2_PREMIER PAS EN PYTHON
INTRODUCTION AU TRAITEMENT DES DONNÉES

SOMMAIRE :

I VUE D'ENSEMBLE DES LOGICIELS :

Les logiciels Bureautique à interface graphique :

- Excel
- Les logiciels généralistes à interface graphique :
- SAS
- SPSS
- Tanagra
- Statistica
- RapidMIner

Les logiciels généralistes en ligne de commande :

- R
- Python
- Les logiciels spécialisés :
- Requêtage SQL : Oracle SQL Developper, PL/SQL Developper
- Requêtage Hadoop : Hive, Pig
- Visualisation : D3JS

POURQUOI ET COMMENT PYTHON ?

Ses caractéristiques :

- Langage de programmation généraliste avec un écosystème scientifique.
- Open source et donc en constante évolution.
- Forte communauté et donc facilité à trouver de l'aide sur les forums.
- Cadre unifié pour mettre en œuvre tout le processus de l'analyse données.
- Langage interprété et non compilable -> On peut traiter le programaamme ligne par ligne.

Python en mode interactif

Peut s'utiliser comme une calculatrice (très améliorée).
On lance python dans un shell.
Le prompteur >>> python apparaît.
Puis on "discute" avec python en tapant des commandes oui instructions.

Les scripts

Un programme est une séquence d'instructions.
Dans le cas d'un programme en langage Python, on parle souvent de script Python.
Un script se présente sous la forme d'un fichier texte avec l'extension .py

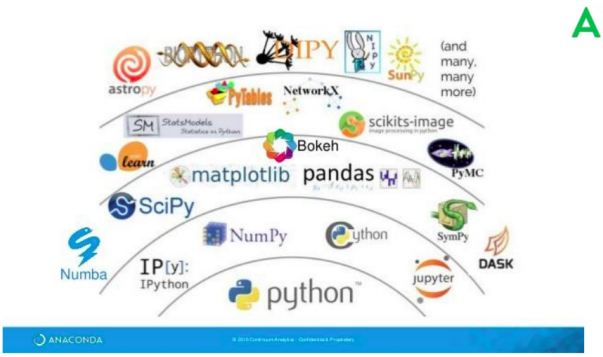
Les bibliothèques sous python :

- Une des grandes forces du langage Python réside dans le nombre important de bibliothèques logicielles externes disponibles.
- Une bibliothèque est un ensemble de fonctions. Celles-ci sont regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.
- Une bibliothèques doit être importé pour être utilisée.

• NumPy Module pour la manipulation de matrices, tableaux multidimensionnels et fonctions mathématiques opérant sur ces tableaux.
• SciPy Module pour l'optimsation, l'algèbre linéaire, les statistiques, traitement d'image, ...
• Pandas Module permettant la manipulation et l'analyse de données, notamment des données numériques et des série temporelle -> DataFrame, Panels, …
• Matplotlib Permet de tracer et visualiser des données sous forme de graph
• Scikit-learn Destiné à l'apprentissage automatique -> Forêts aléatoires, régression logistiques, classement, …
• Tensorflow Apprentissage automatique pour intelligence Artificielle

UN SERPENT PEUT EN CACHER UN AUTRE :

Anaconda installe Python avec une multitude de bibliothèques



ANACONDA

POURQUOI PYTHON ?

