Nom :	Prénom :	Classe :

SNT — Fonction : du prototypage à la création et l'utilisation

Objectifs :
- Savoir transcrire un prototypage de fonction en code Python.
- Savoir créer (définir) sa propre fonction en Python, à partir d'un descriptif donné.
- Savoir utiliser (appeler) une fonction dans différents contextes.

Introduction

Nous avons vu précédemment comment réaliser le schéma (le prototype) d'une fonction[1], découvrant par la même occasion qu'une fonction prenait **un** ou **plusieurs paramètres d'entrée**. Nous avons également remarqué qu'une fonction retournait une **valeur de sortie**.

Pour résumer ce que nous savons, nous pouvons dire qu'une fonction est une sorte de *machine*.
- Une *machine* qui accepte une ou plusieurs informations en entrée (paramètres d'entrée).
- Une *machine* qui modifie ces informations.
- Une *machine* d'où ressort une information (valeur de sortie).

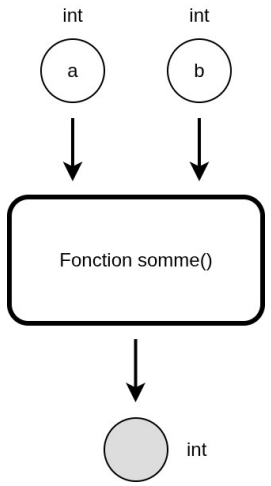
Pour rappel, il existe des *fonctions préexistantes* dans Python — des fonctions comme par exemple `print()`, `range()` ou `input()` — mais le cours que vous tenez entre les mains porte sur la possibilité de créer ses *propres fonctions personnelles*, puis de les utiliser dans différents contextes.

Découverte

Sans plus tarder, prenons un premier exemple de création de fonction, dont le nom est `somme()` :

- Cette fonction n'est pas une fonction préexistante, nous sommes en train de la créer.
- Cette fonction est d'abord schématisée (cf colonne "Prototypage" ci-dessous).
- Cette fonction est ensuite transcrite en langage Python (cf colonne "Code Python" ci-dessous).

Prototypage	Code Python

	`def somme(a, b):`
	`    resultat = a + b`
	`    return resultat`

1 Voir le cours « Fonction : appel et prototypage ».

➤ **Exercice 1** — Prédire

En observant page précédente le prototypage et le code de la fonction `somme()`, à votre avis, que va-t-il se passer lors des trois appels de fonction suivants :

N°	Code Python	Que va-t-il se passer ? (à compléter)
1	`print( somme( 10, 40 ) )`	…
2	`age = somme(10, 8)` `print(age)`	…

3	`objet_1 = 160`
	`objet_2 = 200`
	`total = somme(objet_1, objet_2)`
	`print(total)`
		…

➤ **Exercice 2** — Exécuter

Vérifions ensemble votre prédiction, en regroupant la création de la fonction `somme()` ainsi que les trois appels dans un même code Python qui ne nous reste plus qu'à **exécuter** :

Lien

Alors, votre prédiction était-elle bonne ?

Structure d'une fonction en Python

La structure d'une fonction est assez proche des structures déjà étudiées : condition, boucle, etc.

