fonctions, 1e partie

November 5, 2019

La fonction mathématique “inverse”, définie de R dans R, et qui à chaque x associe son inverse -x, est définie en python de la manière suivante:

[1]:
```
def inv(i):
    return -i
```

i est le *paramètre* (formel): implicitement, puisqu'on utilise l'opération -, il faut que la valeur passée soit d'un type compatible avec cette opération (int ou float). l'instruction de “renvoi” ou “retour” de la valeur de la fonction est *return*.

La versions suivante, vue en cours, explicite les étapes du calcul en utilisant une variable supplémentaire, et utilise la *docstring* qui permet d'ajouter un commentaire à la définition d'une fonction.

[2]:
```
def inverse(i):
    "renvoie l'inverse de i (réel ou entier)"
    j = -i
    return j
```

Exemple d'utilisation: appel de la fonction avec le paramètre -4

[3]:
```
x = inverse(-4)
print(x)
```

```
4
```

Une fonction peut être définie avec deux arguments, comme par exemple la fonction somme().

[4]:
```
def somme(a,b):
    return a + b
```

[6]:
```
print("la somme de 3 et 4 vaut", somme(3,4))
```

```
la somme de 3 et 4 vaut 7
```

Passons maintenant à une fonction qui n'est pas déjà pré-définie: la conversion d'une température donnée en degrés Celsius vers une température en degrés Farenheit.

[7]:
```
def farenheit(t):
    "convertit des degrés celsius en farenheit"
    f = (9/5)*t + 32
    return f
```

[8]:
```
print(farenheit(99.9))
print(farenheit(38.5))
```

```
211.82000000000002
101.3
```

Utilisons cette fonction pour écrire un petit programme de conversion, qui demande une température en Celsius et affiche la valeur convertie. Première version, très explicite, avec toutes les étapes détaillées:

[9]:
```
x = input("quelle est la température à convertir ? ")
xi = int(x)
yi = farenheit(xi)
print(yi)
```

```
quelle est la température à convertir ? 37
98.60000000000001
```

Deuxième version, où l'on exploite à fond le fait que toutes les opérations réalisées sont en fait des "calculs de fonction":

[10]:
```
print(farenheit(int(input("quelle est la température à convertir ? "))))
```

```
quelle est la température à convertir ? 37
98.60000000000001
```

On pourrait envisager de découper le travail différemment, en définissant une fonction qui ne renvoie rien, mais qui affiche directement la conversion:

[13]:
```
def aff_farenheit(t):
    "convertit des degrés celsius en farenheit"
    f = (9/5)*t + 32
    print(f)
    return
```

Dans ce cas, on peut vérifier que la fonction ne renvoie "rien":

[14]:
```
x = aff_farenheit(36)
print(type(x))
# print(x)
```

```
96.8
<class 'NoneType'>
```

Le "programme principal" qui fait appel à la fonction farenheit s'en trouve évidemment simplifié:

[15]:
```
x = int(input("quelle est la température à convertir ? "))
aff_farenheit(x)
```

```
quelle est la température à convertir ? 38
100.4
```

[16]:
```
y = farenheit(34) + farenheit(22)
y = aff_farenheit(34) + aff_farenheit(22)
```

```
93.2
71.6
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-16-bd7504b873f4> in <module>
         1 y = farenheit(34) + farenheit(22)
    ----> 2 y = aff_farenheit(34) + aff_farenheit(22)


       TypeError: unsupported operand type(s) for +: 'NoneType' and 'NoneType'
```

Poussons un cran plus loin: pourquoi ne pas inclure la commande d'input() dans la fonction elle-même ?

[18]:
```
def convert_farenheit():
    t = int(input("quelle est la température à convertir ? "))
    f = (9/5)*t + 32
    print(f)
```

Alors le programme principal s'est trouve encore plus simple.

[19]:
```
convert_farenheit()
```

```
quelle est la température à convertir ? 42
107.60000000000001
```

Discussion : il faut réfléchir au "découpage" en fonctions : définir les meilleurs regroupements possibles d'instruction qui permettent de faire du code lisible, facile à débugger et efficace.

1 Organisation d'un fichier

Revenons en arrière et proposons une division du travail comme suit, entre une fonction qui ne fait que la conversion, et un programme principal, que l'on appellera *main*. Dès lors, on peut envisager la programmation comme la réalisation d'un certain nombre de fonctions (au sens large), qui sont définies dans le fichier, et qui sont executées selon les besoins par la seule ligne de commande du programme, qui appelle le programme principal.

[20]:
```
def farenheit(t):
    "convertit des degrés celsius en farenheit"
```

```
    f = (9/5)*t + 32
    return f

def main():
    t = int(input("quelle est la température à convertir ? "))
    print(farenheit(t))

if __name__ == "__main__":
    main()
```

```
quelle est la température à convertir ? 66
150.8
```