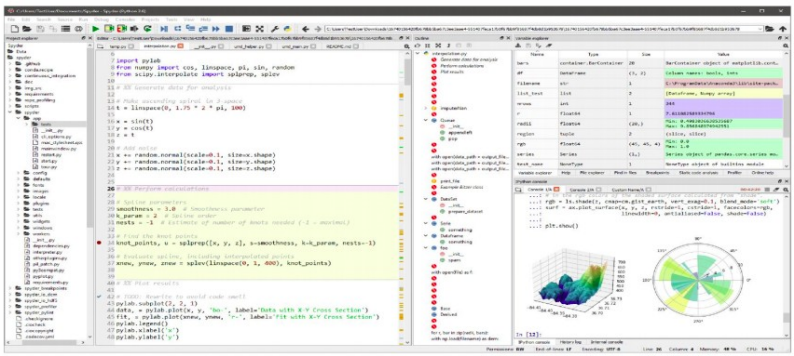
Les librairie sous python :

- SciPy
- NumPy
- Pandas
- Scikit-learn
- Tensorflow

ANACONDA

Interface de développement :

SPYDER
The Scientific Python Development Environment



INTRODUCTION JUPYTER NOTEBOOK :

Application web qui permet aux utilisateurs d'interagir avec le code et de transformer le navigateur web en terminal interactif et en logiciel de traitement de texte en même temps.

- Programmer dans le navigateur
- Le code, les instructions et la sortie son afficher en ligne
- Utile pour écrire un code qui raconte une histoire
- Utile pour réaliser des compte rendue
- Utiliser par les scientifiques et les chercheurs

Pour pouvoir utiliser Jupyter, il est judicieux d'installer la distribution Anaconda. En gros, une distribution, c'est un langage de programmation + certaines librairies et autres fonctionnalités.

Anaconda est donc une distribution Python, faite pour la Data Science.

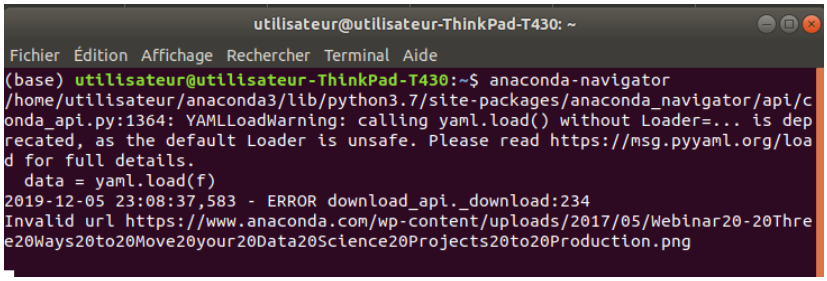
Il installera donc :

- Python
- les librairies de Data Science dont nous aurons besoin : Matplotlib, Scipy, Numpy, Pandas
- Le notebook Jupyter.

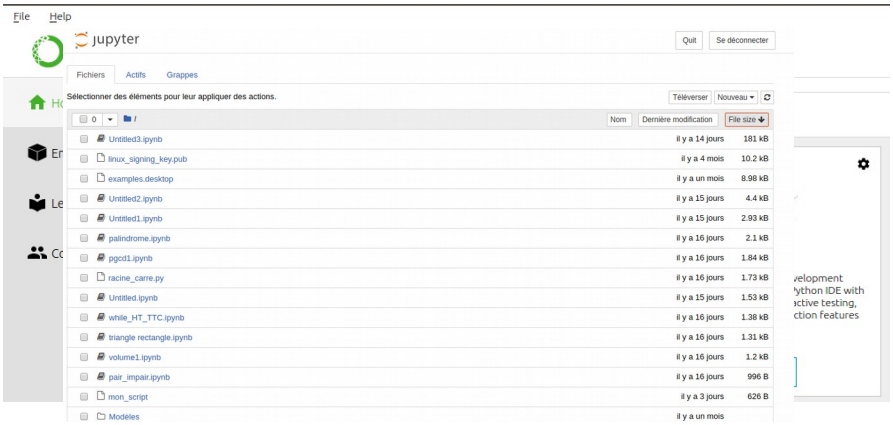
Installation d'Anaconda et Jupyter:

https://docs.anaconda.com/anaconda/install/linux/
https://www.youtube.com/watch?v=DY0DB_NwEu

