BUSINESS SCHOOL

Miia CHABOT

PYTHON

Pour le management, l'économie et la finance

+ EN LIGNE

OFFERT

- 110 QCM interactifs avec corrigés
- Fichiers de données et codes PYTHON

deboeck SUPÉRIEUR

BUSINESS SCHOOL

Miia Chabot

PYTHON

Pour le management, l'économie et la finance

deboeck SUPÉRIEUR

OFFERT : Ressources numériques

- **110 QCM interactifs avec corrigés (en fin de chaque chapitre)**
- **Fichiers de données et codes PYTHON à télécharger à l'adresse https://www.deboecksuperieur.com/site/348394**

Pour toute information sur notre fonds et les nouveautés dans votre domaine de spécialisation, consultez notre site web : **www.deboecksuperieur.com**

Rue du Bosquet 7, B-1348 Louvain-la-Neuve

Dépôt légal :
Bibliothèque nationale, Paris : janvier 2023 ISSN 1782-8147
Bibliothèque royale de Belgique, Bruxelles : 2023/13647/007 ISBN 978-2-8073-4839-4

SOMMAIRE

AVANT-PROPOS

Cet ouvrage est le fruit d'un travail d'enseignement engagé en 2014 auprès d'étudiants de master. Les évolutions du marché du travail depuis lors, et les attentes fortes des jeunes vis-à-vis de formations faisant sens tant dans leurs pratiques que dans leurs projets ont été un moteur formidable d'innovation pédagogique. La curiosité face aux changements et à ce qu'ils impliquent a été la première préoccupation des participants à ce cours. Après ces quelques années riches d'échanges et de réalisations, il m'a paru important de présenter dans un ouvrage quelques-uns des sujets qui ont alimenté ces réflexions communes.

1. OBJECTIFS DE L'OUVRAGE

Cet ouvrage n'est pas un manuel de syntaxe ou d'algorithmique Python. Il s'agit plutôt d'un manuel qui propose de découvrir Python au sein d'univers différents, tous en lien avec le management, l'économie et la finance. Si l'objectif est bien d'apprendre à manipuler le langage Python, il s'agit surtout de l'utiliser pour résoudre des questionnements concrets dans un monde où la gestion et l'analyse des données sont en pleine transformation. La vitesse à laquelle les informations sont disponibles, leur variété et leur volume appellent à changer les pratiques. Les outils traditionnels d'analyse manquent de flexibilité et ne permettent pas toujours d'obtenir une synthèse pertinente pour l'aide à la décision. L'objectif est ici de mobiliser Python dans les situations où il est le plus performant, c'est-à-dire le plus adapté pour traiter les données, les restructurer et rendre compte d'une situation. Les différentes situations étudiées dans ce livre s'appuient sur des contextes concrets que les étudiants ont eu à gérer dans le cadre de leur formation, de leur stage ou de leur alternance. Dans chacun de ces cas, Python a constitué un véritable atout.

2. À QUI S'ADRESSE L'OUVRAGE

Ce livre s'adresse aux étudiants de master en école de commerce. Il leur permet de manipuler Python dans des contextes qu'ils connaissent déjà, ce qui facilite la compréhension des outils mobilisés. Ce livre s'adresse également aux étudiants en école d'ingénieur. Ils y découvriront l'utilité de Python dans les domaines du management,

de l'économie et de la finance. De façon générale, cet ouvrage s'adresse aux personnes qui cherchent à se former à Python dans des situations concrètes.

3. CONTENU ET PLAN DE L'OUVRAGE

Chaque chapitre de l'ouvrage traite d'un sujet différent. Il est systématiquement accompagné d'un ou plusieurs fichiers Python et des bases de données utilisées afin que le lecteur puisse s'approprier les sujets traités. Ces fichiers sont disponibles sur le site[1] de l'ouvrage. Les thèmes traités sont les suivants :

- **Chapitre -1 : Introduction à Python**

  Ce chapitre accompagne le lecteur dans son premier contact avec Python. C'est l'occasion de découvrir comment l'utiliser et pourquoi ce langage est aussi performant. Une section présentant les tests de recrutements réalisés actuellement dans le secteur de la finance pour évaluer les compétences Python est aussi proposée. Des questionnaires d'entraînement sont ensuite mis à la disposition du lecteur à la fin de chaque chapitre.

- **Chapitre -2 : Évaluer le prix des options**

  Ce chapitre permet de travailler la syntaxe Python tout en élaborant des modèles d'évaluation du prix (*pricing*) de différentes options. Une mise en contexte de la révolution Black and Scholes s'accompagne de différents programmes Python. Le modèle utilisé par le London Metal Exchange, ainsi que le pricing de crypto-options font partie des pépites développées sous Python grâce au soutien des partenaires professionnels.

