Sommaire

1 La programmation avec Python 3

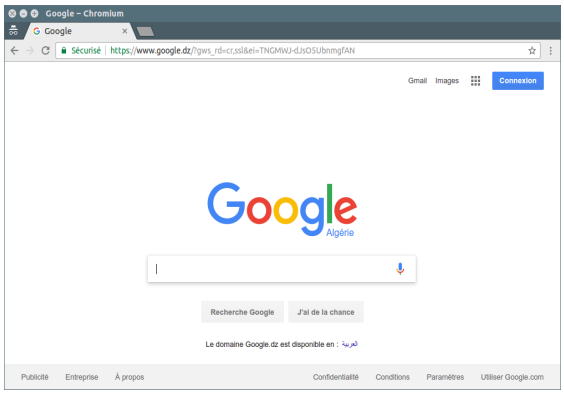
1.1 Introduction

Ceci est une petite introduction où on va parler des programmes et des langages de programmations.

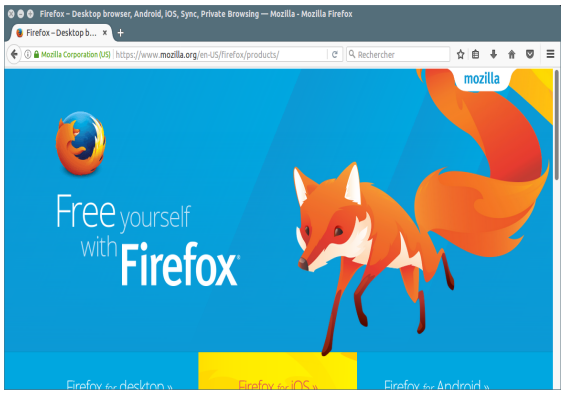
1.1.1 C'est quoi un programme ? ?

Je vais commencer par donner une définition qui peut vous semblez un peu floue, mais cette description va s'éclaircir coûte que coûte au cours de notre formation.

Alors, un programme est un ensemble d'instructions exécuté par l'ordinateur pour qu'il puisse (l'ordinateur) accomplir une tâche bien précise. On a pour exemple des programmes : les navigateurs web (comme chromium, chrome, firefox), qui permettent de visiter les cites et de chatter en ligne ; on a aussi les jeux vidéos, les logiciels de retouches de photos . . . etc



