

Travaux dirigés n° 1

Instruction, variable et instruction conditionnelle

La réalisation des TD nécessite d'installer Python 3 sur votre ordinateur. Vous trouverez ce programme sur le site <https://www.python.org>. Vous trouverez aussi des indications pour l'installation sur ce site <http://docs.python-guide.org/en/latest/>

Vous trouverez par la suite des exercices obligatoires et des exercices pour approfondir. Les exercices obligatoires sont précédés par le caractère *.

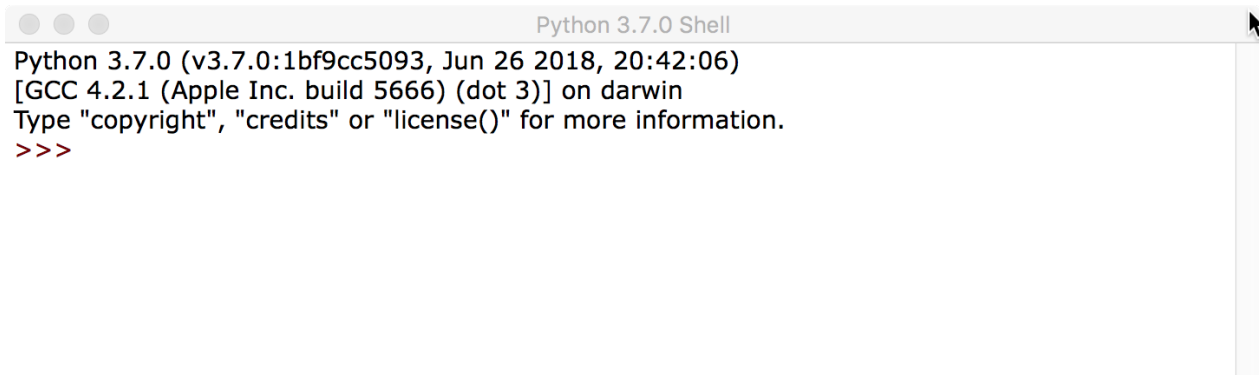
1 Utilisation de l'interpréteur de Python

Le langage Python peut être utilisé de plusieurs manières. Pour commencer vous allez l'utiliser en mode interactif à l'aide de l'interpréteur Python, c'est à dire en « dialoguant » avec lui directement à l'aide du clavier.

Pour accéder à l'interpréteur de Python, il suffit de lancer l'application IDLE. Sous Mac OSX, IDLE se trouve dans le dossier Applications, sous-dossier Python mais vous pouvez aussi taper `idle3&` dans un terminal. Sous Windows, allez dans le menu Démarrer, trouvez le dossier Python et lancer IDLE.

L'interpréteur peut aussi être lancé depuis la ligne de commande (dans un terminal sous OSX ou Linux ou bien depuis l'invite de commande sous Windows) : il suffit de taper la commande `python3`.

Après le lancement de IDLE (ou de Python dans un terminal), vous obtenez une fenêtre similaire à celle ci-dessous. Les chevrons `>>>` indiquent que l'interpréteur est prêt à évaluer l'instruction que vous allez écrire. On définit l'instruction comme l'opération élémentaire d'un langage de programmation,



Ln: 4 Col: 4

1.1 Faire des calculs


*Exercice 1.1

Python est un langage de programmation interprété à savoir qu'il interprète les instructions et les exécute au fur et à mesure qu'il les reçoit. Aussi il est possible de l'utiliser en mode interactif comme une calculatrice. C'est ce que nous allons faire dans cette exercice. Vous allez saisir des instructions qui seront des expressions. on rappelle qu'une expression se définit comme un élément de syntaxe qui combine des littéraux, des variables, des opérateurs, et des fonctions pour en faire l'évaluation et retourner une valeur.

