

Initiation à la programmation Python et à l'algorithmique

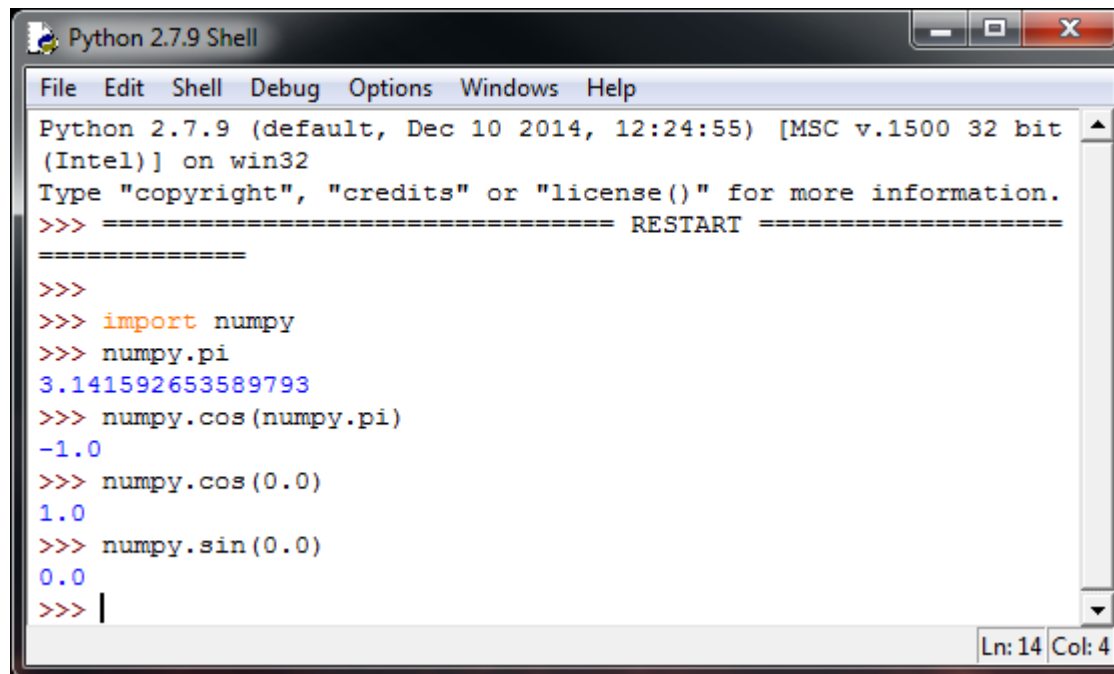


Python est un langage de programmation objet (un objet pouvant représenté un concept, ou une entité physique), multiplateforme.

La syntaxe permet une initiation aisée aux concepts de base de la programmation.

Premier contact : La console

La console est une fenêtre où l'on peut exécuter des commandes les unes après les autres. Les valeurs des variables vont être conservées jusqu'à la fermeture de celle-ci.



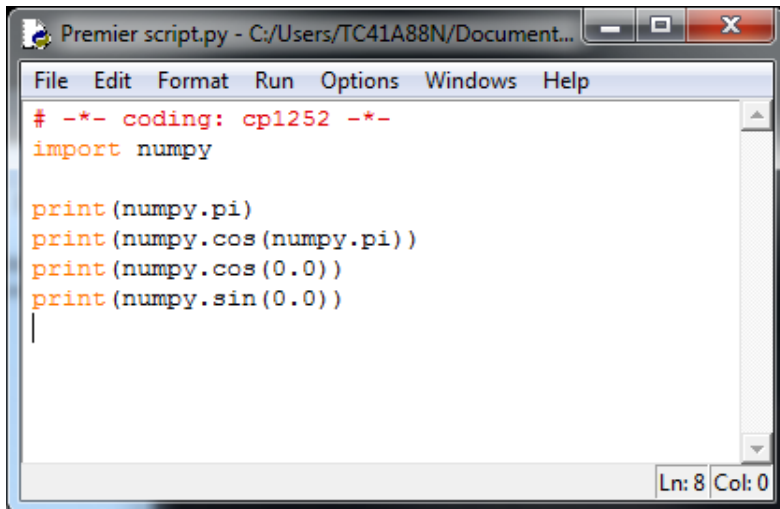
```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
>>> import numpy
>>> numpy.pi
3.141592653589793
>>> numpy.cos(numpy.pi)
-1.0
>>> numpy.cos(0.0)
1.0
>>> numpy.sin(0.0)
0.0
>>> |
```

Ln: 14 Col: 4

Premier contact : Le script

Un script est une zone de texte dans lequel on peut écrire les commandes que l'on veut que Python exécute. Ces commandes vont s'exécuter les unes après les autres du haut vers le bas.