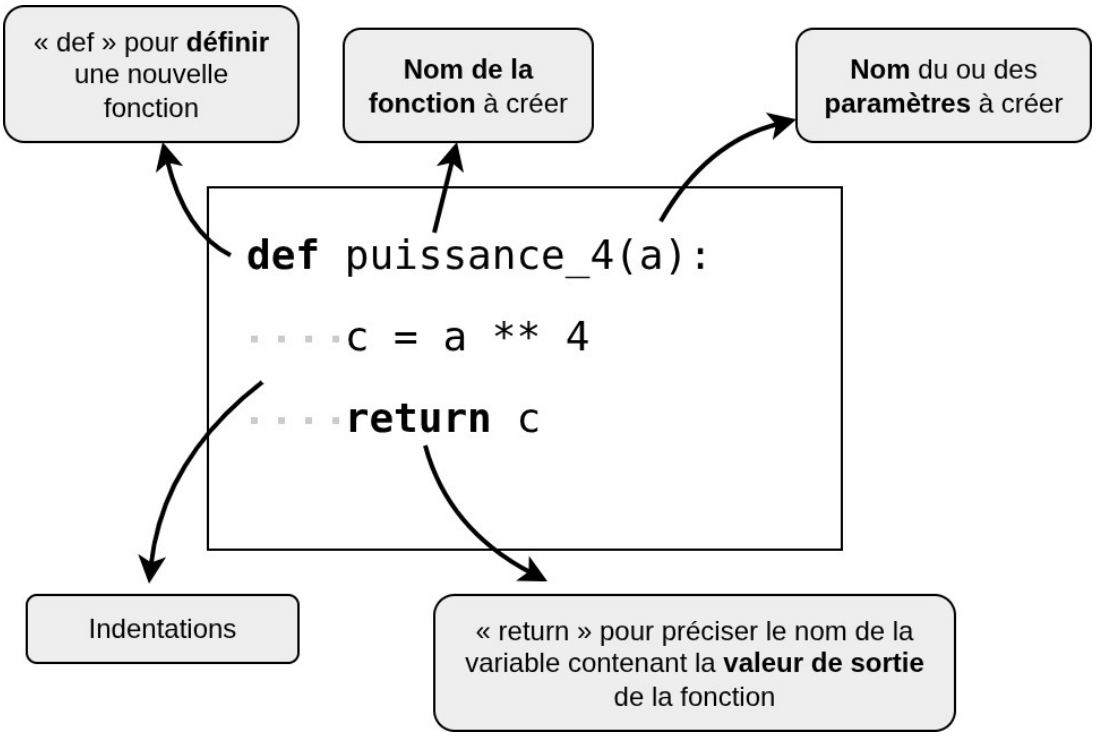
Prenons par exemple le code d'une nouvelle fonction appelée « puissance_4 » :

```
1  def puissance_4(a):
2      c = a ** 4
3      return c
```

Que pouvons-nous *dire* de ce *code* ?

- La création (on peut aussi dire *la définition*) d'une nouvelle fonction
  passe par l'utilisation du **mot-clé** `def` suivi du **nom de la fonction** à créer.
  Ici le nom de la fonction est « puissance_4 »
- Le nom de la fonction est toujours suivi de **parenthèses** contenant le ou les noms des **paramètres d'entrée**. *Ici le nom du paramètre d'entrée est « a »*
- Après les parenthèses, on ajoute toujours le **symbole deux points** `:`
  - Puis, **tout le code** qui se situe **dans la fonction** est **indenté**,
    c'est à dire décalé de quatre espaces vers la droite.
  - **Tout le code contenu dans la fonction** forme un **bloc de code** (ou *bloc d'instructions*).
  - Le bloc de code **se termine** généralement par une instruction qui permet de faire **sortir une valeur** cette sortie est rendue possible par l'utilisation du **mot-clé** `return` suivi du **nom de la variable** contenant la **valeur de sortie**.

Ce qui pourrait se résumer par l'illustration suivante :



➤ **Exercice 3** — Enquêter

Lisez le code des différentes *nouvelles fonctions* suivantes, puis inventoriez-en le **nom de la fonction**, le **nom du paramètre**, et le **rôle présumé de la fonction** en vous posant la question suivante : *à quoi sert cette fonction, et à quoi correspond sa valeur de sortie ?*

Code n°1

```
1  def aire_carre(longueur_cote):
2      resultat = longueur_cote * longueur_cote
3      return resultat
```

Code n°2

```
1  def prix_total(nbre):
2      prix_place = 11.5
3      r = nbre * prix_place
4      return r
```

Code n°3

```
1  def majorite(age):
2      if age >= 18:
3          d = True
4      else:
5          d = False
6      return d
```

À vous de jouer, servez-vous des trois codes ci-dessus pour compléter le tableau suivant :

N°	Nom de la fonction	Nom du paramètre	Rôle présumé de la fonction
1			*Cette fonction retourne ...*
2			*Cette fonction retourne ...*
3			*Cette fonction retourne ...*

Structure d'une fonction en Python, lorsqu'elle attend plusieurs paramètres

Lors de la création (*la définition*) d'une nouvelle fonction, il nous est aussi possible de lui faire accepter **plusieurs** paramètres.

