PYTHON

# INTRODUCTION

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plateformes.

# Python 3.9

## SEMAINE 1

### S1j1=====================================================

## 1. Introduction

### 1.10 A quoi peut bien servir Python ?

#### 1.11 Il peut fonctionner sur un serveur pour créer des applications web

#### 1.12 Il peut être utilisé avec d’autres logiciels pour créer des applications d’entreprise

#### 1.13 Il peut se connecter à toutes sortes de bases de Données, SQL ou NoSQL, lire et modifier les fichiers

#### 1.14 Il peut être utilisé pour le traitement de Big Data

### 1.10 Python est un Langage interprété.

Il fonctionne dans un interpréteur qui permet l’exécution du code à la mesure qu’il est écrit

### 1.20 La bibliothèque Standard.

### 1.30 Les Interfaces graphiques

### 1.40 L’Environnement de développement intégré (IDE)

#### 1.4.1 Editeurs (Anaconda, PyCharm, Sublime Text, Wing, VS Code, ..)

### 1.50 Project based workflow

## 2. Mode d'emploi de l'interpréteur Python

### 2.10 Lancement de l'interpréteur

#### 2.11 Passage d'arguments

#### 2.12 Mode interactif

### 2.20 L'interpréteur et son environnement

#### 2.21 Encodage du code source

## 3. Sémantique du langage

### 3.10 Indentation [pas d'accolades]

### 3.20 Tout est objet

### 3.30 Commentaires

### 3.30 Appels de fonction et de méthodes d'un objet

### 3.50 Variables. Les variables sont des containers pour le stockage des valeurs des données

### 3.60 Typage dynamique. Références dynamiques

### 3.70 Attributs et méthodes

#### 3.71 ”Duck” typing

### S1j2=====================================================

### 3.80 Évaluation stricte des expressions versus évaluation différée (laziness)

(Les computations et les expressions sont évaluées immédiatement)

### 3.90 Objets mutables et immutables

## 4. Structure du langage

### 4.10 Types natifs

(Collection de types natifs pour le traitement de données numériques : strings, valeurs booléens, dates et temps)

#### 4.11 Nombres

#### 4.12 Casting

#### 4.13 Strings

#### 4.14 Booleans (True/ False)

#### 4.15 Opérateurs

#### 4.16 Listes

### 4.20 Utilisation de Python comme une calculatrice

#### 4.21 Nombres

#### 4.22 Chaînes de caractères

#### 4.23 Listes

#### 4.24 Premiers pas vers la programmation

### 4.30 Outils de contrôle des flux d’exécution

#### 4.31 Les instructions conditionnelles : *if, Elif et Else*

#### 4.32 Les instructions répétitives : Boucles (*for loops)*

##### Répétitions en boucle. L’instruction *While*

#### 4.33 La fonction *range ()*

#### 4.34 Les instructions *break, continue* et les clauses *else* au sein des boucles

#### 4.35 L'instruction *pass*

Passage de paramètres (*pass-by-value* et *pass-by-reference*)