1) Lancez l'interpréteur Python et utilisez-le comme une calculatrice pour évaluer les expressions suivantes :

```
5+3
2 - 9
5 + 3 * 2
```

```
(5+3) * 2
20 / 3
20.0 / 3
20 // 3
20.0 // 3
20 % 3
sqrt(2)
```

L'interpréteur Python exécute l'instruction lorsque vous frappez sur la touche entrée (). Faites bien attention à la valeur retournée par l'interpréteur.

- 2) Python respecte-t-il les priorités des opérations arithmétiques ?
- 3) Que se passe-t-il lorsqu'on mélange des entiers et des flottants dans un même calcul ?
- 4) Que font les opérateurs `/` et `//` ?
- 5) Que calcule l'opérateur `%` ?

*Exercice 1.2

La dernière expression de l'exercice 1.1 produit une erreur. En effet, les fonctions mathématiques ne sont pas définies dans l'interpréteur de Python, seules les opérations élémentaires le sont.

Comme beaucoup d'autres langages, Python utilise des modules (ou bibliothèques) où sont définis des ensembles de fonctions spécifiques. Ainsi avec les modules, Python peut s'étendre et s'adapter facilement à des développements spécifiques. Pour avoir accès aux fonctions définies dans un module il faut l'indiquer à l'aide de la clause `import`.

- 1) Tapez et exécutez les expressions suivantes et observez et analyser la réponse de l'interpréteur.

```
pi
from math import *
sqrt(2)
sqrt 2
sin(90)
pi
sin(pi/2)
sin(radians(90))
```

Respectez bien la syntaxe pour l'importation, le caractère `*` est indispensable, il indique qu'on veut importer toutes les fonctions du module.

Pour connaître les fonctions qui sont définies dans un module, il faut utiliser la commande `help("nom_du_module")`. L'interpréteur affiche alors le descriptif de toutes les fonctions qu'il contient (dans un terminal tapez la barre d'espace pour afficher la suite et `q` pour quitter l'affichage). Pour obtenir l'aide sur une fonction en particulier, tapez (après avoir importé le module dans l'environnement) `help(fonction)` sans les guillemets !

- 2) Essayez les commandes `help("math")` et `help(sqrt)` (après avoir exécuté `from math import *`).

Exercice 1.3

- 1) Évaluez les expressions suivantes dans l'interpréteur :

```
13 // 3
13 % 3
3 ** 2
2 ** 0.5
sqrt(2)
```

- 2) Comment savoir si un nombre est impair ?

1.2 Utiliser des variables

Une variable en programmation sert à stocker le résultat d'une expression ou la valeur d'une constante. La variable est identifiée par un nom.

En Python, le nom d'une variable est une séquence de lettres minuscules ou majuscules et de chiffres qui doit toujours commencer par une lettre. Les caractères accentués, les cédilles, les espaces, les caractères de ponctuations et les caractères spéciaux (&, #, @ ...) sont interdits. Le caractère souligné est autorisé. Bien entendu le nom d'une variable ne doit pas non plus être un mot réservés du langage.

Une variable prend une valeur suite à une instruction d'affectation.

*Exercice 1.4

Décrivez le plus complètement possible ce qui se passe lors de l'évaluation de chacune des instructions suivantes :

```
n = 7

rac2 = 1.414

x = n + rac2

print(n + rac2)

msg = "Rien a signaler"

msg

msg2 = msg

print(msg2)

msg3 = print(msg2)

msg3

print(msg3)
```

Notez bien la différence entre le contenu d'une variable et l'affichage du contenu d'une variable !

*Exercice 1.5

Testez les expressions suivantes :

```
a=10 ; b=33
a < b

b < a

a+23 = b

a+23 == b

c = b == a + 23

c

"abcd" < "abc"

"d" < ""

"4" > "123"
```

Attention de ne pas confondre affectation et égalité!

1) Comment s'effectue la comparaison de chaînes de caractères?

Exercice 1.6

1) Testez les instructions suivantes :

```
int(12)
```

```
float(12)
```

```
int(2.71)
```

```
float("10.7")
```

```
int("10.7")
```

```
int("10") + 5
```

2) Que font les fonctions `int` et `float`?