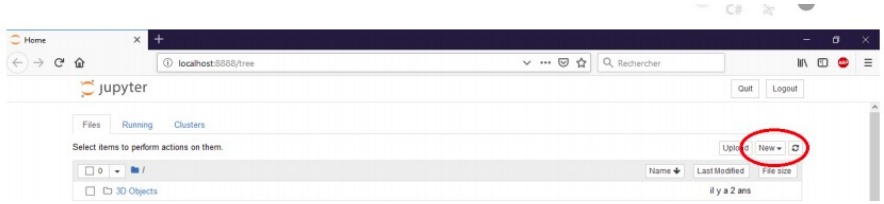
Ouvrir un terminal et taper « anconda-navigator » puis laisser le ouvert tant que l'on s'en servira.



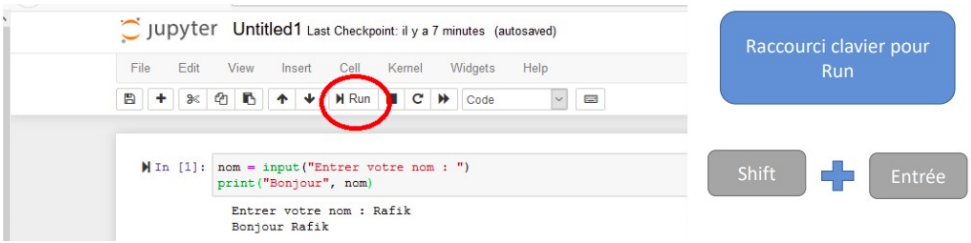
L’application web « ANACONDA NAVIGATOR » s’ouvrira enfin.
En lançant Jupyter, nous arrivons sur les fichiers créés sur le pc et donc nous accédons à tous les dossiers de l’utilisateur.



Pour créer un nouveau fichier, il faut cliquer sur « New ».



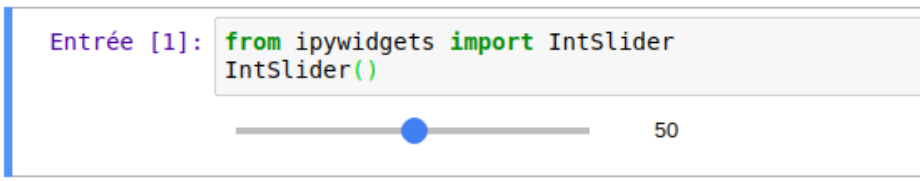
Création d’une variable « nom » que l’on lie avec input de façon à ce que l’utilisateur puisse répondre à la demande et écrire la réponse.
« print » sert à afficher un texte auquel on aura lier une variable, ici « nom ».
Une fois le script terminé, il suffit de l’exécuter en faisant « Run ».



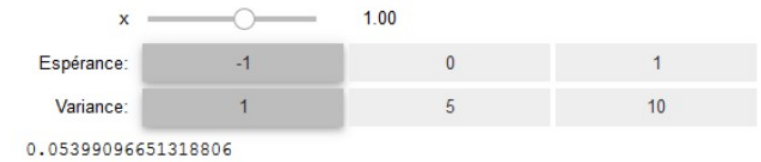

```
Entrée [5]: nom=input("Veuillez inscrire votre nom, svp  :")
            print("Bonjour",nom)

            Veuillez inscrire votre nom, svp  :Amaria
            Bonjour  Amaria
```

Générer un curseur

Ipywidgets *désigne un ensemble de composants graphiques pour le langage Python (*slider, combo box, *boutons, etc.) destinés à rendre les notebooks plus interactifs. En gros, il s’agit d’une architecture permettant de lier un objet Python (le* widget
), tournant dans le noyau, à sa représentation JavaScript/HTML/CSS tournant dans le navigateur.



```
from ipywidgets import ToggleButtons, interact
import scipy.stats


res=interact(scipy.stats.norm.pdf, x=FloatSlider(min=-8, max=8, step=1, value=1),
             loc=ToggleButtons(description='Espérance:', options=[-1, 0, 1]),
             scale=ToggleButtons(description='Variance:', options=[1, 5, 10]));
```




```
0.05399096651318806
```

https://ipywidgets.readthedocs.io/en/stable/examples/Widget%20List.html
pour Togglebuttons

La fonction interact (ipywidgets.interact) crée automatiquement des contrôles d'interface utilisateur pour explorer le code et les données de manière interactive.

