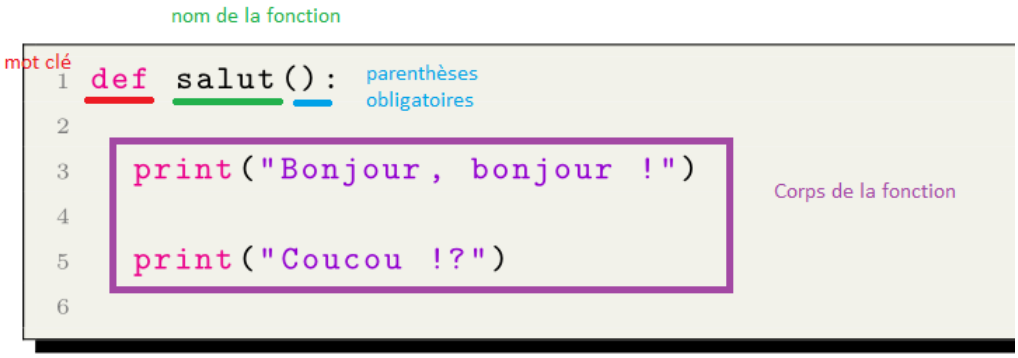
TP3 - Fonction

I Définir une fonction

En informatique, une **fonction** est un sous-programme.
Pour définir une fonction, on utilise le mot clé **def** suivi du nom de la fonction.

Exemple :



Remarque : Si on exécute ce code, rien ne s'affiche car on a juste défini la fonction `salut()`.

II Appeler une fonction

Pour exécuter une fonction, on doit l'appeler.

Exemple :

```
1  # on définit la fonction salut1()
2  def salut1():
3    print("Bonjour, bonjour !")
4    print("Coucou !?")
5
6  #on définit la fonction salut2()
7  def salut2():
8    print("Salut toi !")
9
10 #on appelle la fonction salut2()
11 salut2()
12 #on appelle la fonction salut1()
13 salut1()
```

Lorsqu'on exécute le code ci-dessus :

1. l'ordinateur lit la ligne 2 et stocke dans sa mémoire l'emplacement du début de la fonction `salut1()` ;
2. l'ordinateur lit la ligne 7 et stocke dans sa mémoire l'emplacement du début de la fonction `salut2()` ;
3. l'ordinateur lit la ligne 11 qui est un appel de fonction.
4. l'ordinateur va en ligne 7 (début de la fonction), puis exécute la ligne 8.
   Dans la console, il s'affiche : Salut toi !

5. l'ordinateur va en ligne 13 qui est un appel de fonction.
6. l'ordinateur va en ligne 2 (début de la fonction), puis exécute la ligne 3.
   Dans la console, il s'affiche : Bonjour, bonjour !
7. l'ordinateur exécute la ligne 4.
   Dans la console, il s'affiche : Coucou !?

III Les paramètres d'une fonction

Une fonction peut avoir 0, 1 ou plusieurs paramètres, séparés par des virgules.
On appelle la fonction en remplaçant chaque paramètre par un argument (c'est-à-dire une valeur).

Exemple :

```
1  # fonction avec 2 paramètres : nom et prenom
2  def salut(nom, prenom) :
3    print("Bonjour", prenom, nom, "!")
4
5
6  #2 appels de la fonction
7  salut("Durand", "Paul")
8  salut("Duval", "Alice")
```

Sortie
Bonjour Paul Durand !
Bonjour Alice Duval !

IV Valeurs renvoyées

Une fonction peut renvoyer des valeurs.
Pour cela, on utilise le mot clé `return` suivi de la valeur.
Lorsqu'on appelle une fonction renvoyant des valeurs, on stocke ces dernières dans des variables.

Exemple :

```
1  def perimetre_rectangle(longueur, largeur) :
2    p = 2 * longueur + 2 * largeur
3    return p
4
5  peri = perimetre_rectangle(5, 2) #le résultat est stocké dans peri
```

Q1. Sans utiliser d'ordinateur, indiquer ce qu'il s'affiche.

```
1  def f1(a, b, c):
2      r = a + b * c
3      return r
4  def f2(a, b):
5      return 2 * a + b
6
7  d = f1(1, 2, 3)
8  print(2 * d)
9  print(f2(0, d))
10 print(f1(d, d-1, d+1))
```

Remarque : On peut indiquer le type des paramètres et des valeurs renvoyées.

```
def perimetre_rectangle(longueur : float, largeur : float) -> float :
  p = 2 * longueur + 2 * largeur
  return p
```

Exemple : Une fonction peut renvoyer plusieurs valeurs.

```
def milieu_segment(xA , yA , xB, yB) :
    """ renvoie les coodonnees du milieu d'un segment"""
    xI = (xA + xB) / 2
    yI = (yA + yB) / 2
    return xI, yI

x, y = milieu_segment(1, 2, 5, 6) # on stocke le resultat dans les variables x et y
print("abscisse :", x)
print("ordonnee :", y)
```

Remarque : Lorsque l'ordinateur exécute l'instruction `return`, le programme sort de la fonction.

V Exercices

Q2. Ecrire une fonction `bonne_annee() -> None` qui affiche le décompte de 10 à 1, puis "Bonne année !"
Cette fonction ne renvoie rien.

Q3. Ecrire une fonction `somme_des_carres(n : int) -> int` qui renvoie la somme $\sum_{k=0}^{n} k^2$. On utilisera obligatoirement une boucle `for`.
Ecrire l'instruction qui calcule la somme $\sum_{k=0}^{10} k^2$? (On doit trouver 385).

Q4.

1. Ecrire une fonction `factorielle(n:int) -> int` avec $n \in \mathbb{N}$ qui renvoit $n!$.
2. Faire afficher 0!, 1!, ..., 10!. ($10! = 3\,628\,800$). On utilisera une boucle `for`.
3. Ecrire une fonction `combinaison(k : int, n : int) -> int` avec $k \in [\![0; n]\!]$, qui renvoie $\binom{n}{k}$.
4. Afficher $\binom{8}{6}$. ($\binom{8}{6}$ doit valoir 28)
5. Calculer $\sum_{k=0}^{5} \binom{5}{k} 3^k$. (On doit trouver $(3+1)^5 = 1024$)

Q5. Un nombre entier supérieur ou égal à 2 est dit premier si et seulement si il n'est divisible que par 1 et par lui-même.

1. Ecrire une fonction `est_premier(n : int) -> bool` qui renvoie `True` si `n` est premier et `False` sinon.
2. Afficher tous les nombres premiers inférieurs ou égaux à 100.
3. Combien y a-t-il de nombres premiers inférieurs ou égaux à 100 ? On doit trouver 25.
4. Ecrire une fonction `prochain_premier(n : int) -> int` qui renvoie le plus petit nombre premier strictement supérieur à `n`.

   Les nombres suivants sont premiers : 101, 103, 107, 109, 113, 127. Tester la fonction `prochain_premier(n)`.

5. La conjecture de Goldbach affirme que tout nombre pair supérieur à 3 est somme de deux nombres premiers.

   Afficher tous les nombres pairs inférieurs ou égaux à 100 comme somme de deux entiers premiers.

Sortie

$4 = 2 + 2$
$6 = 3 + 3$
$8 = 3 + 5$
...
$98 = 19 + 79$
$100 = 3 + 97$