### S1j3,4,5 – exercices, repetition ==================================

#### 4.37 *Range* et *XRange*

#### 4.38 Expressions ternaires

## 5. Structure des données

### 5.10 Les listes

#### 5.11 Utilisation des listes comme des piles

#### 5.12 Utilisation des listes comme des files

#### 5.13 Ajout et suppression d’éléments

#### 5.14 Concaténation et combinaison des listes

#### 5.15 Tri des listes

#### 5.16 Recherche binaire et maintien d’une liste triée

### 5.20 Tuples et séquences

#### 5.21 Emballage et déballage de plusieurs valeurs dans des Tuples

#### 5.22 Tuple nommé

### 5.20 L'instruction *del*

### 5.40 Fonctions de séquences intégrées

#### 5.41 Enumérer

#### 5.42 Trier

#### 5.43 Zip

#### 5.44 Inversion de l’ordre des listes : .*reverse() et reversed()*

### 5.50 Dictionnaires

#### 5.51 Créer des dictionnaires à partir de séquences

#### 5.52 Valeurs par défaut

#### 5.53 Types d’objets utilisés comme des clés de dict. valides (*hash function*)

### 5.60 Set (collection d’éléments uniques)

### 5.70 Compréhensions. Filtre des éléments d’une liste, d’un Set ou d’un Dictionnaire

### 5.80 Compréhension de lestes imbriquées

### 5.90 Comparer des séquences avec d'autres types

## SEMAINE 2

## 6.0 Les fonctions

### 6.10 Les fonctions sont des Objets

### 6.20 Définir une Fonction.

Name Spaces, Scope et Fonctions locales

### 6.30 Retour de valeurs multiples

### 6.40 Anonymous : les Fonctions Lambda

### 6.50 Fonctions que retournent d’autres Fonctions

### 6.60 Générateurs

#### 6.61 Générateurs d’expressions

### S2j1, 2=================================================

## 9. Classes et Objets

### 9.10 Objets et noms : préambule

### 9.20 Portées et espaces de nommage en Python

#### 9.21 Exemple de portées et d'espaces de nommage

### 9.30 Une première approche des classes

#### 9.31 Syntaxe de définition des classes

#### 9.32 Objets classes

#### 9.33 Objets instances

#### 9.34 Objets méthode

#### 9.35 Classes et variables d'instance

#### 4.36 Traitement des exceptions

#### 6.62 Module Itertools

## 6. Modules

### 6.10 Fonctionnement des modules

### 6.20 Les modules en détail

#### 6.21. Exécuter des modules comme des scripts

#### 6.22 Les dossiers de recherche de modules

#### 6.23 Fichiers Python « compilés »

### 6.30 Modules standards

### 6.40 La fonction *dir()*

### 6.50 Les paquets

#### 6.51 Importer depuis un paquet

#### 6.52. Références internes dans un paquet

#### 6.53 Paquets dans plusieurs dossiers

## 7. Les entrées/sorties

### 7.10 Formatage de données

#### 7.11 Les chaines de caractères formatées (f-strings)

#### 7.12 La méthode de chaine de caractères *format()*

#### 7.13 Formatage de chaînes à la main

#### 7.14 Anciennes méthodes de formatage de chaînes

### 7.20 Lecture et écriture de fichiers

#### 7.21 Méthodes des objets fichiers

#### 7.22 Sauvegarde de données structurées avec le module json

## 8. Erreurs et exceptions

### 8.10 Les erreurs de syntaxe

### 8.20 Exceptions

### 8.30 Gestion des exceptions

### 8.40 Déclencher des exceptions

### 8.50 Chaînage d'exceptions

### 8.60 Exceptions définies par l'utilisateur

### 8.70 Définition d'actions de nettoyage

### 8.80 Actions de nettoyage prédéfinies

## SEMAINE 3

## 9. Classes et Objets

### 9.10 Objets et noms : préambule

### 9.20 Portées et espaces de nommage en Python

#### 9.21 Exemple de portées et d'espaces de nommage

### 9.30 Une première approche des classes

#### 9.31 Syntaxe de définition des classes

#### 9.32 Objets classes

#### 9.33 Objets instances

#### 9.34 Objets méthode

#### 9.35 Classes et variables d'instance

### 9.40 Remarques diverses

### 9.50 Héritage

#### 9.51 Héritage multiple

### 9.60 Variables privées

### 9.70 Trucs et astuces

### 9.80 Itérateurs

### 9.90 Expressions et Générateurs

## 10. La bibliothèque standard

### 10.10 Interface avec le système d'exploitation (module *os*)

### 10.20 Jokers sur les noms de fichiers (module *glob*)

### 10.30 Paramètres passés en ligne de commande (module *sys, var argparse*)

### 10.40 Redirection de la sortie d'erreur et fin d'exécution (module *sys*)

### 10.50 Recherche de motifs dans les chaînes (module *re*)

### 10.60 Mathématiques (modules *math, random, statistics,* Projet SciPy)

### 10.70 Accès à internet (modules *urllib.request* et module *smtplib*)

### 10.80 Dates et heures (module datetime)

### 10.90 Compression de données (modules *zlib, gzip, bz2, zipfile et tartifle*)

### 10.100 Mesure des performances (module *timeit*)

### 10.110 Contrôle qualité (modules *doctest* et *unitest*)

### 10.120 Piles fournies (paquets email, json, sqlite3, ..)