Prenons par exemple le code Python de notre fonction nommée « somme » :

```
1  def somme(a, b):
2      resultat = a + b
3      return resultat
```

Comme nous pouvons le voir dans cet exemple, chaque **nom de paramètre** est séparé l'un de l'autre par une **virgule**. *Ici, dans cet exemple, la fonction « somme » accepte deux paramètres, nommés* `a` *et* `b`.

➤ Exercice 4 — Enquêter

Dans chacun des codes Python ci-dessous, déterminez le nombre de paramètres d'entrée, ainsi que leurs noms respectifs :

N°	Code Python	Nombre de paramètres	Noms des paramètres

1	`def aire_cercle(r):`
	`  a = 3.14 * r ** 2`
	`  return a`		

2	`def aac(age):`
	`  if age >= 15:`
	`  p = True`
	`  else:`
	`  p = False`
	`  return p`		

3	`def aire_rectangle(w, h):`
	`  res = w * h`
	`  return res`		

4	`def prix_ttc(prix_ht, tva):`
	`  c = prix_ht + prix_ht * tva / 100`
	`  return c`		

Création et appel de fonction

Créer une nouvelle fonction permet de l'exécuter (*l'appeler*) **à tout moment** et **autant de fois que désiré** dans un programme.

Prenons un exemple :

```
1  def prix_ttc(prix_ht, tva):
2      c = prix_ht + prix_ht * tva / 100
3      return c
4
5  switch2 = prix_ttc(320, 20)
6  ps5 = prix_ttc(350, 20)
7  livre = prix_ttc(20.5, 5.5)
8  nuit_hotel = prix_ttc(108, 10)
```

Dans cet exemple, nous avons défini une nouvelle fonction nommée `prix_ttc` (lignes 1-3).

Une fois créée, cette fonction `prix_ttc` peut être utilisée (*appelée*) autant de fois que souhaité dans notre programme :

- Nous avons appelé la fonction pour calculer la valeur stockée dans la variable `switch2` (ligne 5).
- Nous avons appelé la fonction pour calculer la valeur stockée dans la variable `ps5` (ligne 6).
- Nous avons appelé la fonction pour calculer la valeur stockée dans la variable `livre` (ligne 7).
- Nous avons appelé la fonction pour calculer la valeur stockée dans la variable `nuit_hotel` (ligne 8).

La création et l'appel de fonction permet de créer des programmes plus flexibles, en les découpant en « morceaux réutilisables ».

➤ Exercice 5 — Enquêter

Lisez la fonction `logico` ci-dessous, puis déterminez la valeur de `r` dans les différents appels proposés :

Code de la fonction	Appels	Valeur de `r` (à compléter)

`def logico(a, b):`
`  if a > 0:`
`  c = a * b`
`  else:`
`  c = a + b`
`  return c`
	`r = logico(10, 10)`	

	`r = logico(5, 10)`	
	`r = logico(-12, 10)`	
	`r = logico(-1, 1)`	
	`r = logico(100, 1)`	

Descriptif → Prototypage → Code Python

Penchons-nous à présent sur *l'art* de passer de ① la description complète d'une fonction (son *descriptif*) à ② son schéma (son *prototypage*) puis à ③ son code Python. **Prenons un exemple :**

① Descriptif de fonction :

> Créez une fonction `aire_rectangle()` qui prend comme paramètres `longueur` et `largeur` (entiers).
> Cette fonction calcule l'aire d'un rectangle de longueur `longueur` et de largeur `largeur`.
> *Pour rappel, l'air d'un rectangle se calcule en faisant longueur × largeur.*
> Puis, la fonction retourne le résultat entier de ce calcul.