- **Chapitre -3 : Traiter des données de marché**

  Dans ce chapitre, on découvre où trouver la donnée de marché, et comment la traiter. Après une mise en contexte critique du trading algorithmique, différents outils d'analyse sont développés sous Python. La réalisation de visuels interactifs sous forme de pages html, l'élaboration de compteurs de vitesse, mais aussi le codage d'un algorithme sont proposés dans le but de montrer au lecteur comment traiter le signal-prix.

- **Chapitre -4 : Traiter des données d'économie et de gestion**

  Différents types de bases de données sont présentés afin d'illustrer la nature variée et parfois complexe des informations à traiter. Un guide des bonnes pratiques liste les réflexes importants à avoir. Trois problèmes distincts sont explorés dans ce chapitre : la fragmentation des données au sein d'un cloud, l'observation du réchauffement climatique, et la mesure du bonheur dans le monde.

1. https://www.deboecksuperieur.com/site/348394

- **Chapitre -5 : Cartographier des risques**

  Dans ce chapitre, une collaboration active avec des spécialistes de la gestion des risques a mené à la réalisation d'un programme Python permettant de construire des cartographies de risque focalisées sur le couple sévérité – impact de différentes catégories d'évènements. Ce chapitre se poursuit ensuite par une étude du risque crédit sur la base d'une démarche exploratoire. La dernière section traite d'un risque climatique extrême, à savoir le risque de tsunami.

4. REMERCIEMENTS

Cet ouvrage est l'occasion de remercier l'ensemble des partenaires professionnels qui m'ont aidée à développer ce cours. Les échanges entretenus lors des différentes LeX à Londres, le matériel professionnel transmis pour exemple, les feedbacks prodigués tant sur le cours que sur les modalités de recrutement de nos étudiants ont considérablement enrichi le message pédagogique. Ils ont ouvert les esprits et repoussé des barrières.

Je voudrais également remercier l'ensemble des promotions d'étudiants qui se sont plongées dans ces questions de programmation et qui ont fait preuve d'une persévérance et d'un dynamisme exceptionnels.

Chapitre 1
POURQUOI PYTHON EST SI IMPORTANT

Python existe depuis déjà plusieurs années, et il connaît aujourd'hui beaucoup de succès. Les raisons de ce succès sont largement documentées dans les médias. Il est simple, rapide, flexible, gratuit... et il apporte de nombreuses solutions à des problèmes parfois complexes. En vérité, il est loin d'être le seul langage de programmation capable d'autant de merveilles. Alors, pourquoi Python est-il si important ?

Qu'il s'agisse d'économie, de finance, de sciences de gestion au sens le plus large, ou encore du climat, la quantité d'information dont nous disposons désormais est telle que les outils traditionnellement utilisés pour l'analyse sont devenus trop lents et souvent inefficaces. Nous avons besoin d'être de plus en plus agiles, d'être en mesure d'accéder à l'information rapidement et de la traiter avec pertinence pour apporter des solutions concrètes. En finance, la gestion de portefeuille s'accommode peu des outils VBA – Excel : les univers d'investissements, toujours plus larges, nécessitent d'utiliser des bases de données très lourdes, et demandent des temps de calcul trop importants. En marketing, l'étude du comportement des consommateurs impose de prendre en compte les individus dans toute leur diversité de choix et d'actions. En « transaction services » (conseil), les analyses nécessitent de manipuler la donnée avec agilité et d'en évaluer la nature avec précision. L'élaboration d'outils de mesure et de suivi de la performance doit permettre de mettre en œuvre un pilotage plus efficace, plus innovant et toujours plus dynamique de l'entreprise. Il en va de même avec la gestion des risques et la cartographie de ces risques, qui évoluent sans cesse pour intégrer de nouvelles catégories de risques, comme le risque climatique dans toutes ses dimensions. Les champs d'application sont trop nombreux pour être listés ici.

Cette réalité vient bouleverser les pratiques et le 'data analytics' a fait son entrée dans à peu près toutes les formations du supérieur. Dans cette transition qui s'installe désormais, Python a pris l'avantage. Il n'a cependant d'intérêt que s'il est utilisé pour résoudre des questions concrètes, et c'est la philosophie de cet ouvrage. L'objet de ce premier chapitre est de présenter ce langage de programmation et son écosystème. S'il s'agit de votre premier contact avec Python, la troisième et la quatrième section de ce chapitre vous aideront à devenir un utilisateur et à vous familiariser avec la syntaxe. La dernière section permet de découvrir comment se préparer aux tests de recrutement utilisés dans le secteur de la finance, et offre la possibilité de s'exercer.