(a) Google Chrome



(b) Mozilla Firefox

FIGURE 2 – Le jeu Assassin's Creed

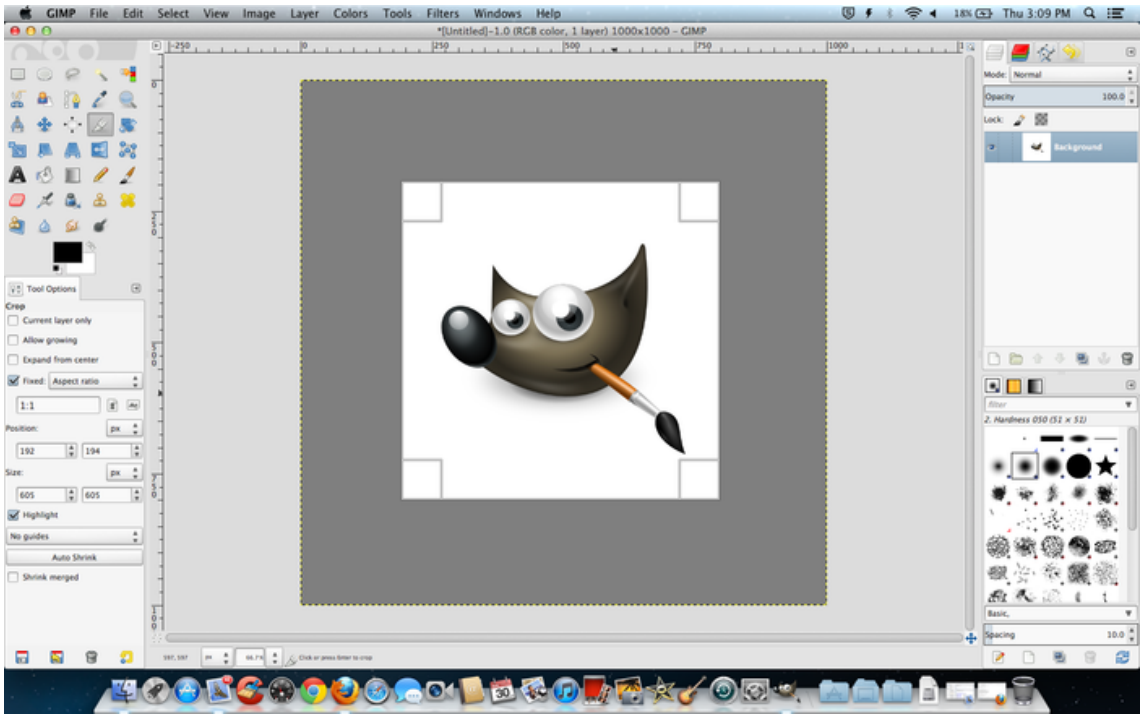


FIGURE 3 – Logiciels de retouches "Gimp"

1.1.2 Comment sont crée ces programmes ? ?

L'ordinateur est une machine bête et discipliné qui ne comprend qu'un seul langage, qui est le langage *binaire*. Ce langage est une succession de 1 et de 0 comme suit :

```
01101100001100110011000111101011101
```

Donc, normalement, pour créer des programmes, il fallait apprendre ce langage. Mais heureusement, les informaticiens ont eu la réflexion de créer d'autres langages intermédiaires (qu'on appel langage de programmation) qui sont beaucoup plus facile que le binaire. Ces langage ont tous le même but : permettre de créer des programmes plus facilement qu'en binaire.

Voici comment cela fonctionne :

- On donne nos instructions à l'ordinateur en utilisant un langage de programmation.
- Les instructions sont traduites en binaire grâce à un programme de traduction.
- L'ordinateur peut alors lire le binaire et faire ce qu'on l'a demandé.

Voici une image qui illustre ce qu'on vient de dire :

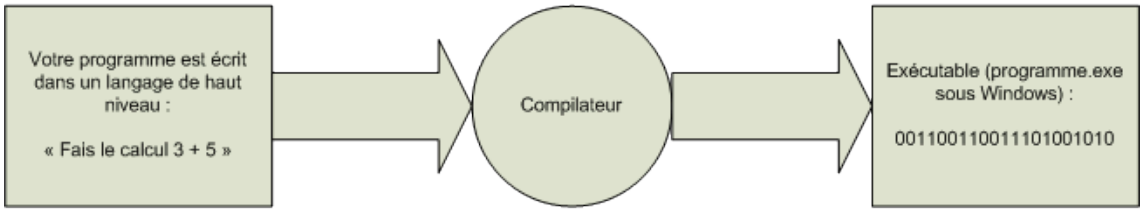


FIGURE 4 – Les étapes de création d'un programme

Une remarque très importante : les instructions que nous écrivons dans un langage de programmation sont appelé **Code Source**. Gardez bien ce terme à l'esprit, parce qu'on va beaucoup l'utiliser par la suite. Voici un exemple de code source python que nous écrirons plus tard :

```
#!/usr/bin/python3

import os
try:
    import psutil
except ImportError:
    import pip
    print("Installing missing packages (this will be done the first time only) :")
    if os.getuid() == 0:
        pip.main(["install", "psutil"])
    else:
        pip.main(["install", "psutil", "--user"])
    print("Done.\nRerun the command now.")
    exit(0)
import argparse
from time import sleep, localtime


# This function will execute until the end of the copy
def copy(time, verbose):
    while True:
        bytes_read = psutil.disk_io_counters().read_bytes / (1024*1024)
```

FIGURE 5 – Coude source Python

1.1.3 Les langages de programmation les plus connus

C++ : La formation est faite par YASSER.

Java : La formation est faite par WISSAM et SOFIANE.

Python : C'est la formation à laquelle vous assistez maintenant.

(a) Logo C++

(b) Logo Java

(c) Logo Python

1.1.4 Compilé vs Interprété

Nous avons dit tout à l'heure que les instructions sont traduites en binaire pour que l'ordinateur puisse les comprendre et les exécuter. Mais ce qu'on a pas dit c'est qu'il y a deux façons de traduire les instructions : *Compilation* et *Interprétation*.