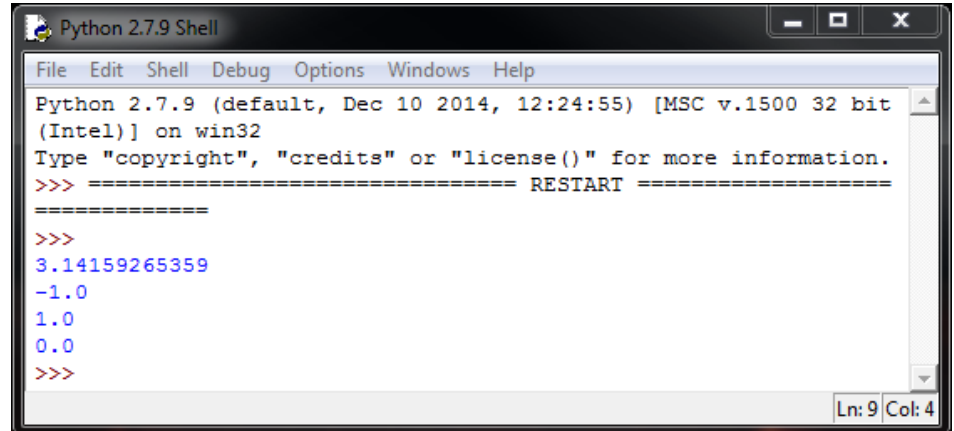
Lors de l'exécution, les informations vont apparaître dans la console.



```
File Edit Format Run Options Windows Help
# -*- coding: cp1252 -*-
import numpy

print(numpy.pi)
print(numpy.cos(numpy.pi))
print(numpy.cos(0.0))
print(numpy.sin(0.0))
|
```

Ln: 8 Col: 0

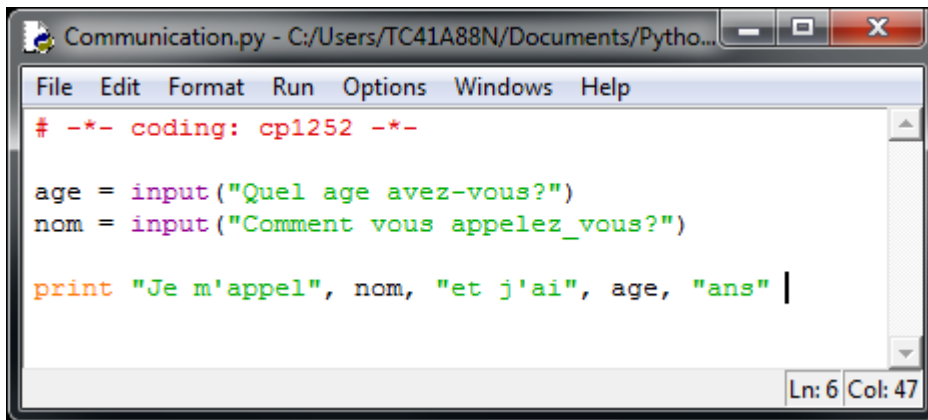


```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
3.14159265359
-1.0
1.0
0.0
>>>
```

Ln: 9 Col: 4

Communication Script/User

Pour communiquer avec votre programme, vous pouvez utiliser les commandes `input` et `print`.



```
File Edit Format Run Options Windows Help
# -*- coding: cp1252 -*-

age = input("Quel age avez-vous?")
nom = input("Comment vous appelez-vous?")

print "Je m'appel", nom, "et j'ai", age, "ans" |
```

Ln: 6 Col: 47

```
>>>
Quel age avez-vous?28
Comment vous appelez-vous?"Thomas"
Je m'appel Thomas et j'ai 28 ans
>>> |
```

***Attention :** Si vous voulez rentrer une chaîne de caractères, il ne faut pas oublier les guillemets (`'` ou `"`) !

Les variables

En programmation, les variables servent à manipuler les informations.

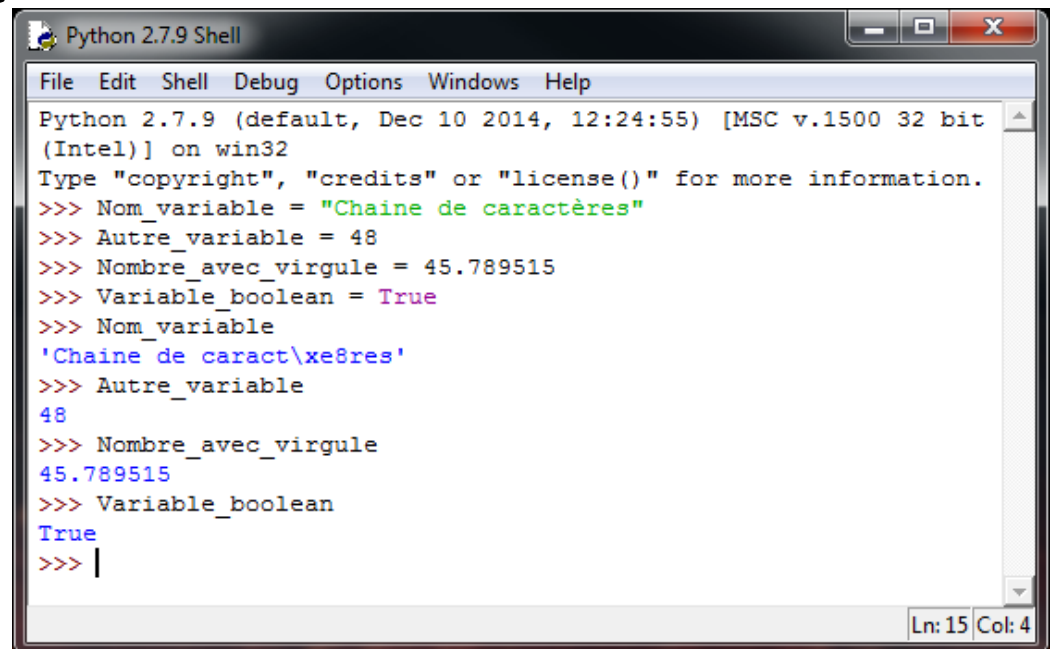
Plus le nom des variables est clair, plus vous vous y retrouverez dans votre code.