## 11. Autres modules de la bibliothèque standard

### 11.10 Formatage de l'affichage (Modules *reprlib, textwrap, locale* )

### 11.20 Gabarits / Templates (Module *string*, classe *Template*)

### 11.30 Traitement des données binaires (Module *struct*)

### 11.40 Fils d'exécution (Module *threading*)

### 11.50 Journalisation (Module *logging*)

### 11.60 Références faibles

### 11.70 Outils pour les listes

#### 11.71 Le module *Array*

#### 11.72 Le module *Collections*

#### 11.73 Le module *Heapq*

### 11.80 Arithmétique décimale à virgule flottante

#### 11.81 Le module *Décimal*

## 12. Environnements virtuels et paquets

### 12.10 Introduction

### 12.20 Création d'environnements virtuels

### 12.30 Gestion des paquets avec pip

## 14. Édition interactive des entrées et substitution d'historique

### 14.10 Complétion automatique et édition de l'historique

### 14.20 Alternatives à l'interpréteur interactif

## 15. Arithmétique en nombres à virgule flottante : problèmes et limites

### 15.10 Erreurs de représentation

## 16. Annexe

### 16.10 Mode interactif

#### 16.11 Gestion des erreurs

#### 16.12 Scripts Python exécutables

#### 16.13 Configuration du mode interactif

## SEMAINE 4

## Développement Web avec Python

## 17. FLASK 2.0

### 17.10 Installation

#### 17.11 Installation de SQLite avec Flask

##### - Créer un nouveau document à la racine du projet *app.db*

#### 17.12 Installation de l’extension SQLAlchemy en utilisant pip.

Objet Relationel Mapping (ORM)

#### 17.13 Configuration du modèle de la BD (

##### - Structure de chaque table et associations entre les tables

##### - Définition des types pour chaque champ (entiers, strings, bolééns)

#### 17.14 Connexion de la base de Données avec Flask

#### 17.15 Insertion d’une nouvelle table dans la base de Données

### 17.20 Templates

#### 17.21 Le moteur de modèles Jinja (*render\_template*)

#### 17.22 Loops

#### 17.23 Héritage de Templates

### 17.30 Web Forms

#### 17.31 Introduction

#### 17.32 Flask WFT

#### 17.33 Formulaire de Logng Utilisateur

#### 17.34 Formulaires des Templates

#### 17.35 Formulaires de Vues

#### 17.36 Reception des Formulaires de Données

#### 17.37 Validation des Champs

#### 17.38 Création de Liens

### 17.40 Databases

#### 17.41 Les bases de données en Flask

#### 17.42 Migrations de dataBases

#### 17.43 Configuration de Flask-SQLAlchemy

#### 17.44 Modèles de bases de Données

#### 17.45 Création du répertoire de Migration

#### 17.46 Première Migration d’une Base de Données

#### 17.47 Flux des données (Upgrade et Downgrade)

#### 17.48 Relations entre bases de données

#### 17.49 Shell Context

### 17.50 Authentification des utilisateurs

#### 17.51 Password

#### 17.52 Flask-Loging

#### 17.53 Préparation del Modèle Utiisateur pour Flak-Loging

#### 17.54 Enregistrement des utilisateurs

#### 17.55 Loging des utilisateurs (in et out)

#### 17.56 Demande aux utilisateurs de s’authentifier

#### 17.57 Templates pour l’affichage du profil créé par l’utilisateur

#### 17.58 Enregistrement de l’utilisateur

### 17.50 Gestion des Erreurs

#### 17.51 Gestion des erreurs en Flask

#### 17.52 Mode de débogage

#### 17.53 Personnalisation de la page d’erreur

#### 17.54 Envoi d’un message de correction d’erreurs par mail

#### 17.55 Fixation de l’erreur d’un nom d’utilisateur dupliqué

### 17.60 Gestion des Abonnés

#### 17.61 Database Model Representation

#### 17.62 Ajout et suppression des abonnés

#### 17.63 Réception des ”posts” des utilisateurs abonnés

#### 17.64 Commination des ”posts” propres avec ceux des utilisateurs

#### 17.65 Unité de test du Modèle Utilisateur

#### 17.66 Intégration des Abonnés avec l’Application

### 17.70 Mise en Page

#### 17.71 Navigation

#### 17.72 Gestion d’un Blog

##### - Soumission de posts par les utilisateurs

##### - Publication des Posts

##### - Mise en page des Posts

#### 17.73 Mise en page des pages dans le profil des utilisateurs

### 17.80 Flask-Email

#### 