Compilation : Avec ce processus, toutes nos instructions sont traduites en binaires à la fois. Ce qui nous donne en résultat un gros fichier binaire qui peut être exécuté directement par l'ordinateur.

Interprétation : Avec cette technique, on ne traduit pas tous les instructions en binaires, mais on va plutôt traduire une instruction par instruction à chaque fois qu'on veut exécuter notre programme. Par exemple, si mon code source contient 3 instructions, quand je veux l'exécuter je traduit la première instruction en binaire, et je la passe à l'ordinateur pour qu'il l'exécute ; ensuite je traduit la deuxième instructions, et je la passe à l'ordinateur pour qu'il l'exécute . . . et ainsi de suite jusqu'à la traduction de toute mes instructions.

Le programme qui fait l'interprétation est appelé l'*interpréteur*, et dans notre cas on va utiliser l'interpréteur *Python*.

1.1.5 Le langage Python

Python est un langage de programmation interprété, facile à apprendre, et très pratique. On cite quelques avantages de ce langage :

- Python est facile à apprendre et son code est facile à lire.
- Il multiplate-forme, il marche sous Linux, Windows, et Mac OS.

- C'est un langage orienté objet (on en parlera de ça dans le prochaines séances).
- Il a une bibliothèque tierce qui permet de faire de la programmation web d'une manière très flexible. Cette bibliothèque c'est *Django*.
- Il permet de développer très rapidement en utilisant peu de code, et de crée des applications très complexe avec facilité. Tous ça grâce à son style et à sa bibliothèque standard très complète.

Pour finir, il faut savoir que les géants de l'informatique comme Google, Yahoo, NASA préférent le langage python ; et ce langage est intégré à la quasi totalité des distributions Linux (C'est-à-dire que vous pouvez l'utiliser dans ces distributions sans devoir l'installer).

Juste pour information, il en existe deux versions de python qui ne sont pas compatibles entres eux. Ces deux versions sont les 2.x et les 3.x . Nous on va utiliser la version 3.x qui est le future de python.

1.2 Les logiciels nécessaires pour la programmation

Pour pouvoir créer des programmes en python, il existe deux manières :

1. Soit on écrit nos instructions dans un fichier texte, et puis on appel l'interpréteur python pour qu'il exécute nos instructions et nous affiche le résultat. Cette méthode comporte de nombreux avantages, et elle très flexible, mais elle est compliqué (surtout pour les débutants). Donc nous allons optez pour la deuxième méthode dans cette formation.
2. Cette méthode consiste à utiliser un programme appelé *IDE* (Integrated Development Environment), ou *EDI* en français (Environnement de Développement Intégré). Ce programme contient tous ce dont on a besoin pour créer nos programmes : une partie ou on peut écrire nos instructions, des boutons pour demander à python d'exécuter nos instructions, et finalement une partie ou le résultat de l'exécution est affiché.

L'IDE qu'on va utiliser est appelé *PyCharm* (Figure 7). C'est un IDE *Open Source* et très puissant.