Les variables

Les variables sont des espaces de mémoire auquel ont donne un nom et une valeur. Il existe plusieurs type de variables:

- String = chaînes de caractères
- Integer = Nombres entiers
- Float = Nombre à virgule
- Boolean = True/False



```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Nom_variable = "Chaine de caractères"
>>> Autre_variable = 48
>>> Nombre_avec_virgule = 45.789515
>>> Variable_boolean = True
>>> Nom_variable
'Chaine de caract\xe8res'
>>> Autre_variable
48
>>> Nombre_avec_virgule
45.789515
>>> Variable_boolean
True
>>> |
```

Ln: 15 Col: 4

Les variables

Le nom d'une variable doit avoir les conditions suivants:

- Ne doit pas commencer par un chiffre
- Ne doit pas contenir d'espace
- Ne doit pas contenir de caractères spéciaux hormis l'underscore (« _ »)

Les variables

Le succès d'un script efficace vs déménagement:

DÉMÉNAGER



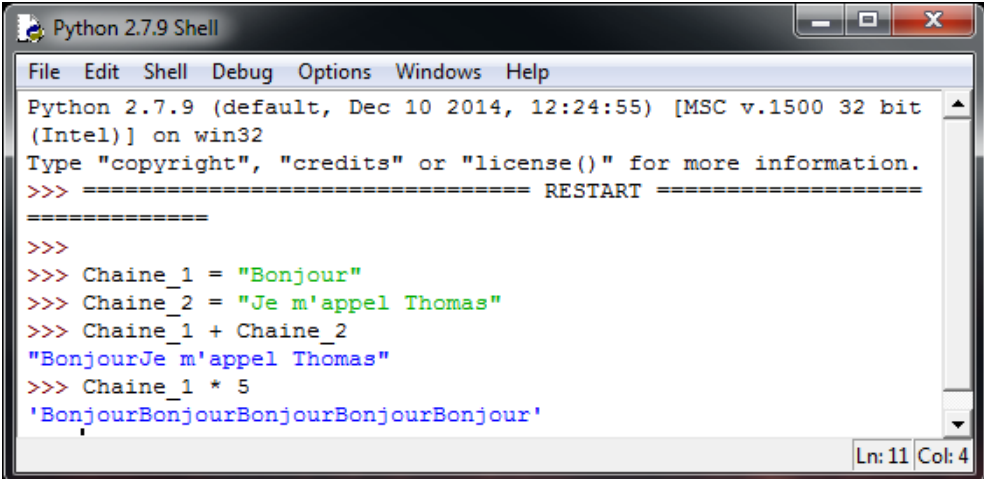
- Bien réfléchir aux variables que vous allez utiliser. Certaines variables peuvent ne pas être utiles pour votre projet.
- Si vous avez beaucoup de variables, pensez à aérer votre code pour le rendre plus lisible et n'hésitez pas à le commenter.
- N'hésitez pas à donner des noms représentant les données que vous mettrez dans vos variables pour éviter de vous poser la question.
- Pour des projets importants, n'hésitez pas à regrouper les variables en Classes pour plus de confort.

Les opérations : Les chaînes de caractères

Une chaîne de caractères (string) est une suite de caractères (alphabétiques, chiffres, spéciaux) considéré par l'ordinateur comme du texte.

Les opérations sur les chaînes de caractères sont les suivantes:

- L'addition (+) permet de concaténer deux string
- La multiplication (*) permet de multiplier un string par un nombre entier



```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
>>> Chaîne_1 = "Bonjour"
>>> Chaîne_2 = "Je m'appel Thomas"
>>> Chaîne_1 + Chaîne_2
'BonjourJe m'appel Thomas'
>>> Chaîne_1 * 5
'BonjourBonjourBonjourBonjourBonjour'
```



Les opérations : Les chaînes de caractères

Il est possible d'ajouter des variables dans une chaîne de caractères. Pour cela, il faut ajouter les balises suivantes:

- « %s » : Pour ajouter un string
- « %i » : Pour ajouter un nombre entier
- « %.5f » : Pour ajouter un nombre à virgule (ici, 5 correspond au nombre de décimales voulues)

```
import numpy

pi = numpy.pi
nom = "Pi"

sortie = "La valeur de %s est de %f mais on peut l'arrondir à %.2f."%(nom, pi, pi)
print(sortie)
```