Exercice 1.7

Pour chacune des expressions booléennes suivantes affectez les valeurs `True` ou `False` aux variables `a`, `b`, `c` de telle sorte que la valeur de l'expression soit `True`.

```
a or b and c
```

```
a and not b or c
```

```
a and (not b or a)
```

```
a and not(b or a)
```

```
(a or b) and not(a and b)
```

Exercice 1.8

1) Testez les instructions suivantes :

```
r= 5
```

```
pi = 3.1415927
```

```
s = pi * r ** 2
```

```
s
```

```
type(r)
```

```
type(pi)
```

```
type(s)
```

```
type("surface")
```

```
type(sqrt)
```

2) A quoi sert la fonction `type`?

1.3 Instruction conditionnelle

Une instruction conditionnelle sert à ajouter une action effectuée en fonction de l'évaluation d'une condition booléenne, à savoir vraie ou fausse. Des alternatives entre différents blocs d'instructions sont ainsi insérées dans un flux d'instructions (un programme).

Comme la structuration d'un programme Python se définit par son indentation. L'introduction d'un bloc d'instructions se termine par le caractère "deux points". Le bloc d'instructions commence à la ligne suivante la ligne d'introduction avec une indentation, par convention de 4 espaces. On n'utilise pas le caractère de tabulation sauf si l'éditeur de texte le transforme en 4 espaces. La fin du bloc se termine par le retour à l'indentation de la ligne d'introduction.

L'instruction conditionnelle utilise les mots clefs `if`, `elif` et `else`. La valeur de la condition est fausse si l'expression retourne une valeur, vide, 0 ou `False`.

*Exercice 1.9

Lorsque vous écrivez une instruction conditionnelle, les trois points qui apparaissent quand vous passez à la ligne (en frappant la touche **Entrée** du clavier) signifient que l'instruction que vous tapez n'est (peut-être) pas terminée. Frappez une fois de plus la touche **Entrée** pour indiquer que c'est effectivement terminé et que l'interpréteur peut maintenant exécuter l'instruction.

Attention à la syntaxe de l'instruction conditionnelle : n'oubliez pas les deux points juste après la condition avant de passer à la ligne.

1) Tapez les instructions suivantes dans l'interpréteur de Python en prenant bien soin de taper une tabulation après les 3 points affichés par l'interpréteur lorsque c'est nécessaire. Les tabulations sont indispensables en langage Python et délimitent les blocs d'instructions!

```
a = 150
if (a > 100):
    print("a dépasse la centaine")
```

2) Recommencez la même instruction conditionnelle mais en affectant la valeur 20 à la variable `a`. Que se passe-t-il ?

3) Saisissez maintenant la séquence d'instructions suivantes. Quelle est la différence avec la séquence précédente ?

```
a = 20
if (a > 100):
    print("a dépasse la centaine")
else:
    print("a ne dépasse pas la centaine")
```

Exercice 1.10

Tapez une séquence d'instructions pour affecter des valeurs (de votre choix) à deux variables `x` et `y`, puis à l'aide d'expressions d'alternative, afficher `x est le plus grand` ou bien `y est le plus grand` ou bien `x égale y` selon les valeurs de `x` et `y`. Faites bien attention aux tabulations pour indenter les blocs d'instructions. En particulier si vous voulez passer des lignes à l'intérieur d'un bloc d'instructions vous devez quand même taper les tabulations sur les lignes vides.

Exercice 1.11

Tapez une séquence d'instructions pour affecter des valeurs à trois variables de votre choix puis, à l'aide d'alternatives, faites afficher le nom de la variable dont la valeur est la plus grande.

2 Premiers programmes

Dans les exercices précédents vous avez utilisé Python en mode interactif via son interpréteur. C'est pratique pour tester des instructions mais si vous voulez exécuter plusieurs fois les mêmes instructions en changeant quelques valeurs vous devez tout retaper.

Vous allez désormais taper vos instructions dans un éditeur de texte puis les conserver dans un fichier afin d'en faire un programme que vous pourrez exécuter avec Python.