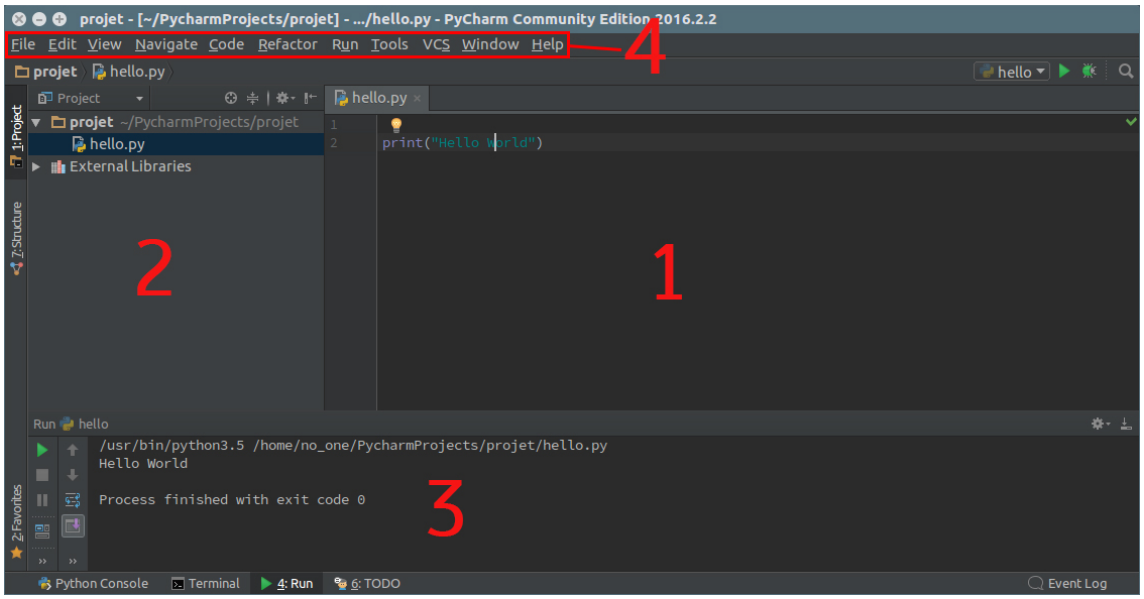


FIGURE 7 – Pycharm

PyCharm est composé essentiellement de 4 partie :

La zone principale (1) : c'est dans cette zone où on écrira notre code source (nos instructions).

La liste des fichiers du projet (2) : dans cette zone on peu voir tous les fichiers de notre projet. On voit bien que le projet présenté dans la Figure 7 contient un seul fichier seulement.

La zone d'affichage (3) : c'est cette zone qui nous affichera le résultat d'exécution de notre programme, où les éventuelles erreurs s'il y en a.

La barre d'outils (4) : elle contient de nombreux boutons qui font différentes choses : exécuter le projet, configurer l'IDE . . . etc, mais nous on utilisera que quelques boutons.

1.3 Hello World

Nous allons écrire maintenant notre premier programme. Un programme petit et simple pour comprendre le fonctionnement de ce langage fascinant.

On commence par créer un nouveau projet en cliquant sur la barre d'outils sur `File -> New Project`. Ensuite on choisis où on veut placer notre projet, et la version de l'interpréteur qu'on veut utiliser (Figure 8). Choisissez n'importe quelle version 3.x disponible sur votre ordinateur (Pour moi, j'utilise la version 3.5).

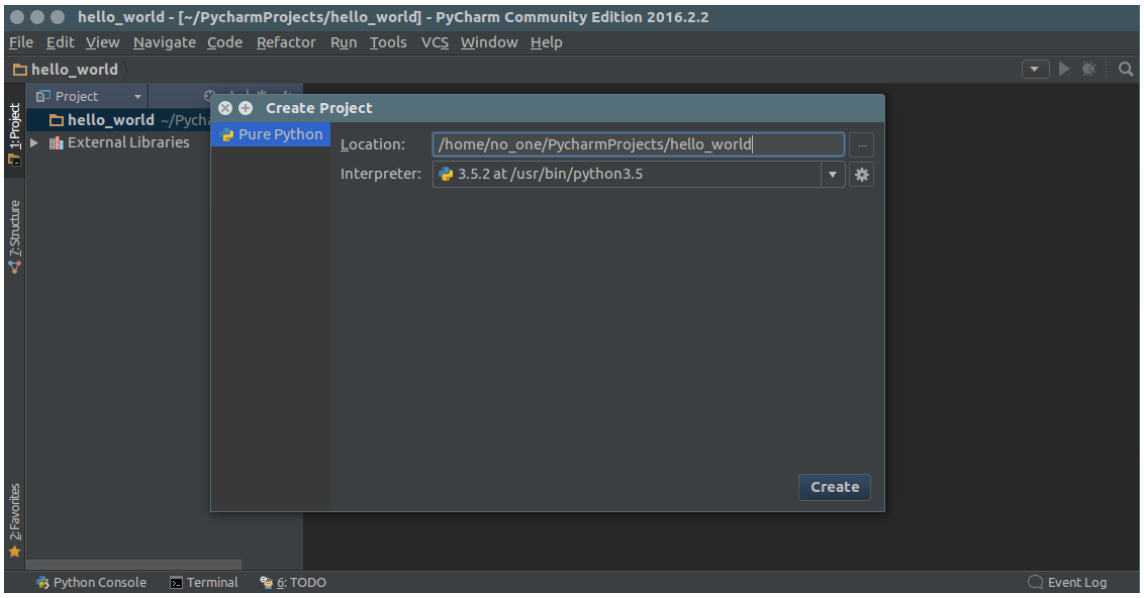


FIGURE 8 – Création d'un nouveau projet

On a crée notre projet, mais ce dernier est vide, il ne contient aucun fichier. Nous allons maintenant ajouter un fichier où nous allons écrire nos instructions (programmes). On fait un clique sur la barre d'outils sur `File -> New` et on choisit `Python File`. On lui donne un nom à notre fichier, et on clique sur `OK` (Figure 9).