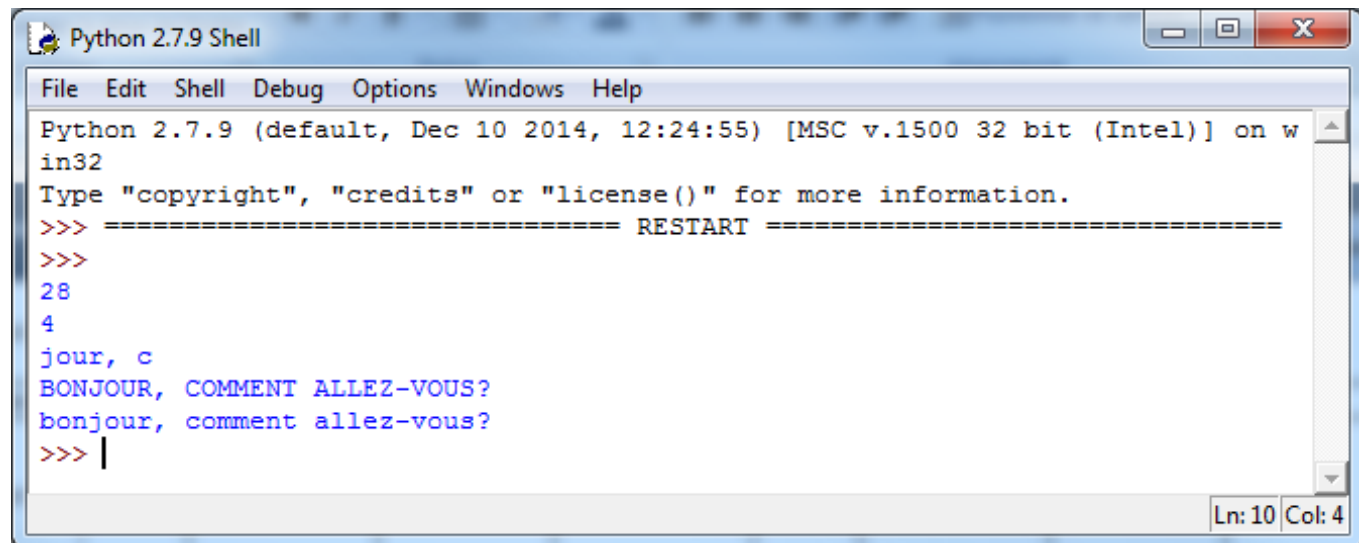
```
>>>
La valeur de Pi est de 3.141593 mais on peut l'arrondir à 3.14.
>>>
```

Les opérations : Les chaînes de caractères

Les commandes suivantes peuvent être utilisées:

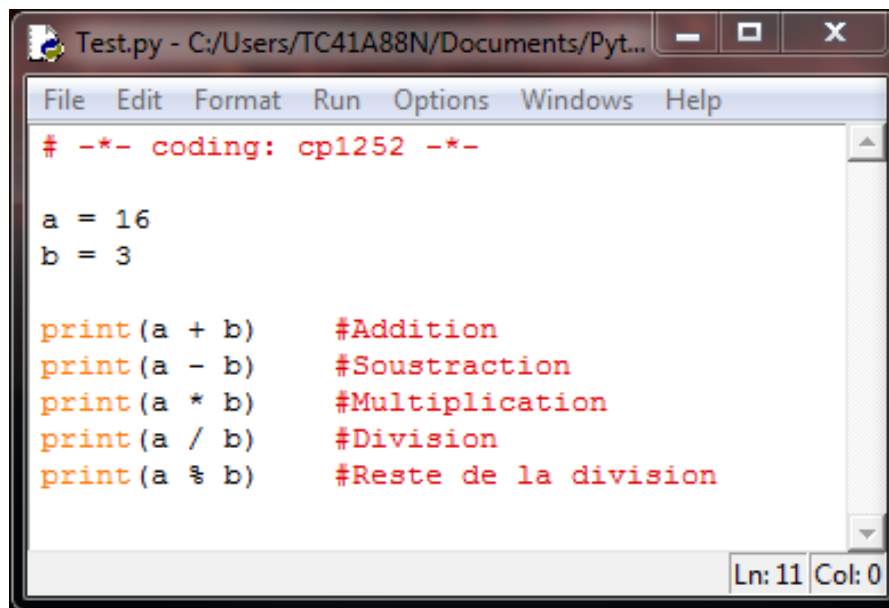
```
string1 = "Bonjour, comment allez-vous?"

print(len(string1))          #Renvoie le nombre de caracteres
print(string1.count("o"))    #Renvoie le nombre de "o" dans string1
print(string1[3:10])         #Renvoie les caracteres du 3eme au 9eme
print(string1.upper())       #Convertit en majuscules
print(string1.lower())       #Convertit en minuscules
```

A screenshot of a Python 2.7.9 Shell window. The window has a title bar with the text 'Python 2.7.9 Shell' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The main text area shows the following content: 'Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)] on win32', 'Type "copyright", "credits" or "license()" for more information.', a prompt '>>>>' followed by a line of dashes and the word 'RESTART', another prompt '>>>>', the output '28', the output '4', the output 'jour, c', the output 'BONJOUR, COMMENT ALLEZ-VOUS?', the output 'bonjour, comment allez-vous?', and a final prompt '>>>>' with a cursor. The status bar at the bottom right shows 'Ln: 10 Col: 4'.