1. PYTHON, LANGAGE DE PROGRAMMATION

Python est un langage de programmation interprété, ce qui signifie que c'est un langage qui est lu ligne par ligne par un interpréteur. L'interpréteur est un programme qui va, ligne de code après ligne de code, lire les différentes instructions que vous aurez rédigées sur votre console. C'est la raison pour laquelle l'ordre dans lequel vous allez taper ces instructions est extrêmement important. Si les instructions sont mal organisées, l'interpréteur ne pourra pas lire, analyser et préparer vos instructions pour le processeur. L'interpréteur a pour objectif de convertir vos instructions pour qu'elles soient lisibles en langage machine. En effet, votre ordinateur ne peut pas exécuter du code Python sans ce langage machine. La conversion se termine lorsque l'ensemble des lignes de codes ont été interprétées. Elle peut néanmoins stopper en cours de route, car une erreur se sera glissée dans les instructions. C'est l'une des grandes qualités d'un langage de programmation interprété : la ligne de code où se trouve l'erreur est tout de suite identifiée, puisque l'affichage des erreurs se fait après chaque ligne. Une autre grande qualité de ce genre de langage est la vitesse de la 'traduction' en langage machine. Lorsqu'on programme en Python, on peut exécuter très facilement son script (l'autre mot pour programme), avoir un feedback immédiat sur ses éventuelles erreurs, les corriger et relancer l'exécution dans la foulée. Si l'on ne sait pas comment corriger ses erreurs, il suffit de faire une recherche sur le web. La communauté d'utilisateurs Python est grande et communique[1] activement au sujet des erreurs et/ou difficultés que l'on peut rencontrer lors de l'exécution d'un script.

Python est aussi un langage de programmation orienté objet (POO).

Lorsqu'on cherche à écrire un programme, on va d'abord passer en revue tous les objets dont on aura besoin pour l'exécuter. On les organise ensuite en classes d'objets. Un objet est donc ce que l'on appelle une instance de classe. Chaque objet aura ses propres attributs (ou variables) et on pourra lui appliquer différentes méthodes (fonctions). La programmation orientée objet permet de rédiger plus rapidement des codes complexes et facilite le développement rapide d'applications.

Ce langage de programmation est flexible, car il offre la possibilité d'utiliser plusieurs fichiers avec l'extension *.py*. Lorsqu'on travaille sur un fichier relativement long, on peut avoir intérêt à le découper. Par exemple, lorsqu'on développe un jeu, utiliser différents fichiers pour les sons, les visuels et la cinétique simplifie considérablement le travail. On utilise ensuite un fichier principal qui appellera les autres au moment où ils seront nécessaires. Ce genre de démarche facilite aussi beaucoup le travail en équipe ; lorsque plusieurs personnes développent du code et que chacun a ses spécialités, on va travailler sur un espace commun, mais au sein de fichiers *.py* dédiés.

Enfin, un grand avantage de Python repose sur sa bibliothèque de packages[2], disponible gratuitement, qui permet de réaliser des analyses très pointues intégrant

1. https://stackoverflow.com/questions/tagged/python
2. https://pypi.org/

les dernières innovations. La communauté Python publie régulièrement sur ces nouveautés[3], en partageant au passage des exemples de codes et de réalisations que l'on peut reproduire soi-même.

2. L'ÉCOSYSTÈME PYTHON

Même si Python a un petit coût d'entrée pour un parfait débutant en programmation, il ne s'adresse pas seulement aux développeurs professionnels.

Les développeurs professionnels trouvent dans Python tout ce dont ils peuvent avoir besoin pour créer efficacement de grandes applications. Ces types d'utilisateurs construisent généralement leurs propres packages. Ils ont élaboré leurs propres applications, les améliorent et les optimisent au fil du temps. Ils adaptent l'écosystème à leurs besoins spécifiques. Ces groupes d'utilisateurs se lancent souvent dans des projets collaboratifs de longue durée, au sein desquels ils prototypent rapidement de nouveaux codes, et où ils explorent en profondeur le potentiel de leurs idées.