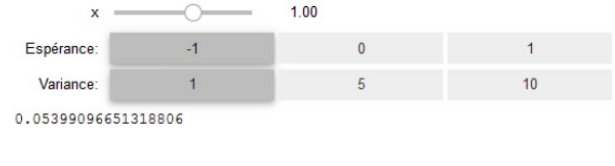
```
Entrée [13]: from ipywidgets import interact

             def my_function(x):
                 return x

             # create a slider
             interact(my_function, x=10)
```


```
10

Out[13]: <function __main__.my_function(x)>
```

Scipy.stats
Ce module contient un grand nombre de distributions de probabilités ainsi qu'une bibliothèque croissante de fonctions statistiques.

ICI :

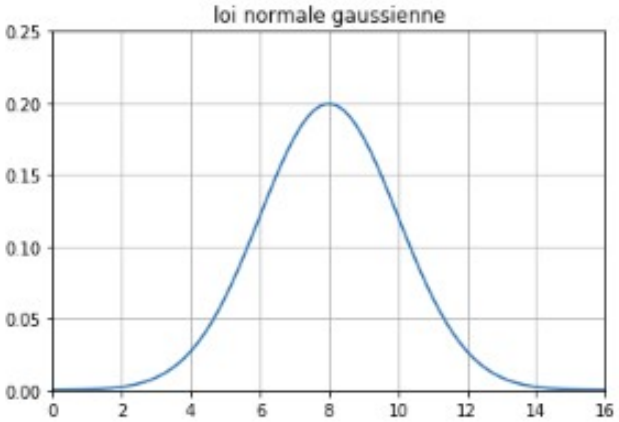
```
In [36]:
from ipywidgets import ToggleButtons, interact
import scipy.stats


res=interact(scipy.stats.norm.pdf, x=FloatSlider(min=-8, max=8, step=1, value=1),
             loc=ToggleButtons(description='Espérance:', options=[-1, 0, 1]),
             scale=ToggleButtons(description='Variance:', options=[1, 5, 10]));
```




```
0.05399096651318806
```

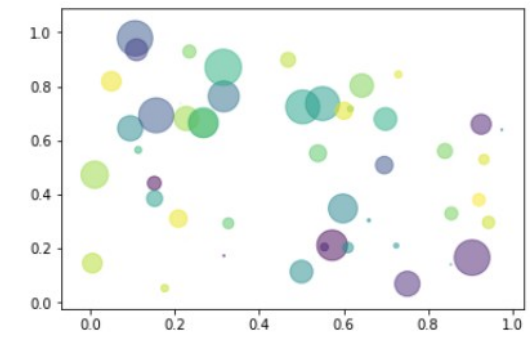
Tracer un graph

```
import matplotlib.pyplot as plt
# pour créer des graphes
import scipy.stats
#Module pour l'optimsation, l'algèbre linéaire, les statistiques, traitement d'image, ...
import numpy as np
# Module pour la manipulation de matrices,
#tableaux multidimensionnels et fonctions mathématiques opérant sur ces tableaux.

x = np.linspace(0,16,100)
# tableau avec 100 chiffres allant de 0 à 16

plt.plot(x,scipy.stats.norm.pdf(x,8,2))
# L'instruction plot() permet de tracer des courbes
#qui relient des points dont les abscisses et ordonnées
#sont fournies dans des tableaux.
#pdf -> Probability density function. -> densité (prend des réels en argument)

plt.grid()
#rajout d'une grille

plt.xlim(0, 16)
plt.ylim(0, 0.25)
plt.title('loi normale gaussienne')
plt.savefig('normale_distribution.png')
plt.show()
```



