fonctionspremiereapprochecalculatrice

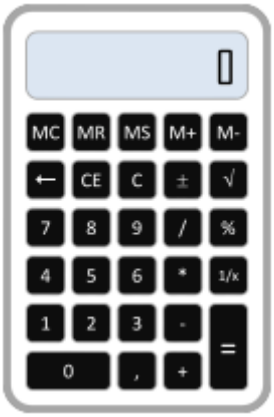
October 15, 2024

1 Première approche - La calculatrice

Produire un code clair et optimisé est une priorité d'un bon développeur. *** **Objectif:** Factoriser le code à l'aide des fonction ***

1.1 Sommaire

1.2 La calculatrice



Derrière chaque touche une fonction est exécutée.

La fonction peut être vue comme une boîte noire utilisable quand on en a besoin.

À retenir
>Une **fonction** :
- accepte (éventuellement) des paramètres en entrée,
- renvoie (éventuellement) une valeur
***

1.3 Fonction en informatique

1.3.1 Description

À retenir
>En informatique, une **fonction** est un outil autonome, utilisable dans le programme quand cela est nécessaire.
- Elle accepte 0, 1, ou plusieurs **paramètres**.
- Elle **renvoie** éventuellement une valeur, un objet….
***

1.3.2 Déclaration : construire la touche de la calculatrice

```
def carre(x: int) -> int:
    """
    calcule le carré d'un nombre
    Paramètres:
        x (int): le nombre
    Renvoie:
        int: le résultat de x*x
    """
    resultat = x*x
    return resultat
```

La fonction carré de la calculatrice.

`def`

Le mot-clé pour déclarer une fonction

`carre(x: int) -> int`

La signature de la fonction caractérise ses entrées et sorties.

Signature de la fonction La fonction **carre** accepte un **paramètre** de type entier et **renvoie** un entier. ***

Remarque
> Il est possible d'écrire la signature de la fonction comme ci-dessous :
> `python >  def carre(x): > >`
>
> Cependant on préférera fournir un maximum d'informations :

```
>   python >    def carre(x: int) -> int: >         """ >        calcule le carré
d'un nombre >        Paramètres: >           x (int): le nombre >       Renvoie:
>            int: le résultat de x*x >        """ > ***
```

À retenir
La **docstring** d'une fonction est un élément indispensable dans un travail d'équipe. ***

`resultat = x*x`

Corps de la fonction

`carre(x: int) -> int`

– Le mot-clé **return** permet de renvoyer la valeur.

À retenir
Certaines fonctions ne renvoient aucun résultat. La ligne avec return n'est donc pas obligatoire dans une fonction. ***

Activité 1

Consigne Dans le même fichier :
1. Écrire la fonction **cube(x: int) -> int** qui renvoie le résultat de l'opération x^3.
2. Écrire la fonction **puissance(x: int, n: int) -> int** qui renvoie le résultat de l'opération x^n.

Correction Cliquez ici pour afficher la solution

```
def cube(x: int) -> int:
    """
    calcule le cube de x

    Paramètres:
        x (int): le nombre

    Renvoie:
        int: le résultat de x^3
    """
    return x*x*x
# ou alors return x**3

def puissance(x: int, n: int) -> int:
    """
    calcule la puissance n de x
    Paramètres:
        x (int): le nombre
        n (int): la puissance
    Renvoie:
        int: le résultat de x^n
```

```
    """
    return x**n
```

Remarque
>Dans un souci de concision, la **docstring** n'est pas systématiquement indiquée dans la correction.

Activité 2

Consigne Pour calculer la racine carrée d'un entier n, on peut appliquer l'algorithme de Héron :
- Initialiser la valeur de la racine à la moitié de n.
- Répéter l'opération suivante dix fois : $racine = \frac{racine + \frac{n}{2}}{2}$

Écrire la fonction **racine_carree(n: int) -> float** qui renvoie une valeur approchée de $\sqrt{n}$ en appliquant l'algorithme de Héron.

Correction Cliquez ici pour afficher la solution

```
def racine_carree(n: int) -> float:
    racine = n//2
    i = 0
    while i < 10:
        racine = (racine + n/racine)/2
        i = i+1

    return racine

def racine_carree(n: int) -> float:
    racine = n//2
    for i in range(10):
        racine = (racine + n/racine)/2
    return racine
```

Une autre solution

1.3.3 Appel : Utiliser la touche de la calculatrice

À retenir
Écrire une fonction c'est créer un outil utilisable n'importe où dans le programme.
Il faut **appeler** la fonction (dans le programme principal, dans une autre fonction) pour l'exécuter.

```
carre(4)
# 2 16
```

Appel (dans la console) de la fonction carre avec l'argument 4

À retenir
- La fonction carre accepte le paramètre x de type entier.
- Quand on l'appelle on lui passe en argument l'entier 4.