Les développeurs occasionnels aiment quant à eux utiliser Python pour des problèmes spécifiques pour lesquels ils savent que ce langage est un véritable atout. Par exemple, en visitant la page de la galerie de matplotlib[4], ils vont copier différentes lignes de code et créer de belles visualisations de données. Ils adaptent le code à leurs besoins sans avoir à rédiger entièrement un nouveau script. Enfin, il existe aussi un dernier groupe important d'utilisateurs de Python : les débutants, qui commencent tout juste à programmer. L'objectif de cet ouvrage, si vous êtes débutant, est de vous permettre d'utiliser Python pour résoudre des problèmes concrets, en adaptant les codes qui vous seront fournis dans les chapitres à venir, ou en allant chercher en ligne d'autres solutions. Avant de se lancer dans nos premières lignes de codes, il reste cependant à présenter les principaux packages utilisés dans le domaine des sciences économiques et des sciences de gestion. Les packages les plus connus sont regroupés sous le nom de « scientific stack ». Cette « pile » regroupe notamment les paquets (packages) suivants :

- **NumPy**[5] **:** Il s'agit d'une bibliothèque qui permet d'effectuer toutes sortes de calculs numériques. Cette bibliothèque offre aussi la possibilité de gérer plus facilement des tableaux. De nombreuses fonctions pourront être utilisées pour travailler sur des objets tableau. NumPy est aussi suffisamment pointue pour vous permettre de stocker des données hétérogènes et de gérer les difficultés que ce genre de données peut impliquer au quotidien.
- **SciPy**[6] **:** Scipy regroupe différents sous-packages et fonctions qui vont permettre les calculs les plus utiles en science et en finance. C'est une boîte à outils de routines courantes de calculs numériques, et elle est regroupée par fonctions.

3. https://python-bloggers.com/
4. https://matplotlib.org/
5. https://numpy.org/
6. https://scipy.org/

Algèbre linéaire, optimisation, interpolation linéaire, intégration numérique, statistiques, traitement d'image : les fonctions sont nombreuses.

- **matplotlib :** Il s'agit du package de visualisation de données le plus populaire pour Python. Il offre la possibilité de faire des visualisations en 2D et en 3D. Il est considéré aujourd'hui comme l'alternative open source à MATLAB.

Dans les chapitres qui vont suivre, nous utiliserons plusieurs packages supplémentaires. Pour faciliter la compréhension et l'exécution des codes, ces derniers seront listés au sein des chapitres où ils sont utilisés.

3. DEVENIR UTILISATEUR PYTHON

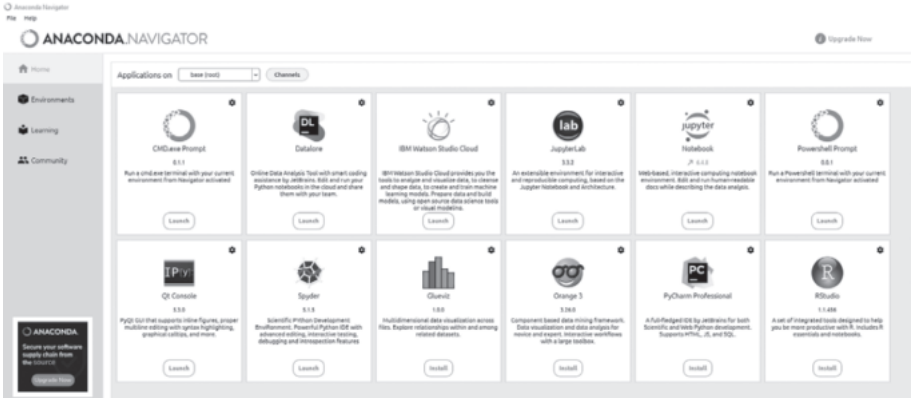
Pour un premier contact avec Python, on peut commencer par installer Anaconda[7], qui est assez populaire. Cet environnement de développement pour Python est libre et multiplateforme. Le navigateur Anaconda est une interface gratuite à partir de laquelle différentes applications sont disponibles. Comme le montre la figure 1, ce navigateur vous permet entre autres d'utiliser Spyder, Jupyter notebook ou encore JupyterLab. Spyder est idéal pour les projets de taille intermédiaire qui nécessitent d'utiliser de la donnée. On va pouvoir combiner la gestion d'un projet de développement et l'exploration des données.

Jupyter est quant à lui un projet open-source qui se compose de deux interfaces. La première, le notebook, permet de stocker son code. La seconde, le lab, est un environnement de développement. Jupyter sert surtout à l'exploration et à la visualisation de données et ne peut servir qu'en complément. De nombreuses autres applications sont disponibles. PyCharm est un outil de développement Python plus adapté à la gestion de projets complexes et d'envergure.

Les codes et les exemples qui sont développés dans cet ouvrage ont été réalisés sur Spyder 5.1.5 et Python 3.9.12 , qui sont les versions disponibles sur Anaconda au moment où s'écrit cet ouvrage.