Un autre exemple de graph :

```
import numpy as np
import matplotlib.pyplot as plt

N=50
x=np.random.rand(N)
y=np.random.rand(N)
colors=np.random.rand(N)
area=np.pi * (15 * np.random.rand(N))**2

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



Créer un DataFrame :

```
Entrée [16]: import pandas as pd
             import numpy as np

             df =pd.DataFrame(np.random.randn(2,5))
             df.head()
```

Out[16]:

	0	1	2	3	4
0	0.380591	0.668969	1.355046	-1.537125	1.497141
1	-0.462271	0.118547	-1.067797	1.510320	0.326470

Les fonctions Magic :

2 types de fonctions Magic :

- Les fonctions Magic, préfixées par un seul symbole %, agissent uniquement sur la ligne sur la quel le se trouvent
- Les fonctions Magic, préfixées de 2 symboles %% agissent sur l'ensemble de la cellule.

Exemples :

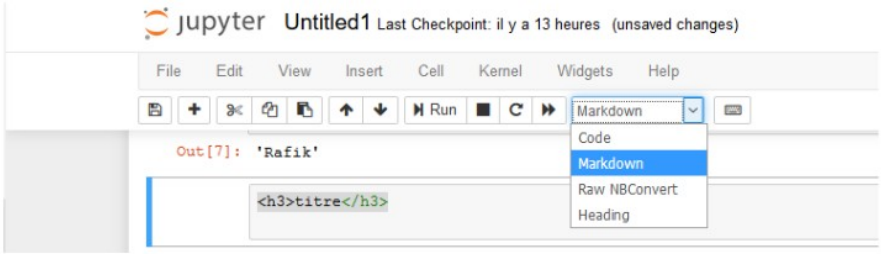
```
In [41]: %pwd
Out[41]: 'C:\\Users\\rafik'

In [42]: %timeit range(1000)
         233 ns ± 7.92 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

In [43]: %%timeit
         range(10)
         range(100)
         range(1000)
         587 ns ± 16.9 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