Si vous utilisez IDLE, il suffit de choisir **New File** dans le menu **File** pour ouvrir un éditeur de texte dans lequel vous pouvez écrire votre programme.

*Exercice 1.12

Saisissez le programme suivant dans un éditeur de texte, enregistrez-le dans un fichier que vous nommez `ex12.py` (on utilise l'extension `.py` pour les programmes écrits en Python). Ne pas saisir les numéros de ligne

```

1 # ceci est un commentaire
2 # programme ex012.py
3 # mon premier programme python
4
5 # Il faut convertir le resultat de input qui est toujours une chaine de caracteres
6 a = int(input("Entrez un entier :"))
7 # On utilise des alternatives imbriquées
8 if a < 0 :
9     print("Cet entier est negatif")
10 else :
11     if a > 0 :
12         print("Cet entier est positif")
13         if a < 100 :
14             print("et plus petit que 100")
15         else:
16             print("et superieur ou egal a 100")
17     else:
18         print("Vous avez entre 0")

```

Lorsque vous avez terminé et enregistré votre programme, exécutez-le :

- soit en demandant la commande **Run Module** du menu **Run** dans IDLE
- soit depuis une fenêtre de commande, en vous plaçant dans le répertoire où vous avez enregistré votre programme et en tapant `python ex12`.

S'il y a des erreurs à l'exécution, corrigez votre programme dans l'éditeur, sauvez-le de nouveau et ré-exécutez-le.

*Exercice 1.13

Écrivez un programme qui affiche le menu suivant, puis demande à l'utilisateur de saisir son choix (l'utilisateur entre un entier) puis affiche OK si le choix est valide ou bien **Choix incorrect** sinon.

```

1. Enregistrer la partie
2. Charger une partie
3. Nouvelle partie
4. Quitter
Votre choix :

```

*Exercice 1.14

Écrivez un programme qui demande la saisie de trois entiers et affiche combien de ces entiers sont égaux à zéro. Pensez à utiliser des opérateurs logiques dans vos conditions.

*Exercice 1.15

Écrivez un programme permettant de convertir une température donnée dans une unité (Celsius, Fahrenheit ou Kelvin) dans les autres unités.

Les unités à considérer sont Celsius (C), Fahrenheit (F) et Kelvin (K) et les formules de conversion sont :

$$C = \frac{9}{5}(F - 32) \quad F = \frac{5}{9}C + 32$$

$$K = C + 273,15$$

Le programme demandera la valeur de la température puis l'unité dans laquelle elle est exprimée donnée sous la forme d'un caractère : C, F ou K.

Exemples d'exécution :

```
Donnez la temperature :27
Donnez l'unite C, F ou K : F
-9.0 C = 27.0 F = 264.15 K
```

```
Donnez la temperature :314
Donnez l'unite C, F ou K : K
40.85 C = 54.6944444444 F = 314.0 K
```

Exercice 1.16

Écrivez un programme qui demande la saisie, sous forme de flottant, de la longueur, de la largeur et de la hauteur d'un parallépipède rectangle et affiche son volume.

Exercice 1.17

Écrivez un programme qui demande la saisie d'un entier et qui produit l'affichage **Entier pair** ou **Entier impair** selon que l'entier saisi est pair ou impair.

Exercice 1.18

Écrivez un programme qui, selon le choix de l'utilisateur, permet de calculer l'aire et le périmètre d'un carré, d'un rectangle, d'un triangle rectangle (dont on connaît les longueurs des côtés de l'angle droit), d'un parallélogramme ou d'un losange.

Exemples d'exécution :

1. Carre
2. Rectangle
3. Triangle rectangle

Quelle figure :3

Donner la valeur du premier cote :2

Donner la valeur du deuxieme cote :3

Aire = 3.0

Perimetre = 8.60555127546

1. Carre
2. Rectangle
3. Triangle rectangle

Quelle figure :1

Donner la valeur du cote :9

Aire = 81

Perimetre = 36