7. https://www.anaconda.com/products/distribution. Pour davantage d'aide sur l'installation d'Anaconda, consultez la documentation: https://docs.anaconda.com/anaconda/install/

Figure 1. Navigateur Anaconda

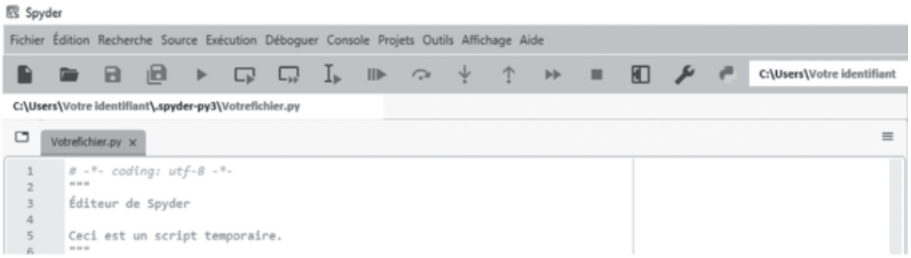


Commençons par découvrir l'interface de développement Spyder en le lançant depuis le navigateur Anaconda (il est possible de mettre Spyder directement en raccourci sur votre bureau). À l'ouverture, on trouve une fenêtre pour le code, une fenêtre pour la console, et un explorateur de variables. La fenêtre en haut à gauche (figure 2) est celle où vous allez rédiger vos lignes de codes. C'est ici que vous développerez vos programmes (scripts). Le chemin d'accès où trouver votre fichier *.py* est précisé dès le départ.

Le premier onglet qui vous accompagnera dans votre prise en main de Python est l'onglet d'aide. Dans cet onglet, vous pouvez commencer avec une visite interactive guidée de l'ensemble de votre espace de travail. Un premier contact de bienvenue s'accompagne de présentations ciblées sur les principaux outils à connaître tels que l'éditeur de code, la console, l'explorateur de variables et bien d'autres choses encore. Ce guide est idéal pour bien comprendre le fonctionnement de ces outils dans Spyder.

L'aide propose aussi un accès à une documentation en ligne qui s'accompagne d'un blog et d'un open chat où poser ses questions en cas de besoin. On retrouve également un lien vers différentes vidéos de présentations et de conseils. L'onglet se termine avec des tutoriels qui ont pour but d'accompagner l'utilisateur dans l'écriture de ses premières lignes de code, avec l'incontournable premier exemple « Hello World ». Un sommaire des raccourcis offre la possibilité aux plus aguerris de gagner en rapidité au moment de la rédaction de leur code.

Figure 2. Spyder : L'éditeur de code



Une fois Spyder installé, l'aide disponible facilite donc énormément la prise en main. À vous de vous approprier cet environnement de développement interactif en naviguant au sein des différentes fenêtres. Une fois que vous vous sentirez à l'aise avec Spyder, vous pourrez vous lancer dans la réalisation de votre premier fichier. *Votrefichier.py* est l'exemple sur lequel nous allons travailler au sein de ce chapitre. Il est disponible en téléchargement sur le site de De Boeck Supérieur[8], mais l'intérêt consiste surtout à s'entraîner à le rédiger vous-même, en vous appuyant sur la démarche pas-à-pas proposée ici.

3.1 *PREMIER CODE*

Lorsque vous avez lancé Spyder, il a ouvert automatiquement un fichier temporaire : *temp.py*. Ce dernier a déjà été renommé dans la figure 2. Pour faire de même, passez par l'onglet Fichier, et enregistrez votre fichier avec le nom de votre choix. Dans l'éditeur de code, la première ligne (1) qui apparaît en gris vous informe du type de codage utilisé :

 # -- coding: utf-8 -*-*

UTF 8 est l'abréviation pour « Universal Character Set Transformation Format ». Ce codage de caractères est le plus populaire à l'heure actuelle. Il est compatible avec les standards Unicode, d'Internet et la norme internationale ISO/CEI 10646. Pour autant, cette ligne n'est pas une ligne de code. Il s'agit juste d'une information pour l'utilisateur. En effet, lorsqu'on écrit autre chose que du code dans un *fichier.py*, pour éviter de générer des erreurs, on utilise systématiquement le caractère #. La machine ne lira pas cette ligne de votre programme. À chaque nouveau fichier, Spyder ajoutera cette première ligne, ainsi que l'encadré qui couvre les lignes 2 à 6. Ces lignes ont pour but de présenter le contenu du script.

C'est en quelque sorte un petit résumé de ce qui vient ensuite. Commençons par le modifier en changeant le titre et en ajoutant quelques informations sur ce que nous allons développer :

8. Pour rappel, l'adresse est https://www.deboecksuperieur.com/site/348394

Encadré 1. Présenter le contenu de son script

```
"""
Premier Script Python
1. Devenir utilisateur
2. Premier contact
"""
```

Rédigeons nos premières lignes de codes. Nous allons créer une fonction *devenir_utilisateur*, qui aura pour objectif de renvoyer le message texte suivant : «Premier objectif».