Les opérations : Les nombres

Les opérations classiques sur les nombres sont les suivantes:

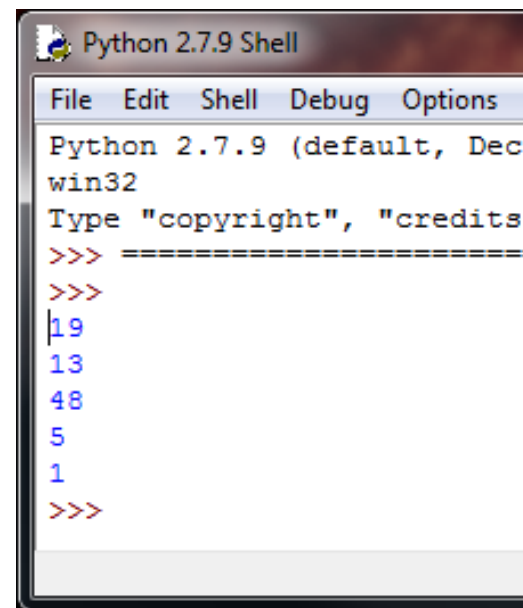


```
Test.py - C:/Users/TC41A88N/Documents/Pyt...
File Edit Format Run Options Windows Help
# -*- coding: cp1252 -*-

a = 16
b = 3

print(a + b)      #Addition
print(a - b)      #Soustraction
print(a * b)      #Multiplication
print(a / b)      #Division
print(a % b)      #Reste de la division

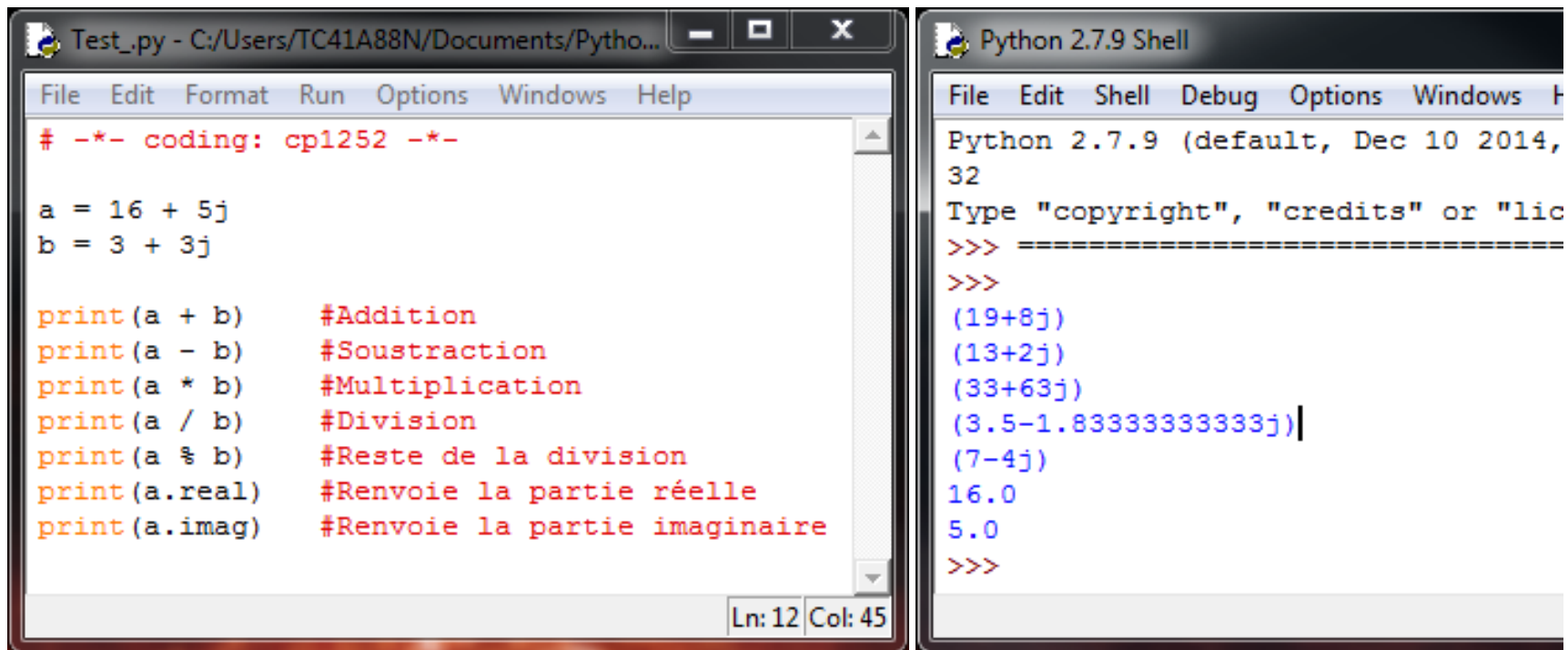
Ln: 11 Col: 0
```



```
Python 2.7.9 Shell
File Edit Shell Debug Options
Python 2.7.9 (default, Dec
win32
Type "copyright", "credits"
>>> =====
>>>
19
13
48
5
1
>>>
```

Les opérations : Les nombres complexes

Les opérations classiques sur les nombres complexes sont les suivantes:



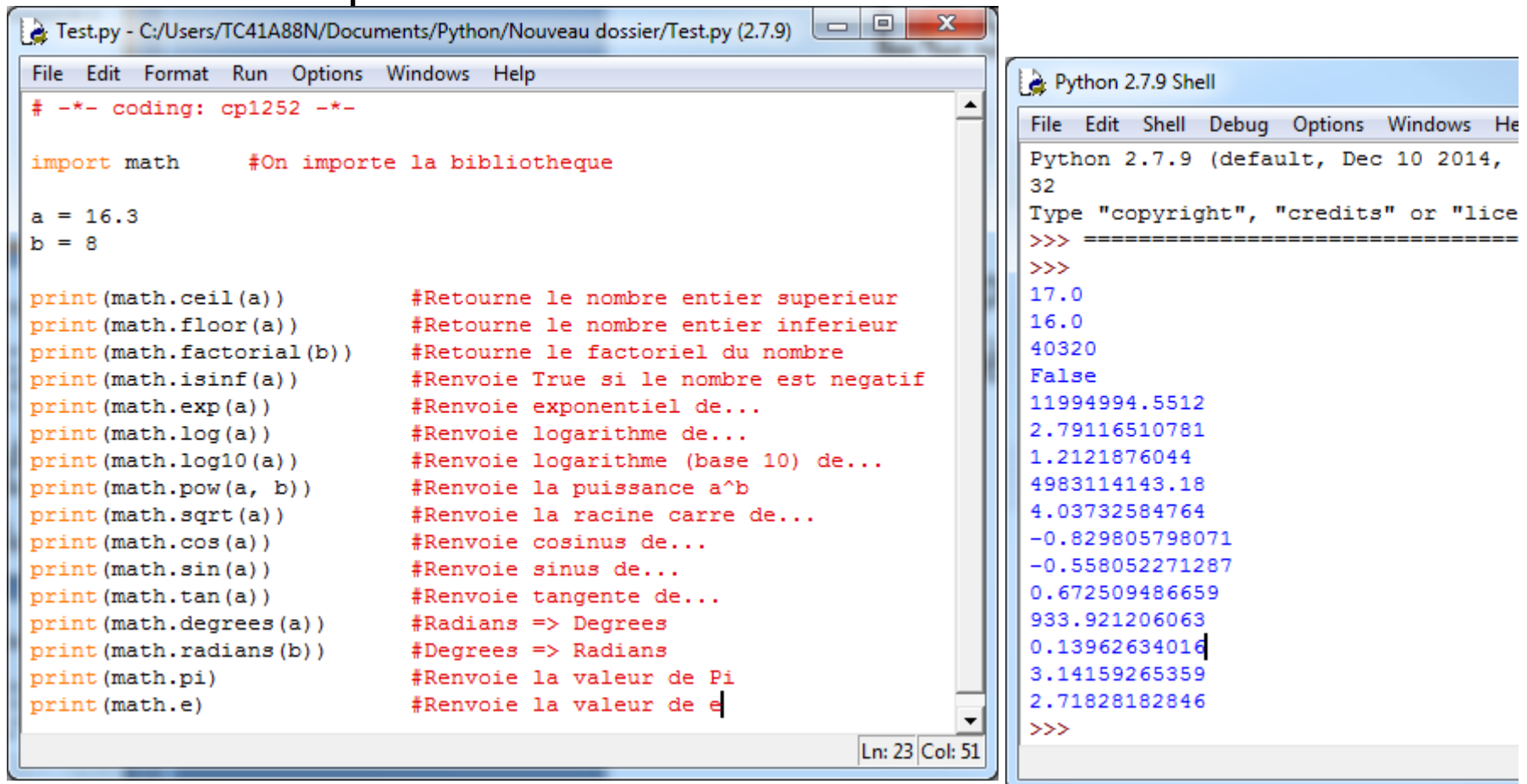
The image shows two side-by-side windows. The left window, titled 'Test_py - C:/Users/TC41A88N/Documents/Pytho...', contains a Python script. The script defines two complex numbers, $a = 16 + 5j$ and $b = 3 + 3j$, and then performs several operations on them, with comments in French. The right window, titled 'Python 2.7.9 Shell', shows the output of these operations.

```
# -*- coding: cp1252 -*-  
  
a = 16 + 5j  
b = 3 + 3j  
  
print(a + b)      #Addition  
print(a - b)      #Soustraction  
print(a * b)      #Multiplication  
print(a / b)      #Division  
print(a % b)      #Reste de la division  
print(a.real)     #Renvoie la partie réelle  
print(a.imag)     #Renvoie la partie imaginaire
```

```
Python 2.7.9 Shell  
File Edit Shell Debug Options Windows  
Python 2.7.9 (default, Dec 10 2014,  
32  
Type "copyright", "credits" or "lic  
>>> =====  
>>>  
(19+8j)  
(13+2j)  
(33+63j)  
(3.5-1.833333333333j)  
(7-4j)  
16.0  
5.0  
>>>
```


Les opérations : Bibliothèque « math »