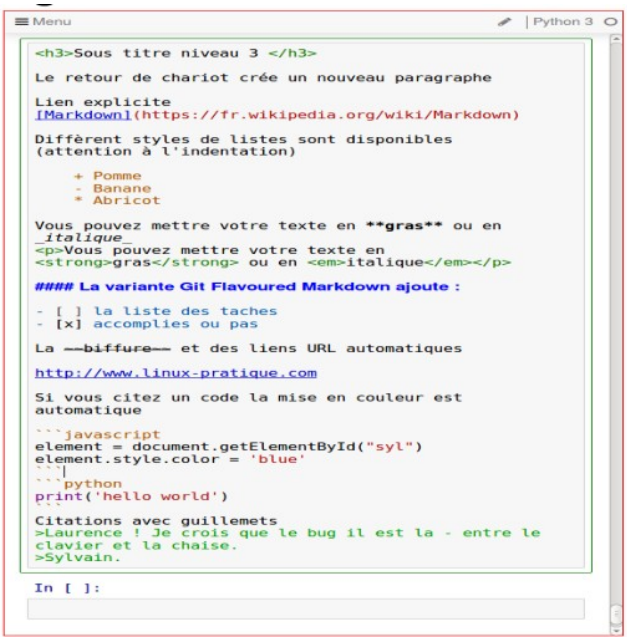
```
%lsmagic
```

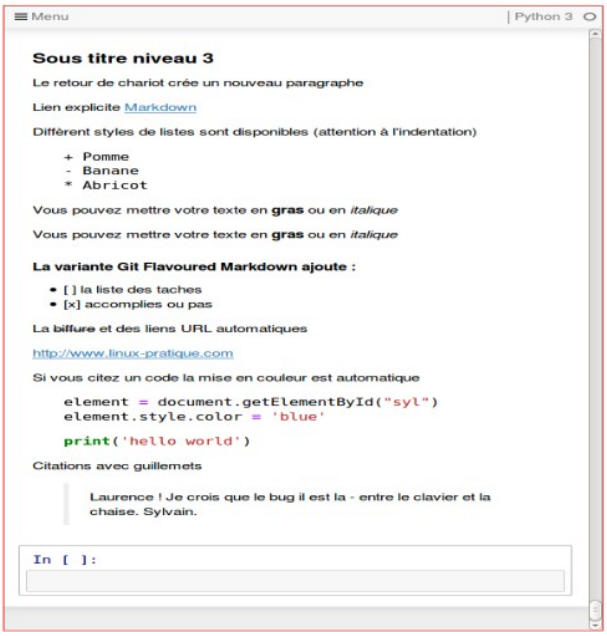
```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %bookmark  %cat  %cd  %clear  %colors  %conda
%config  %connect_info  %cp  %debug  %dhist  %dirs  %doctest_mode  %ed  %edit  %env  %gui  %hist  %history  %killb
gscripts  %ldir  %less  %lf  %lk  %ll  %load  %load_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %logstop
%ls  %lsmagic  %lx  %macro  %magic  %man  %matplotlib  %mkdir  %more  %mv  %notebook  %page  %pastebin  %pdb  %pde
f  %pdoc  %pfile  %pinfo  %pinfo2  %pip  %popd  %pprint  %precision  %prun  %psearch  %psource  %pushd  %pwd  %pyc
at  %pylab  %qtconsole  %quickref  %recall  %rehashx  %reload_ext  %rep  %rerun  %reset  %reset_selective  %rm  %r
mdir  %run  %save  %sc  %set_env  %store  %sx  %system  %tb  %time  %timeit  %unalias  %unload_ext  %who  %who_ls
%whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%javascript  %%js  %%latex  %%markdown  %%perl  %
%prun  %%pypy  %%python  %%python2  %%python3  %%ruby  %%script  %%sh  %%svg  %%sx  %%system  %%time  %%timeit  %%
writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

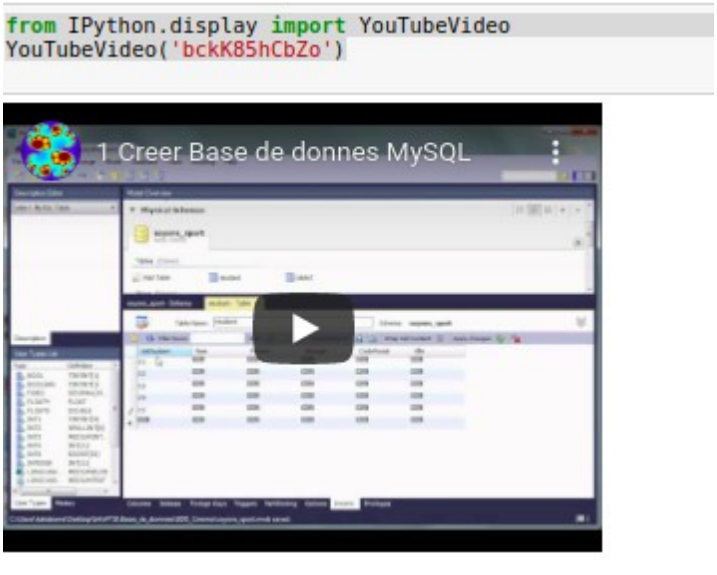
PROGRAMMATION ET REDACTION :

Possibilité de structurer le texte en utilisant des balises html/css, markdow ou LaTex.

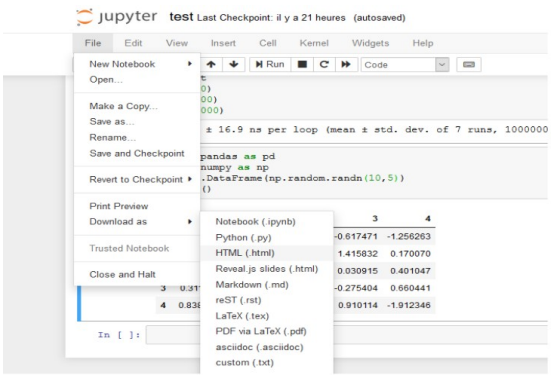








On peut exporter son document.



EN PRATIQUE ÇA MARCHE COMMENT ?

- Variable: un emplacement mémoire que l'on peut référencer par un nom. Sert à stocker une information.
- Autrement dit : une boite dans laquelle on peut mettre une valeur.
- Ensuite, on peut réutiliser cette valeur (sans la connaître) en écrivant le nom de la variable.

x = 3
nom = ‘Amaria'

Python distingue différentes sortes de variables suivant le type de valeur qu'elles mémorisent :

- un nombre entier
- une chaîne de caractères