Encadré 2. Première fonction Python

```
def devenir_utilisateur():
    print("Premier objectif")

devenir_utilisateur()
```

def sert à créer notre fonction. *print* est l'action associée à cette fonction. Pour que l'interpréteur puisse faire son travail correctement, les lignes de codes doivent respecter cet ordre et cette mise en forme. Si vous alignez *print* avec *def*, la machine va renvoyer un message d'erreur (Indentation Error) sur *print*. Il faut en effet décaler *print* pour qu'elle soit bien associée à la fonction *devenir_utilisateur*. La dernière ligne appelle la fonction, ce qui va pousser la machine à exécuter l'action qui lui est demandée. Spyder va donc afficher « Premier objectif ». Utiliser une fonction n'est cependant pas nécessaire dans le cas d'un exemple aussi simple. On peut en effet juste taper *print("Premier objectif")*, ce qui donnera le même résultat. L'intérêt d'une fonction, comme nous le verrons dans le deuxième chapitre, est de réaliser des calculs élaborés. On peut ajouter des arguments et détailler un calcul complexe en plusieurs lignes. Voici, pour commencer, un exemple simple de soustraction entre a et b :

Encadré 3. Calcul d'écart

```
def ecart(a,b):
    c= a-b
    return c

ecart(4,2)
```

Dans cet exemple, *def* est enrichi d'un *return c* qui indique à la machine qu'elle doit retourner la valeur de c tel que définie au-dessus. La fonction dispose de deux arguments, a et b. Au moment d'appeler le résultat, on va renseigner les valeurs de a (4) et de b (2). Spyder fera le calcul et affichera 2.

La figure 3 présente les résultats tels qu'ils devraient apparaître sur votre console une fois les lignes exécutées. Pour lancer des lignes de codes depuis l'éditeur, il suffit de les sélectionner et de cliquer sur l'icône  (exécuter la sélection ou la ligne courante) :

Figure 3. Console Python

```
Python 3.9.12 (main, Apr 4 2022, 05 :22 :27)[MSC V.1916 64 bit (AM064)]
Type "copyright", "credits" or "license" for more information.

IPython 8.2.0 – An enhanced Interactive Python.

In [1]:
   ...:def devenir_utilisateur():
   ...:   print("Premier objectif")
   ...:
   ...:devenir_utilisateur()
Premier objectif

In [2]:
   ...:print("Premier objectif")
Premier objectif

In [3]:def ecart(a,b):
   ...:   c=a-b
   ...:   return c
   ...:
   ...:ecart(4,2)
Out[3]:2
```

3.2 *PREMIER PACKAGE*

Les exemples présentés jusqu'à maintenant n'ont pas nécessité d'utiliser de packages particuliers. La plupart du temps, ils sont cependant nécessaires. Certains packages, lorsque vous travaillez sur Anaconda, sont déjà pré-installés. Il suffit alors de les importer pour les utiliser. Pour d'autres, il faut commencer par les installer.

Encadré 4. Installer un package

```
!pip install [LE NOM DU PACKAGE]
```

Une fois installé, le package doit être importé pour que la machine soit en mesure d'exécuter les lignes de code. Prenons l'exemple de NumPy, qui est un package très populaire de calculs numériques. L'import de ce package se fait d'une façon assez simple sur Python. On va au passage le renommer (np), pour qu'il soit ensuite plus

facile de l'appeler là où on aura besoin de lui. Dans l'encadré qui suit, on cherche à calculer log (2) :

Encadré 5. Importer et utiliser un package

```
import numpy as np
np.log(2)
```

Le package a été renommé *np*. Lors de l'exécution du calcul, on utilise l'abréviation *np* pour indiquer à Spyder qu'il a besoin de cet outil pour faire son log.

Les abréviations utilisées pour les packages les plus connus le sont aussi. Nous connaissons désormais *np* pour NumPy, il y a *sy* pour sympy, *pdr* pour Pandas_datareader, ou encore *plt* pour matplotlib.pyplot, etc. Dans les prochains chapitres, les packages nécessaires aux différentes analyses seront systématiquement listés au démarrage des scripts.

4. ÉLÉMENTS DE SYNTAXE PYTHON EN SITUATION CONCRÈTE

Python n'est pas qu'une calculatrice. C'est aussi un langage de programmation riche, et polyvalent. À l'heure actuelle, il existe de très nombreux supports de cours et d'exercices en ligne pour étudier la syntaxe de ce langage. Dans cette section, quelques bases seront présentées rapidement. L'objet de cet ouvrage n'est pas de rédiger un précis d'algorithmique ou un aide-mémoire syntaxique. Il est néanmoins important de comprendre la logique de Python pour pouvoir bien appréhender les différentes applications qui seront proposées par la suite.