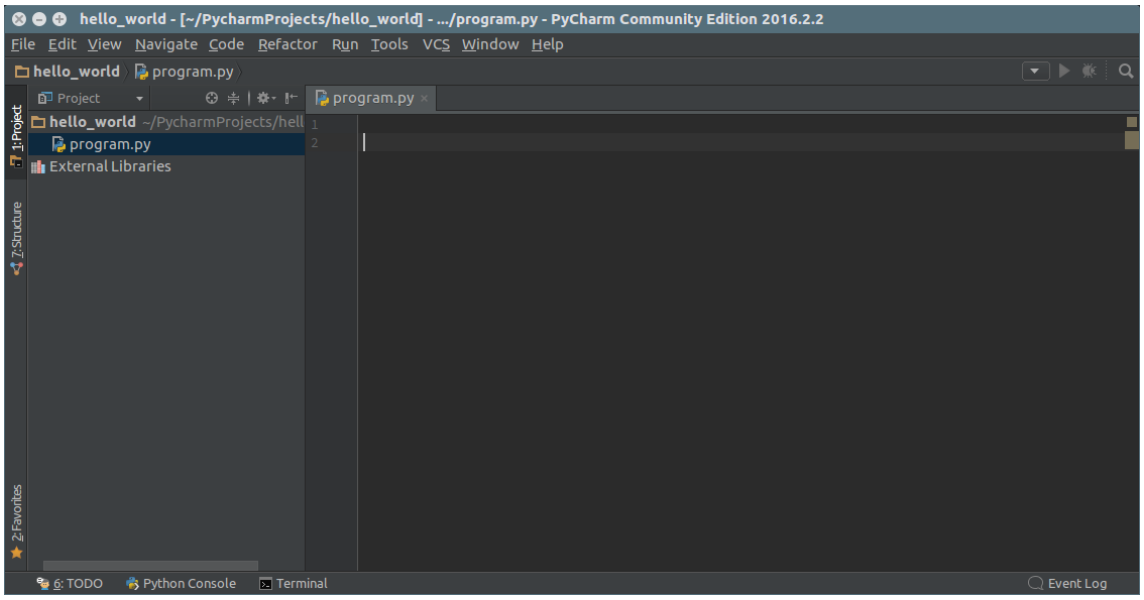


FIGURE 9 – Création d'un fichier pour écrire les instructions

Doré avant, je vais seulement vous affiché les codes que vous devez écrire dans la zone principale (la zone **1**, Figure 7).

On arrive maintenant à la programmation, voici votre premier programme :

```
1 print("Hello World!")
```

Écrivez ce code source dans la zone principale (zone n° 1 Figure 7), et demandez à PyCharm d'exécuter notre programme en cliquant sur la barre d'outils sur `Run -> Run`, et on choisit le fichier à exécuter (on en a un seul dans notre cas). Voici le résultat de l'exécution :

```
/usr/bin/python3.5 /home/no_one/PycharmProjects/hello_world/program.py
Hello World!

Process finished with exit code 0
```

Alors, la ligne que nous avons écrit dans l'IDE `print("Hello World!")` est une instruction qui demande à l'ordinateur d'afficher `Hello World!` à l'écran. Nous pouvons écrire n'importe quoi à la place de `Hello World!` et l'ordinateur se chargera d'afficher le message que nous avons demandé.

Avant de continuer, nous allons détaillé ce que `PyCharm` nous a affiché :

/usr/bin/python3.5 c'est le chemin vers l'interpréteur python que PyCharm a utilisé pour exécuté notre programme. La majorité des programmes sous *Linux* sont installés dans le dossier `/usr/bin/` .