Pour les opérations plus complexes, il peut être utile d'utiliser la bibliothèque « math ».



```
Test.py - C:/Users/TC41A88N/Documents/Python/Nouveau dossier/Test.py (2.7.9)
File Edit Format Run Options Windows Help
# -*- coding: cp1252 -*-

import math      #On importe la bibliotheque

a = 16.3
b = 8

print(math.ceil(a))      #Retourne le nombre entier superieur
print(math.floor(a))     #Retourne le nombre entier inferieur
print(math.factorial(b)) #Retourne le factoriel du nombre
print(math.isinf(a))     #Renvoie True si le nombre est negatif
print(math.exp(a))       #Renvoie exponentiel de...
print(math.log(a))       #Renvoie logarithme de...
print(math.log10(a))     #Renvoie logarithme (base 10) de...
print(math.pow(a, b))    #Renvoie la puissance a^b
print(math.sqrt(a))      #Renvoie la racine carre de...
print(math.cos(a))       #Renvoie cosinus de...
print(math.sin(a))       #Renvoie sinus de...
print(math.tan(a))       #Renvoie tangente de...
print(math.degrees(a))   #Radians => Degrees
print(math.radians(b))   #Degrees => Radians
print(math.pi)           #Renvoie la valeur de Pi
print(math.e)            #Renvoie la valeur de e

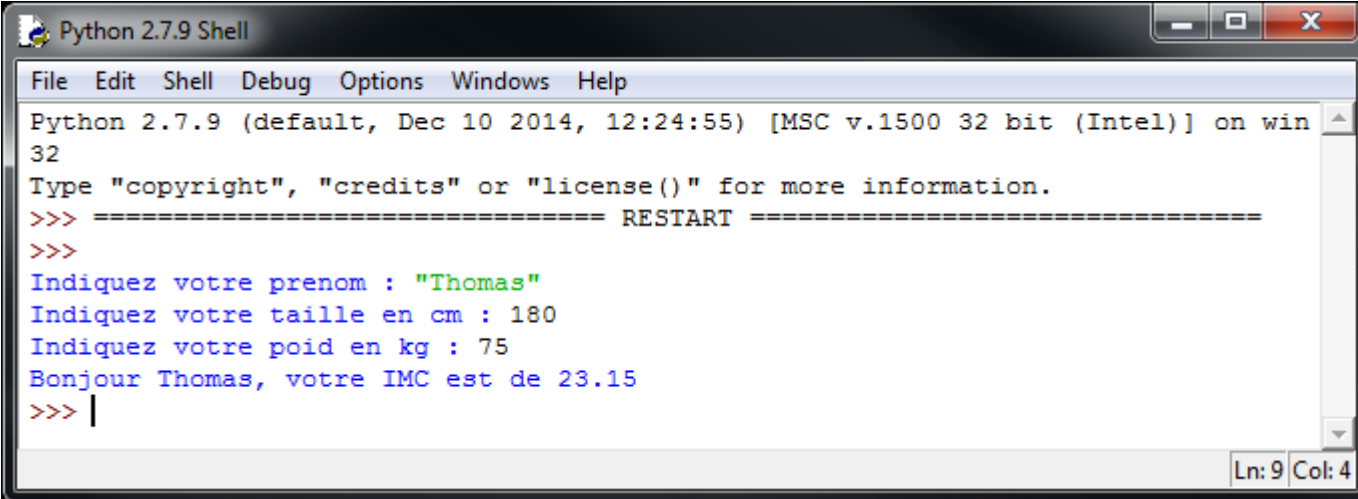
Ln: 23 Col: 51
```

```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows He
Python 2.7.9 (default, Dec 10 2014,
32
Type "copyright", "credits" or "lice
>>> =====
>>>
17.0
16.0
40320
False
11994994.5512
2.79116510781
1.2121876044
4983114143.18
4.03732584764
-0.829805798071
-0.558052271287
0.672509486659
933.921206063
0.13962634016
3.14159265359
2.71828182846
>>>
```

Les opérations : Exercices

Créer un programme qui vous demande votre nom, taille et poids pour vous donner votre IMC

$$IMC = \frac{Masse(kg)}{(Taille(m))^2}$$



```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Indiquez votre prenom : "Thomas"
Indiquez votre taille en cm : 180
Indiquez votre poid en kg : 75
Bonjour Thomas, votre IMC est de 23.15
>>> |
```

Les opérations : Exercices

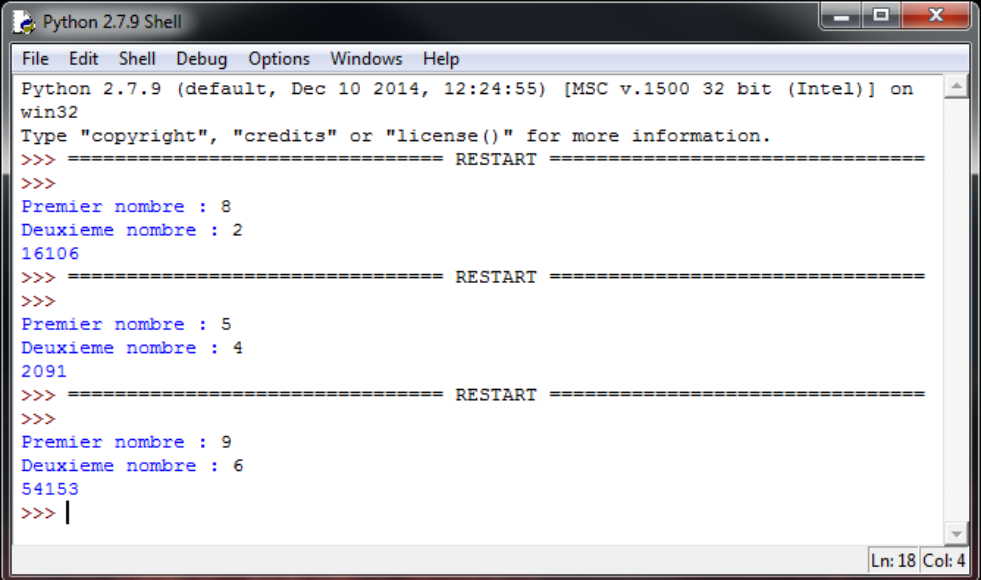
Créer un programme qui résolve ce casse tête