4.1 *VARIABLES ET VALEURS*

Savoir comment Python utilise différents types de données pour traiter des opérations générales est essentiel. Les données, dites données d'entrées acceptées par le langage, vont lui permettre de définir, de déclarer, de stocker et d'exécuter des valeurs, des opérations mathématiques, et des opérations logiques. Lorsque nous écrivons un code, on définit une variable avec son nom. Un langage de programmation, lui, a besoin de passer par trois étapes : d'abord, il faut définir le type de la variable, puis allouer un espace réservé en mémoire à cette variable, et enfin y assigner une valeur. Ce processus sur Python est beaucoup plus rapide, il fait en quelque sorte un trois en un.

Encadré 6. Variable

```
a=4
a
Out[1]: 4
```

Dans l'encadré précédent, Python a compris que *a* était un entier, a réservé un espace pour *a* et lui a assigné sa valeur. Si vous avez tapé le contenu de l'encadré dans Spyder, vous pouvez vérifier que l'identification s'est bien passée en consultant l'explorateur de variables dans la fenêtre en haut à droite. On y trouvera *a*, son type, sa taille et sa valeur. Spyder va garder en mémoire vos variables, et la valeur assignée. Tant que vous ne changerez pas la valeur de *a*, elle restera égale à 4.

Les différents types de données sur Python sont les entiers (*int*), les nombres décimaux (*float*), les caractères (*str*), les listes (*list*), les nombres complexes (*complex*), et les booléens (*bool*). Dans l'encadré 3, c est une variable *int*. En plus de l'explorateur de variables disponible sur Spyder, Python est capable de vous renseigner sur le type de données que vous avez entré (encadré 7).

Encadré 7. Types de données

```
b = 2
type(b)
Out[3]: int

d= b+1
type(d)
Out[4]: int

e= 1 + 6j
type(e)
Out[5]: complex

g = 1.5
type(g)
Out[6]: float

h= 'exemples'
type(h)
Out[7]: str

k=True
type(k)
Out[8]: bool

Maliste= ["Chapitre 1", "types", 256, 12.5, "etc."]
type(Maliste)
Out[9]: list
```

Attention au cas particulier de la variable *e*. Sur Python, pour les nombres complexes, on utilise toujours *j* pour désigner la partie imaginaire. *i* est par ailleurs à éviter, car il sert plus dans les boucles. Il y a, comme pour tout langage de programmation, des conventions d'écriture. On ne peut pas utiliser les mots *print*, *range*, *from*, *for*, etc., pour des noms de variables. Il faut également faire attention aux majuscules et aux minuscules : truc, Truc et truC sont trois variables différentes pour Python.

Travailler sur des données économiques, financières et de gestion implique aussi parfois de devoir convertir certaines valeurs. C'est très simple sur Python. Dans l'exemple qui suit, on va transformer *b* en variable caractère, puis en nombre décimal.

Encadré 8. Conversion de types

```
b=2
str(b)
Out[10]: '2'

float(b)
Out[11]: 2.0
```

Imaginons maintenant que dans un texte, vous avez besoin d'extraire la deuxième lettre du premier mot de la première phrase. Ou encore, qu'il faille extraire les trois dernières lettres du premier mot de cette même phrase, etc. Vous pouvez aussi avoir besoin de concaténer des séries de texte pour construire une analyse. Ceci peut paraître très simple et parfaitement inutile, mais en fait, c'est tout le contraire. En effet, dans certaines catégories de métiers comme l'audit, l'expertise, le conseil, le rating, etc., on doit traiter de grandes quantités d'information qualitative. Les outils d'intelligence artificielle sont pour cela très utiles, puisqu'ils vont nous aider à élaborer des tris, et à mettre en forme des extractions de données qui constitueront une bonne base de travail pour élaborer un diagnostic. Prenons par exemple le cas d'une agence de rating qui souhaite éplucher les rapports annuels d'un panel d'entreprises, dans le but d'évaluer leur capacité à communiquer sur leur exposition ESG. On va établir des mots clés, ou indiquer à la machine où trouver cette information dans la rubrique concernée. La machine sera alors en mesure de faire cette analyse sur l'ensemble des documents fournis, et de retourner une concaténation de résultats dans le format qui lui sera demandé (tableau, texte, nuage de mots graphiques, etc.). Python est non seulement capable de réaliser cela en un temps record, mais en plus sur une quantité exceptionnelle d'information. Une requête classique sur Excel, avec VBA, ne sera pas capable de suivre. Voici un petit exemple simple, à titre très préliminaire, pour illustrer ces propos : dans l'encadré 9, on assigne non pas un texte ou un lien Internet vers un texte, mais simplement le mot 'Audit' à la variable *s*. On demande ensuite à Python d'en extraire la première lettre (0), la deuxième (1), et le bloc de lettres 2 à 6. Ensuite, on crée une autre variable qu'on va concaténer avec la précédente. Python affiche les résultats dans l'ordre où sont rédigées les instructions, à l'issue du programme, après l'instruction *print*.

