AP02 LES FONCTIONS

I. La gestion des entrées - sorties

Gestion des entrées en Python

Il est possible en python de demander à l'utilisateur d'entrer une valeur au clavier. Cela s'effectue via la commande `input` La valeur tapée par l'utilisateur est convertie au format chaine de caractère (str)

Code Python

```
a = input("Entrer une valeur au clavier")          # a sera de type str
b = int(input("Entrer une valeur au clavier"))     # b sera de type int
```

Méthode 1 : Affichage dans la zone programme

Code Python

```
def f(x):
    y = x ** 2  # Fonction calculant x**2
    return y

f(4)            # PAS D'AFFICHAGE: f(4) est créé mais est perdue
print(f(4))     # AFFICHAGE: f(4) créé puis affiché. La valeur est perdue
carre = f(6)    # PAS D'AFFICHAGE: f(6) créé puis sauvegardé dans carre.
print(carre)    # AFFICHAGE: affichage de la valeur de la variable carre
```

On peut afficher des chaines complexes avec le séparateur virgule ,

Code Python

```
a, b = 6, 4
print("La somme de", a, "et",b, "est", a + b)
```

Méthode 2 : Affichage dans la console python

La console python accepte en plus de la commande `print` un affichage direct en appelant la variable.

```
>>> a = 6
>>> a
6
```

Méthode 3 : Formatage de l'affichage avec `f-string` et `format()`

L'idée est de créer un texte à trou que l'on va compléter par des variables.

```
>>> name = 'Paul'
>>> age = 25
>>> print(f'Votre nom est {name} et vous avez {age} ans.')
Votre nom est Paul et vous avez 25 ans.
```

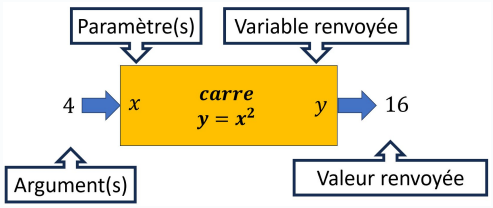
```
>>> print('Votre nom est {} et vous avez {} ans'.format(name, age))
Votre nom est Paul et vous avez 25 ans
```

II. Les fonctions

Présentation

Une **fonction** en python est un objet qui se comporterait comme une fonction mathématique. On peut prendre pour exemple une fonction f définie par $f(x) = x^2$.

- f est le nom de la fonction
- x est un paramètre (une variable en mathématique). C'est juste un concept dont on va se servir pour appliquer la fonction. Ce nom de variable n'est connu qu'à l'intérieur de la fonction.
- x^2 est l'expression de la fonction. C'est la manière dont on va calculer l'image de x par la fonction f.



- Ici la valeur 4 est un argument, c'est à dire la valeur sur laquelle on va appliquer la fonction.
- La valeur 16 est la valeur renvoyée par la fonction. Ne pas confondre `variable renvoyée` et `valeur renvoyée`

Code Python

```
1 def f(x):
2     """ Fonction renvoyant x ** 2 """
3     y = x ** 2          # Calcul de x ** 2
4     return y            # valeur renvoyée: y
```

- **Ligne 1 :** `def` permet de définir le nom de la fonction. Ici la fonction se nomme `f`.
- **Ligne 1 :** `f(x)` permet de définir le nom et le nombre des paramètres. Ici la fonction `f` a besoin d'un seul paramètre `x`
- **Ligne 2 :** commentaire permettant de décrire ce que fait la fonction. Cette ligne n'est pas interprétée
- **Ligne 3 :** on lit de la droite vers la gauche : « on prend le contenu de la variable `x`, on le met au carré et on l'affecte à une nouvelle variable appelée `y` »
- **Ligne 4 :** on renvoie la valeur de la variable `y`.

On vient de définir une nouvelle fonction. Il faut maintenant lui donner un argument pour qu'elle puisse calculer une valeur et la renvoyer. C'est ce que l'on appelle **appeler** la fonction.

Code Python

```
a = 4               # on affecte 4 à la variable a
z = f(4)            # on appelle la fonction f avec comme argument 4
                    # et on récupère la valeur renvoyée dans la variable z
```

- la commande `return` renvoie la valeur contenue dans y
- elle provoque la sortie immédiate de la fonction
- si possible on ne met qu'un seul `return` dans la fonction
- il peut n'y avoir aucun `return`, on parle alors de `procédure`

Prototypage d'une fonction

Code Python

```
1 def ma_fonction(param1, param2, ...):
2     ''' Docstring: Explication de ce que fait ma_fonction
3     : description des paramètres et leurs types
4     : description des valeurs de sortie et leurs types '''
5     # Corps de la fonction
6
7     return valeur     # La variable 'valeur' est renvoyée
```

Exemple

Voici un exemple d'une fonction qui fait quelque chose de très simple : elle prend deux valeurs en paramètres (`a` et `b`) et renvoie la somme des ces deux valeurs. Cette somme a été créée et affectée à la variable `somme`.

Code Python

```python
def calc_somme(a, b):
    ''' Fonction calculant la somme de deux valeurs
    : a et b deux entiers (int)
    : valeur renvoyée est un entier (int) '''
    somme = a + b       # le résultats de a + b est affecté à la variable somme
    return somme        # la valeur de somme est renvoyée par la fonction
```

Il est maintenant possible de se servir de cette fonction.

Code Python

```python
s = calc_somme(6, 8)     # s récupère la valeur renvoyée par calc_somme(6, 8)
print(s)                 # la valeur de la variable s est affichée (14)
```

Remarque : Variables locales et variables globales

- les **variables locales** ne sont connues que dans la fonction
- les **variables globales** sont connues dans tout le programme

Code Python

```python
print(a)    # ERREUR -> a est une variable locale (tout comme b et somme)
```

Il est donc tout à fait normal que les variables `a`, `b` et `somme` ne soient pas connues à l'extérieur de la fonction. Il est de bon usage de ne pas appeler de la même façon des variables locales et des variables globales.

Vérification de la valeur d'une variable

La fonctionnalité `assert` vérifie qu'une affirmation est vraie. Si c'est le cas, rien ne se passe, sinon un message d'erreur est donné

Code Python

```python
assert calcule_somme(6, 8) == 14          # True -> aucune message
assert calcule_somme(6, 8) == 12          # False -> message d'erreur
assert not(calcule_somme(6, 8) == 12)     # True -> aucun message
```