- un nombre à virgule
- des valeurs complexes de type liste

ON PASSE À LA PRATIQUE

En utilisant la bibliothèque math :

cf : 2.2_PREMIER PAS EN PYTHON EXO

1. Trouver la solution de $3x^2-7x=23$

2. Il y a une fonction qui renvoie le plus grand diviseur commun à a et b.
Trouver la forme irréductible de la somme :

$$\frac{217}{440}+\frac{101}{256}+\frac{86}{71}$$

3 La saisie d'un rayon et d'une hauteur, calcule le volume d'un cône droit. (Utiliser l’instruction input pour saisir les valeurs et print pour afficher le resultat).
Le volume d'un cône de révolution est égal à un tiers de l’aire de sa base multipliée par la hauteur du cône h.
Si la base d'un cône est un disque de rayon R son aire est égale à : π × R2,
alors la formule du volume du cône est égal à : π × R2 × h ÷ 3

Les conditions :
if test : effectue les instructions qui suivent si le test est respecté.
On remarque que les instructions qu'on veut exécuter dans la structure conditionnelle ne sont pas alignées avec le if mais sont en retraits : on dit que ces instructions sont indentées.

- Structure conditionnelle sous Python

if test :
instructions1
else :
Instructions2
Cela effectue les instructions1 lorsque le test est vérifié, sinon effectue les instructions2.

- La ligne qui précède l'indentation se finit toujours par deux points. Appuyer sur la touche Entrée après avoir tapé « : » effectue automatiquement l'indentation.

Cela effectue les instructions1 indentées lorsque le test1 est vérifié, sinon effectue le test2 et, si celui-ci est vérifié, effectue les instructions2 indentées.

Remarques :
• On peut enchaîner autant de "elif" que nécessaire.
• On peut terminer une série de "elif" par un "else" afin d'être sûr de traiter tous les cas.

On passe à la pratique :

cf : 2.2_PREMIER PAS EN PYTHON EXERCICES EXO

1. Ecrire un programme qui à partir d'une saisie d'un nombre vous dit s'il est paire ou impaire.