Encadré 9. Extraire de l'information qualitative : Audit !

```
s = 'Audit!'
print (s[0])
print (s[1])
print (s[2:6])
s1 = 'Réaliser un'
print (s1 +s)

A
u
dit!
Réaliser un Audit!
```

Évidemment, pour travailler sur de grosses quantités d'information, on ne va pas écrire nous-mêmes dans Python le contenu du texte à traiter.

On peut, par exemple, lui donner le chemin d'accès où trouver la donnée qui sera stockée en interne. Ou alors, et c'est tout de même beaucoup plus simple, lui dire à quel endroit trouver l'information qui est publiée en ligne par l'entreprise. La publication des rapports annuels se fait régulièrement et ces rapports sont accessibles publiquement. Python est capable d'aller chercher toutes sortes d'informations, à condition qu'on lui dise à quel endroit la prendre, dans quelle quantité et quoi. Et si l'information est payante, il existe aussi des solutions. Des fournisseurs de données comme Reuters – Refinitiv – Eikon ont très bien compris l'utilité de Python, et fournissent directement à leurs abonnés un API qui leur permet de downloader la donnée depuis leur site, sur Python. Ils fournissent même les codes Python pour réaliser les analyses. La qualité des supports, des contenus et donc du diagnostic est considérablement améliorée. Le temps que l'on passe à réaliser ce genre d'analyse est franchement réduit, et pour un rendu que l'on peut personnaliser très facilement. Autrement dit, fini le copier-coller des documentations techniques privées réservées à l'usage professionnel, fini la rédaction « artisanale » de rapports – type sur Word ou autre. Python le fait pour vous et automatisera le process aussi souvent que vous le souhaiterez. La communauté des développeurs Python met par ailleurs régulièrement à disposition en ligne des exemples de codes en accès libre, pour que l'on puisse créer son propre programme aux couleurs de son entreprise.

Dans l'encadré qui suit, voici comment gérer des listes d'informations. Encore une fois, cela n'a d'intérêt que si vous ne tapez pas le contenu de la liste vous-même, mais pour illustrer le fonctionnement de l'outil ici, c'est nécessaire.

BUSINESS SCHOOL

Miia Chabot est professeure de finance HDR à l'Essca. Elle enseigne la programmation sous Python et sous R appliquée à la finance, la gestion de portefeuille, la finance de marché, ainsi que l'utilisation avancée de Bloomberg et de Refinitiv auprès d'étudiants en master et en doctorat.

Elle a publié dans de nombreuses revues scientifiques : *Revue Économique*, le *Journal of Business Research*, la *Harvard Business Review* ou la *revue Finance*. Spécialiste de la gestion des risques, ses travaux portent sur les risques systémiques financiers, l'analyse des réseaux et les risques climatiques.

Boostez vos compétences Python

Ce manuel, destiné aux étudiants en économie et gestion débutant sur Python, aux étudiants ingénieurs en spécialisation finance et plus largement à la formation continue, propose :

- Une introduction à Python et à son univers ;
- Les modèles d'évaluation du prix des options (Black et Scholes, crypto-options) ;
- Le traitement des données de marché (analyse technique, trading algorithmique, compteurs de vitesse et robots) ;
- Le traitement des données économiques, comptables et financières (big data, fragmentation des données, changement climatique, économie post-covid) ;
- La réalisation de cartographies et d'analyses de risques (fréquence-sévérité, risque crédit, risques climatiques extrêmes).

Le livre propose différents parcours de lecture, des conseils, exercices corrigés et outils d'évaluation. Une préparation aux tests de recrutement réalisés dans le secteur de la finance pour évaluer les compétences Python est également disponible. Le lecteur dispose ainsi d'une véritable boîte à outils pour ses futures missions en entreprise.

RESSOURCES NUMÉRIQUES

- Pour chaque thème traité, un accès offert aux fichiers de données et codes Python est à disposition du lecteur.
- Des QCM en fin de chapitre pour évaluer vos connaissances et préparer vos tests de recrutement dans le secteur de la finance.

ISSN 1782-8147
ISBN 978-2-8073-4839-4

9 782807 348394

deboeck SUPÉRIEUR

www.deboecksuperieur.com