois triangles rectangles dont les côtés de l'angle droit mesurent :

Triangle	**1**	**2**	**3**
Premier côté	*3x*	*2x*	*x*
Deuxième côté	*x+3*	*x+2*	*x+1*

Écrire un programme qui demande une valeur strictement positive de x, puis affiche la somme des trois hypoténuses en utilisant la fonction hypotenuse().

```
from maths import *
def hypothenuse(a,b)
    h = sqrt( a**2 + b**2)
    return h

x = float(input(" valeur de x strictement positive "))
h1 = hypothenuse(3*x, x+1)
h2 = hypothenuse(2*x, x+2)
h3 = hypothenuse(3*x, x+3)
somme = h1 + h2 + h3
Print(somme)
```

APPEL → **Appeler le professeur pour vérification**

BILAN DE L'ACTIVITÉ 1.5 : PROCÉDURES ET FONCTIONS EN PYTHON

En Python, on peut regrouper plusieurs instructions dans un **bloc réutilisable**, défini avec le mot-clé `def`.

Ce bloc peut recevoir des informations appelées **paramètres**.

On distingue deux utilisations :

	Procédure	**Fonction**
Rôle	Exécuter des instructions	Calculer et renvoyer un résultat
Définition	`def nom(parametres):`	`def nom(parametres):`
Instruction caractéristique	`print()` (par exemple)	`return`
Résultat	Peut afficher une information	Renvoie une valeur réutilisable

À RETENIR

- `def` permet de définir une procédure ou une fonction.
- Les **paramètres** permettent de transmettre des informations au bloc.
- Les instructions du bloc doivent être **indentées**.
- `print()` permet d'afficher une information.
- `return` permet de renvoyer une valeur qui pourra être réutilisée dans le programme.
- Une procédure ou une fonction peut être appelée plusieurs fois avec des arguments différents.