nsi_1_1_introduction

April 15, 2022

1 Premiers pas avec le langage Python

1.1 Utilisation du notebook

Pour cette séance, nous allons utiliser Python en mode "notebook". Le fichier que vous venez d'ouvrir, "introduction à Pyhton.ipynb", et qui est affiché ici-même, est un notebook. Faites-en une copie avec la commande ci dessus "File>Make a Copy…", et renommez-le en cliquant sur le titre (à côté de jupyter TD1…), par exemple en rajoutant votre nom.

On peut entrer des expressions (opérations, calculs) ou des instructions (des commandes) dans tous les champs ci-dessous qui commencent par `In[..]`. Pour entrer plusieurs lignes à la suite, on appuie sur la touche entrée.

Le résultat s'obtient avec :
* ctrl + entrée, dans ce cas on reste dans la cellule courante; * majusucule + entrée, dans ce cas on passe à la cellule suivante. * Après exécution, le résultat d'un programme est souvent précédé de Out[ ].

Il est possible qu'un notebook ne réponde plus lors de la tentative d'éxécution d'un commande ou d'un programme. C'est notamment le cas lorsqu'un programme précédent à planté. Commencez par sauver votre travail. Puis allez dans l'onglet "Home", et fermez le TD : cocher la case correspondante, "shutdown" en haut de la page. Fermez ensuite l'onglet du TD, et relancez-le à partir de "Home".

1.2 Expressions et opérations de base

Testez les opérations usuelles (`+ - * /`) dans le champ ci-dessous. On peut ré-écrire sur ce qui est déjà présent.

[ ]:
```
5 + 7
```

Testez le parenthésage:

[ ]:
```
7 + 4*3
```

[ ]:
```
(7+4)*5
```

Enfin, testez les opérateurs suivants `// % **`. Complétez votre cours (§ VI.1) en précisant leur effet.

[ ]:
```

```

1.3 Variables et affectations - entrées-sorties

Une variable est une mémoire qui permet de stocker une valeur numérique, du texte, une liste de courses,... On affecte une valeur à une variable avec `=` . On affiche une variable avec l'instruction `print(variable)`: c'est une instruction de sortie (output).

[ ]:
```
a = 2
print(a)
```

Attention : Le symbole "=" n'a pas du tout la même signification qu'en mathématiques !

L'équation :

$$a = a + 1$$

n'a pas de solution en mathématiques, par contre ceci fonctionne :

[ ]:
```
a = 5
a = a + 1
print(a)
```

On peut affecter une *expression* à une variable:

[ ]:
```
b = 5
c = 2*a + b**2
print(c)
```

Une instruction d'entrée importante est input(). Ci-dessous, on affecte l'entrée à la variable *prenom*.

Remarque : il n'y a pas d'accent à la variable *prenom*, c'est volontaire : cf. le cours photocopié.

[ ]:
```
prenom = input("Donnez votre prénom : ")
print("bonjour",prenom)
```

Qu'est censé faire le code suivant ? Testez-le, et concluez :

[ ]:
```
a = input("entrez un nombre :")
print(3*a)
```

1.4 Typage des données

Comme vous venez de le voir, le résultat du programme précédent n'est pas celui attendu. Les données et les variables sont **typées** dans un langage de programmation, c'est-à-dire que l'on doit préciser si ce sont des nombres, du texte, etc... En python le typage est très souple, ce qui facilité l'écriture des programmes lorsque l'on débute. Néanmoins, il est parfois nécessaire de le préciser. L'instruction `type(variable)` permet de savoir quel est le type d'une variable. Testez et concluez (vous pouvez écrire vos idées après "commentaires" avec un double clic sur la ligne "commentaires", puis ctrl + entrée pour avoir le format texte) :

[ ]:
```
a = 3
type(a)
```

[ ]:
```
a = input("entrez un nombre :")
type(a)
```

Commentaires :

Les types principaux que nous utiliserons cette année sont: * les entiers : type `int`; * les flottants (approximation des nombres réels par des décimaux): type `float`; * les caractères (lettres) : type `char`; * les booléens (vrai ou faux) : type `bool`; * les listes (ou tableaux), que l'on verra ultérieurement : type `list`; * les chaines de caractères (mots, phrases), qui sont des listes de lettres en fait : type `str`. Elles sont entre guillemets simples `'chaine'` ou doubles `"chaine"`.

On peut forcer le type de certaines variables en rajoutant le type souhaité devant la variable.
* Comparez le code ci-dessous avec les précédents ; * puis enlevez le `int` devant le input et testez à nouveau; enfin concluez.

[ ]:
```
a = int(input("entrez un nombre :"))
print(type(a))
print(3*a)
```

On utilise la notation anglo-saxonne avec . pour la virgule, et e pour "10 puissance". Par exemple 1.5e4 = 15000

Des phénomènes étranges peuvent se passer avec les flottants… tester dans la cellule suivante 0.1 + 0.1 puis 0.1 + 0.1 + 0.1 etc.. Conclure.