/home/no_one/PycharmProjects/hello_world/program.py ça c'est notre fichier (là où on est entrain d'écrire notre code source). Donc d'après cette ligne, on comprend que `python3.5` est entrain d'exécuter ce fichier.

Hello World ! c'est le message que nous avons demandé à python d'affiché (on voit qu'il a bien obéi).

Process finished with exit code 0 ceci est un message affiché par `PyCharm` pour nous indiquer que notre programme s'est terminé avec succès.

Donc, `print()` est l'instruction responsable d'afficher un message à l'écran. On lui donne entre les `""` le message qu'on veut afficher. Essayez de Remplacer `Hello World!` par `Salut tout le monde` et ré-exécuté pour voir le nouveau résultat.

1.3.1 Les commentaires

Avant de terminer cette section avec un TP, nous allons parler des commentaires. Dans n'importe quel langage de programmation (y compris python), on a la possibilité d'écrire des commentaires.

Alors c'est quoi les commentaires ? Ce sont des bouts de textes qu'on écrit dans notre code source et qui permettrons d'expliquer le fonctionnement de notre code source. Ça peut vous semblez bizarre et inutile, mais sachez que c'est très commun qu'on revient à notre programme après quelques semaines pour l'améliorer ou pour le modifier, et sans les commentaires on sera obligé de tout repenser du début. Ne vous faites pas avoir par le fait que c'est vous qui a crée ce programme, car même si vous comprenez tous sur votre programme maintenant, vous oublierai après un certain temps.

Les commentaires ne serviront pas seulement à comprendre son code seulement, mais c'est très utile aussi quand vous travaillez en groupe pour crée un programme. Quand vous utilisez les commentaires, vous ne serrez pas obligé d'expliquer la partie que vous avez écrit à vos camarades.

J'insiste exprès sur le fait d'utiliser les commentaires, ils sont très très utiles et indispensables.

Alors, on sait ce que sont les commentaires, mais comment les utiliser avec `python` ? Voici un exemple sans plus tardé :

```
1 # Cette instructions affiche Hello World! a l'ecran.
2 print("Hello World!")
```

Les commentaires commence par un '#', tous ce qui suit ce caractère est ignoré par `python`. Donc on peut écrire autant de ligne de commentaire qu'on veut, il faut juste précédé chaque ligne avec un '#'.

1.3.2 Mini TP

Bon, on a assez bavardé, vous allez écrire votre vrai premier programme crée par vous même. Voici le TP :

- Créez un nouveau projet et surnommé le "Super projet 1".
- Créez un fichier dans ce projet où nous allons mettre notre code source. surnommé ce fichier "main".
- Débrouillez vous pour que le programme affiche :

```
Salut tout le monde.
On est dans une formation python, et on va cree des programmes super cool :D.
```

- N'oubliez pas d'utiliser les commentaires pour expliquer votre code.

1.3.3 Solution du TP

```
1 # Ces deux instructions affiche des messages a l'ecran.
2 print("Salut tout le monde.")
3 print("On est dans une formation python, et on va cree des programmes super
```

1.4 Les variables

Affichage, lecture des variables.

1.5 les opérateurs arithmétique

1.6 Les conditions

Explication des structures avec un exemple après chaque petite explication.

1.7 Les boucles

Explication des structures avec un exemple après chaque petite explication.

1.8 Découper le programme en fonctions

C'est quoi une fonction. Quelles sont les fonctions que nous avons déjà utilisés. Comment créer nos propres fonctions.

1.9 TP

1.10 Le help

Obtenir le help depuis la console.

2 La programmation orientée objet

2.1 C'est quoi déjà un objet ?

2.2 Les différents types d'objets disponible

On parlera des string, listes, tuples et des dictionnaires.

2.3 Les opérations sur les fichiers

Ouverture, lecture, écriture des fichiers.

2.4 Les chaînes de caractères

2.5 Les listes

2.6 Les tuples

2.7 Les dictionnaires

2.8 TP

3 Les bibliothèques standards

3.1 Exécutions des commandes systèmes

3.2 L'aléatoire

3.3 Gestion des mots de passes

3.4 Le réseau

4 Allez plus loin

Des tutos pour la programmation graphique avec PyQt. Le tuto d'openclassrooms.