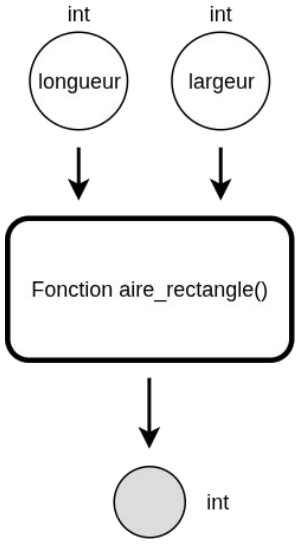
② Prototypage de la fonction :

> Pour prototyper une fonction, nous devons lire *attentivement* son descriptif.

Après une lecture attentive de son descriptif, il nous faut isoler :

- Le **nom** de la fonction.
  Dans cet exemple, la fonction se nomme « aire_rectangle ».
- Le(s) **nom(s)** et le(s) **type(s)** des ses **paramètres d'entrée**.
  *Dans cet exemple, nous avons deux paramètres d'entrée nommés **longueur** et **largeur**,*
  *et ce sont tous les deux des **entiers**.*
- Le **type** de sa **valeur de sortie**.
  Dans cet exemple, il est dit que "la fonction retourne le résultat entier..."
  *Donc sa valeur de sortie est un **entier**.*

Une fois toutes ces informations collectées, nous pouvons en réaliser le prototypage :



*Et enfin, **en nous aidant du descriptif, et du prototypage**, nous pouvons transcrire la fonction en code Python.*

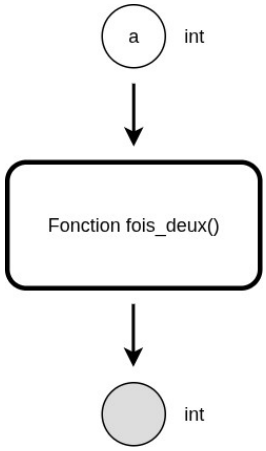
③ Code Python de la fonction :

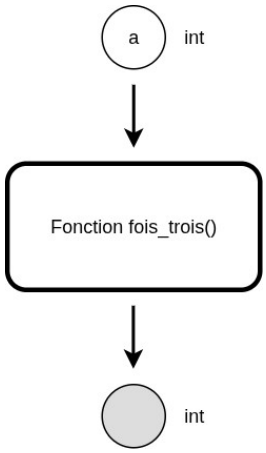
```
1  def aire_rectangle(longueur, largeur):
2      resultat = longueur * largeur
3      return resultat
```

➤ **Exercice 6** — Modifier

Observez la première fonction donnée en exemple (rangée n°1 du tableau ci-dessous)
puis complétez les cases « prototypage » et/ou « code Python » vides des rangées suivantes :

N°	Descriptif	Prototypage	Code Python

1	Créez une fonction `fois_deux()` qui prend comme paramètre `a` (entier).
	
	Cette fonction calcule le résultat de `a` multiplié par la valeur `2`.
	
	Cette fonction retourne le résultat entier de ce calcul.
			`def fois_deux(a):`
			`  res = a * 2`
			`  return res`

2	Créez une fonction `fois_trois()` qui prend comme paramètre `a` (entier). Cette fonction calcule le résultat de `a` multiplié par la valeur `3`. Cette fonction retourne le résultat entier de ce calcul.		
3	Créez une fonction `perimetre_cercle()` qui prend comme paramètre `r` (flottant). Cette fonction calcule le périmètre d'un cercle de rayon `r`. *Pour rappel, le périmètre d'un cercle se calcule en faisant 2 × 3.14 × rayon.* Cette fonction retourne le résultat flottant de ce calcul.		