[ ]:
```
0.1 + 0.1
```

Commentaires :

Remarque : le type complexe est un type de nombre particulier, que les chanceux pourront utiliser en mathématiques en terminale. On rajoute aux nombres réels le nombre i, noté `1j` en Python. Tester i^2, qu'en pensez-vous ?

[ ]:
```
1j * 1j
```

1.5 Un exemple de programme structuré

Le programme ci-dessous calcule le périmètre et la surface d'un cercle de rayon donné par l'utilisateur.
Il est découpé en trois **fonctions** (au sens informatique du terme), qui ont chacune un rôle différent :
* la fonction d'*entrée*, permet de donner les informations nécessaires au programme pour qu'il fasse son travail ; * la fonction de *sortie* permet d'afficher les résultats du programme ; * la fonction de calcul, dite fonction *métier* effectue le travail demandé.

Le programme principal **appelle** ces trois fonctions, et fait passer les **paramètres** de l'une à l'autre.

Reamrque : un mathématicien rigoureux écrit surface d'un disque et périmètre d'un cercle (un cercle a une surface nulle, c'est l'épaisseur de la ligne…)

[3]:
```
# Programme calculant la surface et le périmètre d'un cercle.

def entree_des_donnees() :
    nom = input("Donnez le nom du cercle : ")           # nom est une chaine,␣
 ↪type <str>
    r = int(input("Donnez le rayon du cercle : "))      # r est un entier, type␣
 ↪<int>
    return (nom, r)                                     # on fait passer les␣
 ↪paramètres au programme principal
                                                        # on dit qu'on renvoie␣
 ↪ou retourne le résultat

def affichage_resultats(nom, p, s) :
    """
    La fonction d'affichage récupère les paramètres passés par le programme␣
 ↪principal :
    nom : nom du cercle
    p : périmètre du cercle
    s : surface du cercle
    """
    print("Le cercle ", nom, " a pour périmètre ", p, "unités de longueur, et␣
 ↪pour surface ", s, "unités de surface")
    return       # la fonction d'affichage ne renvoie aucun paramètre

def calcul(r) :
    """
    La fonction de calcul récupère le paramètre passé par le programme principal
    r : rayon du cercle
    """
    PI = 3.141592653589793 # on donne un nom à la valeur approchée de pi plutôt␣
 ↪que de la réécrire à chaque fois
    p = 2 * PI * r
    s = PI * r**2
    return (p, s)     # on retourne les résultats du calcul au programme␣
 ↪principal

# PROGRAMME principal

# Appel de la fonction d'entrée des données. Après cet appel,
#      la variable "nom_cercle" a la même valeur que "nom" dans la fonction␣
 ↪entree_des_donnees()
#      la variable "rayon_cercle" a la même valeur que "r" dans la fonction␣
 ↪entree_des_donnees()
(nom_cercle, rayon_cercle) = entree_des_donnees()

# Appel de la fonction de calcul. Pendant cet appel :
```

```
#      la variable "r" de calcul va prendre la valeur de "rayon_cercle"
#    Après cet appel,
#      la variable "perimetre" a la même valeur que "p" dans la fonction␣
 ↪calcul()
#      la variable "surface" a la même valeur que "r" dans la fonction calcul()
(perimetre, surface) = calcul(rayon_cercle)

# Appel de la fonction d'affichage.
# La logique est la même que pour l'appel de la fonction calcul(). Comme il n'y␣
 ↪a pas de variables renvoyée,
# on ne met pas de " = " avant l'appel de la fonction.
affichage_resultats(nom_cercle, perimetre, surface)
```

```
Donnez le nom du cercle : t
Donnez le rayon du cercle : 4
```

Mettez un # de commentaire devant le programme précédent et testez-le à nouveau. A votre avis, que se passe-t-il précisément ?

Copier le code ci-dessus et collez-le dans python tutor. Cliquer ensuite sur “Visualize execution”, vous verrez le passage des valeurs des paramètres.

1.5.1 A vous de jouer

Ecrire un programme qui demande votre prénom, et votre âge en année et convertit ce dernier en nombre de jours. Le programme doit écrire en sortie une phrase du type “Bonjour Archibald, vous avez 37256 jours”.

En exemple ci dessous, les premières lignes avec des commentaires. Il ne vous reste qu’à compléter !

[ ]:
```
"""
TD1_ex1_fait_par_Super(wo)man
Programme permattant de convertir un nombre d'annees en nombre de jours
"""
def entree_des_donnees():
    p = input("Donnez votre prenom : ")

    return ?

def affichage_resultats(p, ?):
    ?
    return

def calcul_jours(?) :
    ?
    return ?

???
```

frederic.mandon@ac-montpellier.fr, Lycée Jean Jaurès - Saint Clément de Rivière - France