```
print("Savoir si un nom est pair ou impair")
nb=input("Veuillez écrire un nombre, svp : ")
nb = int(nb)

r=nb%2

if (r == 0):
    print("Le nombre ", nb, "est pair.")
else :
    print("Le nombre ", nb, "est impair.")
```

```
Savoir si un nom est pair ou impair
Veuillez écrire un nombre, svp : 23
Le nombre  23 est impair.
```

ETANT DONNÉ LES LONGUEURS DES COTÉ D'UN TRIANGLE (HYPOTÉNUSE, COTÉ ADJACENT) ÉCRIRE UN PROGRAMME QUI VÉRIFIE SI UN TRIANGLE EST RECTANGLE.

Théorème de Pythagore :
On nomme a, b et c les longueurs des trois côtés d'un triangle.
Les triangles pour lesquels on a la relation $a^2 + b^2 = c^2$ sont tous les triangles rectangles dont l'hypoténuse est le côté de longueur c, et seulement eux.

```
print("Vérification si un triangle est rectangle ")
print("La vérification se fait par : c2=a2+b2.")

a = int(input("Veuillez entrer un chiffre pour a, svp : "))
b = int(input("Veuillez entrer un chiffre pour b, svp : "))
c = int(input("Veuillez entrer un chiffre pour c, svp : "))

calc1 = c*c
calc2 = (a*a +b*b)
print("c2 est égal à : ", calc1)
print("a2+b2 est égal à : ", calc2)

if (calc1==calc2):
    print("Le triangle est rectangle.")
else :
    print("Le triangle n'est pas rectangle.")
```

```
Vérification si un triangle est rectangle
La vérification se fait par : c2=a2+b2.
Veuillez entrer un chiffre pour a, svp : 3
Veuillez entrer un chiffre pour b, svp : 5
Veuillez entrer un chiffre pour c, svp : 4
c2 est égal à :  16
a2+b2 est égal à :  34
Le triangle n'est pas rectangle
```

En pratique ça marche comment ?

- programmation orientée objet : un style de programmation dans lequel les données et les opérations qui les manipulent sont organisées en classes et méthodes.
- Liste : Variable contenant une liste ordonnée d'éléments

  maListe = [4.7, 53, 2019, "formation", "Montpellier"]

- maListe.append(x) ajoute l'élément x à la fin de maListe
- maListe[i] donne le ième élément de maListe
- maListe.insert(i, x) insère x à la position i dans maListe
- maListe.remove(x) supprime le premier élément égal à x s'il y est
- maListe.count(x) compte le nb d'occurrences de x dans maListe
- maListe.sort() tri maListe
- maListe.reverse() renverse l'ordre de maListe

Range()

L'instruction range(début,fin,pas) génère une liste d'entiers (les paramètres début et pas sont optionnels)

• Dans l'intervalle [0 ; fin[si un seul paramètre est renseigné :
L=range(5) va créer la liste [0, 1, 2, 3, 4] de 5 termes, le premier terme sera L[0]=0, le dernier L[4]=5.

• Dans l'intervalle [début ; fin[si 2 paramètres sont renseignés L=range(1,5) va créer la liste [1, 2, 3, 4], le premier terme sera L[0]=1 et le dernier L[3]=5.

• Dans l'intervalle [début ; fin[mais de pas en pas, si les 3 paramètres
sont renseignés.

La boucle for :

```
for i in range(4):
print("i a pour valeur", i)
```

Pour tout i dans [0, 1, 2, 3]
afficher « i a pour valeur » la valeur i

```
c = [41, 32, 56, 34, 7]
for i in range(len(c)):
print("i vaut", i, "et c[i] vaut", c[i])
```

Pour tout i dans [0, 1, 2, 3, 5]
afficher « i vaut » la valeur i « et c[i]
vaut » la ième valeur de la liste

La boucle while

```
x = 1
while x < 10:
print("x a pour valeur", x)
x = x * 2
print("Fin")
```

Tant que x est inférieur à 10
afficher « x à pour valeur » la valeur x
x prend la valeur x^2
Afficher « Fin »

Pratique :

cf : 2.2_PREMIER PAS EN PYTHON EXO

CRÉER UN PROGRAMME QUI DONNE LE PRIX TTC APRÈS AVOIR SAISIE LE PRIX HT.
Ce programme doit se répéter pour pouvoir entrer plusieurs prix à la suite et ne s'arrêter que si l'utilisateur rentre 0.

ECRIRE UN PROGRAMME QUI VÉRIFIE SI LA LISTE [1, 2, 3, 4, 3, 2, 1] EST UN PALINDROME.