Activité 3

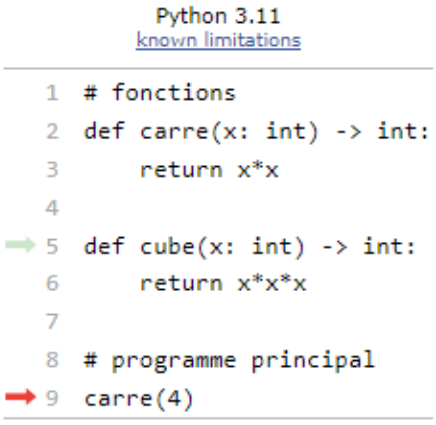
Consigne

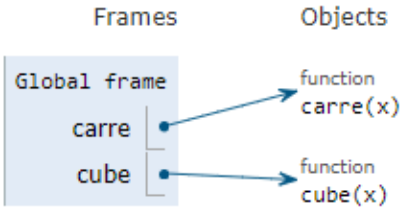
1. Ouvrir le site python tutor
2. Cliquer sur Python
3. Écrire le code ci-dessus et visualiser l'exécution.

Correction Cliquez ici pour afficher la solution

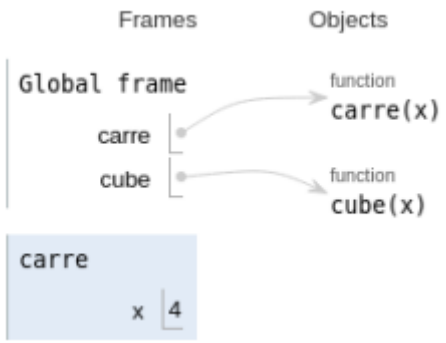
```
def racine_carree(n: int) -> float:
    racine = n//2
    i = 0
    while i < 10:
        racine = (racine + n/racine)/2
        i = i+1

    return racine

def racine_carree(n: int) -> float:
    racine = n//2
    for i in range(10):
        racine = (racine + n/racine)/2
    return racine
```

Une autre solution





À la fin de la ligne 6, les deux fonctions sont stockées dans la mémoire (et attendent qu'on les utilise).



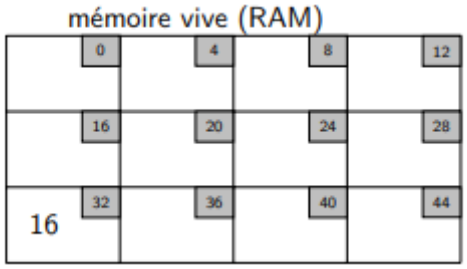
À la ligne 9, on utilise la fonction carre

Remarque
Lors de l'exécution de la fonction nous ne sommes plus dans le programme principal mais dans le corps de la fonction.
***

Activité 4

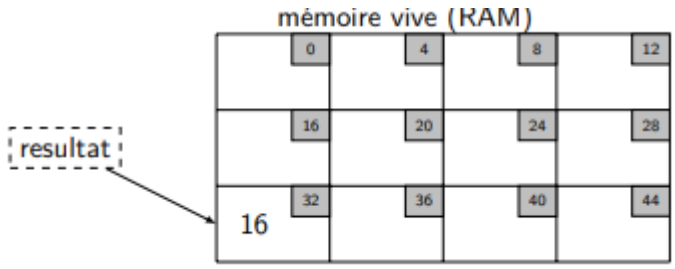
Consigne

1. Dans Thonny, ouvrir l'onglet **variables** ; fermer l'onglet **allocations mémoire**
2. Exécuter le code ci-dessus en mode débogage. Avancer avec **F7**



Correction *La fonction renvoie un entier au programme principal. Il est stockée dans la mémoire mais il est inaccessible.*

```
resultat = carre(4)
```



La fonction renvoie un entier au programme principal. Il faut l'utiliser : on le stocke

```
resultat = carre(4)
print( "Le carré de 4 est", resultat )
# ou alors
print( "Le carré de 4 est", carre(4) )
```

Programme principal

La fonction renvoie un entier au programme principal. Il faut l'utiliser : on l'affiche ici

```
Le carré de 4 est 16
```

Activité 5

Consigne Pour l'entier 2, afficher :
- le carré,
- le cube,
- la puissance de 10,

Correction Cliquez ici pour afficher la solution

```
print("Le carré:", carre(2))
2 print("Le cube:", cube(2))
3 print("La puissance 10 de 2:", puissance(2, 10))
4 print("La racine carrée:", racine_carree(2))
```

À retenir
L'ordre des arguments est imposé par la signature de la méthode.
