Algorithmes et Pensée Computationnelle

Introduction à Python

1 Introduction

1.1 Création d'un script Python

Pour commencer à programmer en python, il est nécessaire de créer un fichier appelé script python. Pour se faire, il suffit de créer un simple fichier vide et de changer l'extension du fichier. Il existe plusieurs manières d'effectuer cela. La manière la plus simple est de passer par le terminal de l'OS.

Conseil

Dans le terminal, entrer la commande **touch hello.txt** pour créer un fichier texte vide nommé hello.txt changer l'extension du fichier directement depuis l'explorateur de fichier de .txt à .py.

1.2 PyCharm

Lors du cours, il vous a été demandé d'installer PyCharm, l'IDE python le plus populaire de ces dernières années. Un IDE (Integrated Developement Environnement) est un programme qui combine différents outils de développement et qui facilite grandement le travail d'un programmeur.

Conseil

Lancez le programme PyCharm et ouvrez le fichier hello.py. Notez qu'il est aussi possible créer directement un fichier depuis PyCharm.

La fenêtre qui vient de s'ouvrir est similaire à un éditeur de texte basique. C'est ici que le code peut être entré et modifié.

1.3 Les fonctions

Dans les langages de programmation, on retrouve un très grand nombre de fonctions. Ces fonctions sont des blocs de code qui, lorsqu'ils sont invoqués avec certains paramètres, effectuent certaines actions. Une des fonctions basique et plutôt importante en python est la fonction **print()**. Cette fonction permet d'imprimer le contenu de la parenthèse qui suit la fonction.

Conseil

Entrez la ligne de code **print("Hello World")** puis exécutez le script en allant dans la section Run/Run... puis en sélectionnant hello. Le programme va afficher une petite fenêtre avec le message entré dans la parenthèse. C'est ici que le script écriras ce qu'on lui demande de faire.

En plus des fonctions incluses dans les librairies python, il est possible de créer des fonctions complètement personnalisées (généralement plusieurs fonctions sont réunies en une seule)au moyen de la fonction **def nomdelafonction :** et de l'utiliser à n'importe quel moment en l'invoquant avec **nomdelafonction**.

Solution

```
1  def ma_fonction_perso():
2      print("Ma_fonction_est_super_!")
3
4  ma_fonction_perso()
```

1.4 Indentation

Comme tous les autres languages de programmation, Python est sensible aux erreurs d'indentation. Il sera donc important de bien comprendre comment celles-ci fonctionnent. Prenons par exemple une simple boucle for qui sera traitée un peu plus loin dans ce cours.

Conseil

Entrez les deux scripts suivants dans PyCharm. Un des deux retourne une erreur d'indentation alors que l'autre fonctionne correctement

Solution

```
1  for i in range(4):
2      print("hello world")
```

Solution

```
1  for i in range(4):
2  print("hello world")
```

L'indentation est importante dans la syntaxe d'un script python. Une erreur d'indentation peut changer le fonctionnement d'un script ou tout simplement empêcher celui-ci de s'executer.

1.5 Création de variable

Les variables dans les langages de programmation sont similaires à des noms données à une valeur précise Pour assigner une valeur à une variable en python, il suffit de respecter la forme suivante **variable=valeur**.

Conseil

- En python, une variable est toujours dynamique (il n'existe pas de variable statique).
- On peut assigner n'importe quelle suite de caractères non-reservée en tant que variable
- Il est aussi possible d'assigner des chaînes de caractères (strings en anglais) ou encore des valeurs booléennes à des variables
- En réassignant une nouvelle valeur à une variable déjà définie, la valeur de la variable va être écrasée et remplacée par la nouvelle valeur
- Il est possible d'additionner les variables du même type
- Plusieurs variables peuvent avoir la même valeur

Voici quelques exemples d'attribution de variables. Entrez ces lignes de code dans PyCharm et observez ce que le programme vous renvois.

Solution

```
1  x=12, aBcD=32, a="Alpha", b=True, c=x
2
3  print("x vaut", x)
4  print("aBcD vaut", aBcD)
5  print("a vaut", a)
6  print("b vaut", b)
7  print("c vaut", x)
8
9  x=99
10
11 print("x vaut maintenant", x)
```