Introduction

Initiation à l'algorithmique et programmation

Revekka Kyriakoglou

Objectifs

L'objectif de ce cours est d'apprendre le language Python :

- produire du code propre,
- déboguer un programme python,
- résolution de problèmes,
- manipulation de Fichiers
- introduction à la Conception d'algorithmes.

Éléments importants :

- Vous avez accès à un ordinateur.
- Vous savez comment utiliser le web pour trouver des informations sur les sujet dont vous avez besoin.
- Vous êtes familiarisé avec l'utilisation du terminal.
- **Vous voulez apprendre à programmer en Python ! ! !**

Evaluation

Ce cours sera noté comme suit :

1. contrôle soutenu
2. petites interogations
3. projet/examen final

Pourquoi Python ?

- Syntaxe simple et lisible.
- Polyvalence (utilisé dans de nombreux domaines).
- Vaste bibliothèque standard et de nombreuses bibliothèques tierces et des frameworks spécialisés.
- Communauté active (vous pouvez trouver facilement des ressources, des tutoriels et de l'aide en cas de besoin).
- Portabilité (disponible sur de nombreuses plateformes, y compris Windows, macOS, Linux)
- Gratuit et open source.

Mise en place

Vous avez besoin de :

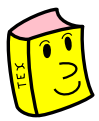
- compilateur,
- un endroit pour mettre vos programmes.

 Pour verifier si le python est installé sur les systèmes d'exploitation Linux, vous devriez taper : *python –version*

 Si vous utilisez une machine Windows, ouvrez l'invite de commandes en appuyant sur *Win + R*, en tapant "cmd" et en appuyant sur Entrée. Dans l'invite de commandes, tapez *python –version* et appuyez sur Entrée.

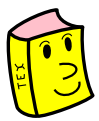
Créer un répertoire **prog/python/cours**

1. Ouvrir le terminal dans le répertoire HOME.



Si vous n'y êtes pas, tapez **cd** pour y retourner.

2. Créer le répertoire **prog/python/cours**



mkdir -p prog/python/cours

Editeur

Vous pouvez utiliser l'éditeur de votre choix.

Voici quelques exemples :

- Gedit
- Visual studio code
- Sublime Text
- Eclipse
- Atom

Les fichiers sources

- Le code en Python doit etre contenu dans un fichier dont le nom se termine par **.py**.

Les fichiers sources

Un fichier source écrit en langage Python est composé de :

- **Commentaires**
  Les commentaires commencent par le caractère **#** et sont utilisés pour ajouter des explications ou des notes dans le code.
- **Instructions**
  Les instructions sont les commandes exécutées par l'interpréteur Python pour effectuer des opérations (ex. opérations mathématiques, appels de fonctions etc).
- **Fonctions**
- **Imports de modules**


Syntaxe très précise à respecter !

Compilation

Python est un langage interprété, ce qui signifie qu'il n'a pas besoin d'être compilé avant l'exécution.

1. Ecrire directement le code source Python dans un fichier avec une extension ".py".
2. Exécuter ce fichier à l'aide de l'interpréteur Python.

Compilation – Comment ?

Dans un terminal (ou une invite de commande), naviguer jusqu'au répertoire où se trouve le fichier (ex "mon_programme.py"). Ensuite, vous pouvez utiliser la commande :

```
python3 mon_programme.py
```

Hello world !

```
1   # Affichage du message Hello world !
2   print("Hello, World!")
```

Hello world !

```
1   # Affichage du message Hello world !
2   print("Hello, World!")
```

- **Commentaires d'une seule ligne :** Les commentaires d'une seule ligne commencent par le caractère # et s'étendent jusqu'à la fin de la ligne.
- **Commentaires multilignes :** Les commentaires multilignes peuvent être créés en utilisant des triple guillemets simples (''') ou des triple guillemets doubles (""") pour entourer le texte du commentaire.
- La ligne 2 est ce qu'on appelle une **fonction**. C'est la fonction qui sera executée au lancement du programme.

Documentation de fonctions

Les docstrings sont des chaînes de caractères placées immédiatement après la définition d'un élément et sont utilisées pour générer de la documentation automatique.

Vous pouvez ajouter des docstrings comme commentaires pour documenter des fonctions, des classes et des modules.

Hello world !

```
 1 def calculate_area(length, width):
 2     """
 3     Calculate the area of a rectangle.
 4
 5     :param length: The length of the rectangle.
 6     :type length: float
 7     :param width: The width of the rectangle.
 8     :type width: float
 9     :return: The area of the rectangle.
10   :rtype: float
11   """
12   area = length * width
13   return area
```

- Utilisez des noms de variables en minuscules avec des mots séparés par des underscores (ex `ma_variable`).
- Utilisez une indentation de quatre espaces pour organiser le code (fonctions, boucles, conditions etc).
- Utilisez des commentaires.
- Utilisez des docstrings pour documenter votre code.
- Regroupez les importations de modules en haut de votre fichier.
- Suivez le principe DRY (Don't Repeat Yourself).

function(void) VS fonction()

```
# Calculate the area of a rectangle

# Input the length and width of the rectangle
length = float(input("Enter the length: "))
width = float(input("Enter the width: "))

# Calculate the area
area = length * width

# Display the result
print("The area of the rectangle is:", area)
```

? Pouvez-vous comprendre ce qui se passe dans ce programme ?

print() et input()

print()

La fonction print() est utilisée pour afficher des messages, des variables, et d'autres informations à la console ou à la sortie standard.

```
print("valeur de x est : ",x) ;
```

input()

La fonction input() est utilisée pour recevoir des données d'entrée de l'utilisateur via la console. Elle permet à votre programme d'interagir avec l'utilisateur en lui demandant de fournir des informations.

```
user_input = input("Entrez votre nom : ")
```

print() et input()

print()

La fonction print() est utilisée pour afficher des messages, des variables, et d'autres informations à la console ou à la sortie standard.

```
print("valeur de x est : ",x) ;
```

input()

La fonction input() est utilisée pour recevoir des données d'entrée de l'utilisateur via la console. Elle permet à votre programme d'interagir avec l'utilisateur en lui demandant de fournir des informations.

```
user_input = input("Entrez votre nom : ")
```

Il est important de noter que input() renvoie toujours une chaîne de caractères !

Erreur et avertissement

Error

Error est suivi d'un message concernant la nature de l'erreur détectée. Ce qui signifie que la compilation a échoué en un point du programme source. L'erreur se situe donc avant ce point.

Warning

Warning est suivi d'un message qui est un avertissement. Le compilateur a réalisé le travail mais vous signale qu'il a détecté un problème potentiel quand vous exécuterez votre programme.

Votre code ne doit pas générer de message d'erreur !

Un code bien écrit n'a pas non plus de message « warning » !

Types d'erreurs

Il existe principalement cinq types d'erreurs dans la programmation :

- Syntax error
- Logical error
- Semantic error

- **Syntax error :** Ces erreurs sont principalement dues à des fautes de frappe ou au non-respect de la syntaxe du langage de programmation spécifié. Une telle erreur se produit si le point-virgule (;) est absent à la fin de la déclaration.
- **Logical error :** L'erreur logique est une erreur qui conduit à une sortie non souhaitée.
- **Semantic error :** Les erreurs sémantiques sont les erreurs qui se produisent lorsque les déclarations ne sont pas compréhensibles par le compilateur. L'utilisation d'une variable non initialisée produit une telle erreur.

C’est parti !

Essayez de créer un programme qui imprime vos informations (nom, prénom, date de naissance, numéro d’étudiant).

Variables

Une variable est un nom qui fait référence à une valeur en mémoire.

- **Déclaration :** En Python, les variables sont créées lorsque vous leur attribuez une valeur :

```
#declaration of the variable 'first_name'
first_name = "Revekka"
#declaration of the variable 'last_name'
last_name = "Kyriakoglou"
```

Variables

Une variable est un nom qui fait référence à une valeur en mémoire.

- **Déclaration :** En Python, les variables sont créées lorsque vous leur attribuez une valeur :

```
#declaration of the variable 'first_name'
first_name = "Revekka"
#declaration of the variable 'last_name'
last_name = "Kyriakoglou"
```

- **Utilisation :** Les variables peuvent être utilisés dans des opérations, des expressions ou des fonctions.

```
#declaration of the variable 'first_name'
first_name = "Revekka"

# concatenation of str
message = "Bonjour " + first_name

print(message)
```

Types de variables

- int (entier)
- float (nombre à virgule flottante)
- str (chaîne de caractères)
- bool (booléen)
- list (liste)
- tuple (tuple)
- dict (dictionnaire)
- set (ensemble)
- NoneType (None)

Types de variables

```python
age = 25 # int

weight = 59.5 # float

name = "Alice" # str

is_student = True # bool

grades_AP = [13, 15, 9, 17] #list

coordinates = (3.0, 4.0) # tuple

info = {"name": "Alice", "age": 25} #dict

courses = {"AP", "English", "Literature"} #set

comments = None
```

Casting (conversion de type)

Si on veut spécifier/changer le type d'une variable, on peut le faire à l'aide d'un casting. Voici comment effectuer des conversions de type (casting) en Python :

```
# Conversion vers un entier
x = 10.5
integer_x = int(x)

# Conversion vers un nombre a virgule
y = 10
float_y = float(y)

# Conversion vers une chaine de caracteres
z = 42
str_z = str(z)

# Conversion vers un booleen
w = 0
bool_w = bool(w)
```

Type d'une variable

Pour obtenir le type d'une variable nous utilisons la fonction `type()`.

```
age = 25
name = "Alice"
id = "257929"
print(type(age))
print(type(name))
print(type(id))
```

Python permet de :

- Attribuer des valeurs à plusieurs variables en une seule ligne.
- Donner la même valeur à plusieurs variables en une seule ligne.

.

```
student_1, student_2, student_2 = "Anne", "Ben", "Che"
print(student_1)
print(student_2)
print(student_3)

student_1 = student_2 = student_2 = "Anne"
print(student_1)
print(student_2)
print(student_3)
```