$b = \text{str}(a)$: converti le nombre a en string

8+2=16106

5+4=2091

9+6=54153

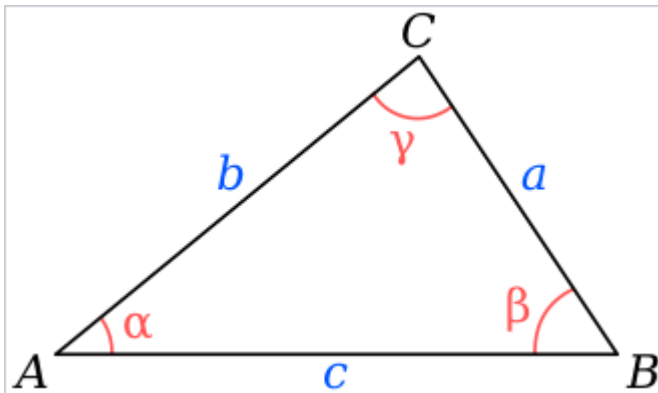


```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Premier nombre : 8
Deuxieme nombre : 2
16106
>>> ===== RESTART =====
>>>
Premier nombre : 5
Deuxieme nombre : 4
2091
>>> ===== RESTART =====
>>>
Premier nombre : 9
Deuxieme nombre : 6
54153
>>> |
```

Les opérations : Exercices

Le théorème d'Al-Kashi permet de déduire les longueurs et les angles d'un triangle quelconque.

$$c^2 = a^2 + b^2 - 2ab \cos \gamma.$$



Ecrire un programme qui, pour γ , b et a donné nous donne les valeurs de c , α et β .

```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows H
Python 2.7.9 (default, Dec 10 2014,
32
Type "copyright", "credits" or "lic
>>> =====
>>>
Donnez une valeur pour a : 10
Donnez une valeur pour b : 10
Donnez une valeur pour gamma : 90
c = 14.14214
alpha = 45.00000
beta = 45.00000
>>> =====
>>>
Donnez une valeur pour a : 25
Donnez une valeur pour b : 10
Donnez une valeur pour gamma : 45
c = 19.27295
alpha = 113.47601
beta = 21.52399
>>> |
```

Opérateurs de comparaison

Opérateur	Signification
==	Est égal à
!=	Est différent de
>=	Est plus grand ou égal que
<=	Est plus petit ou égal que
>	Est plus grand que
<	Est plus petit que

Opérateurs	
AND	[Condition 1] and [Condition 2] Si les deux conditions sont vraies => True
OR	[Condition 1] and [Condition 2] Si une des conditions est vraie => True
NOT	Not [Condition] Si la condition est fausse => True

Opérateurs de comparaison

Les opérateurs de comparaison renvoient « True » ou « False ».

```
a = 5
b = 6

print (a == b)    #Est ce que a est égal à b
print (a != b)    #Est ce que a est différent de b
print (a >= b)    #Est ce que a est supérieur ou égal à b
print (a <= b)    #Est ce que a est inférieur ou égal à b
print (a > b)     #Est ce que a est supérieur à b
print (a < b)     #Est ce que a est inférieur à b
print ("")
print (True and True)    #Opérateur "and"
print (True and False)
print (False and False)
print ("")
print (True or True)     #Opérateur "or"
print (True or False)
print (False or False)
print ("")
print not True           #Opérateur "not"
print not False
```

```
Type "copyright", "credits" or '
>>> =====
>>>
False
True
False
True
False
True

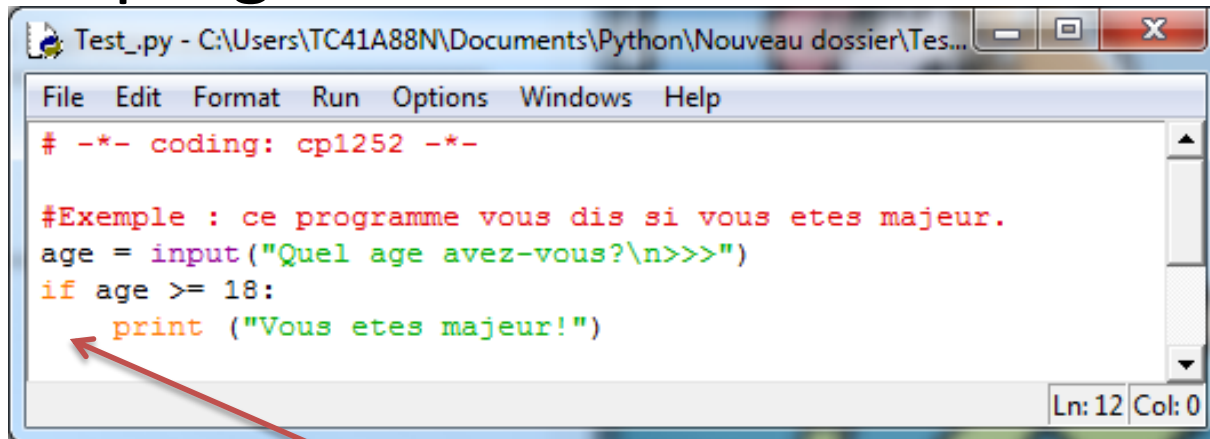
True
False
False

True
True
False

False
True
>>> |
```


Conditions

Dans un programme, les conditions permettent de faire des choix dépendants des différentes données que l'on peut retrouver dans notre programme.



```
Test_.py - C:\Users\TC41A88N\Documents\Python\Nouveau dossier\Tes...
File Edit Format Run Options Windows Help
# -*- coding: cp1252 -*-

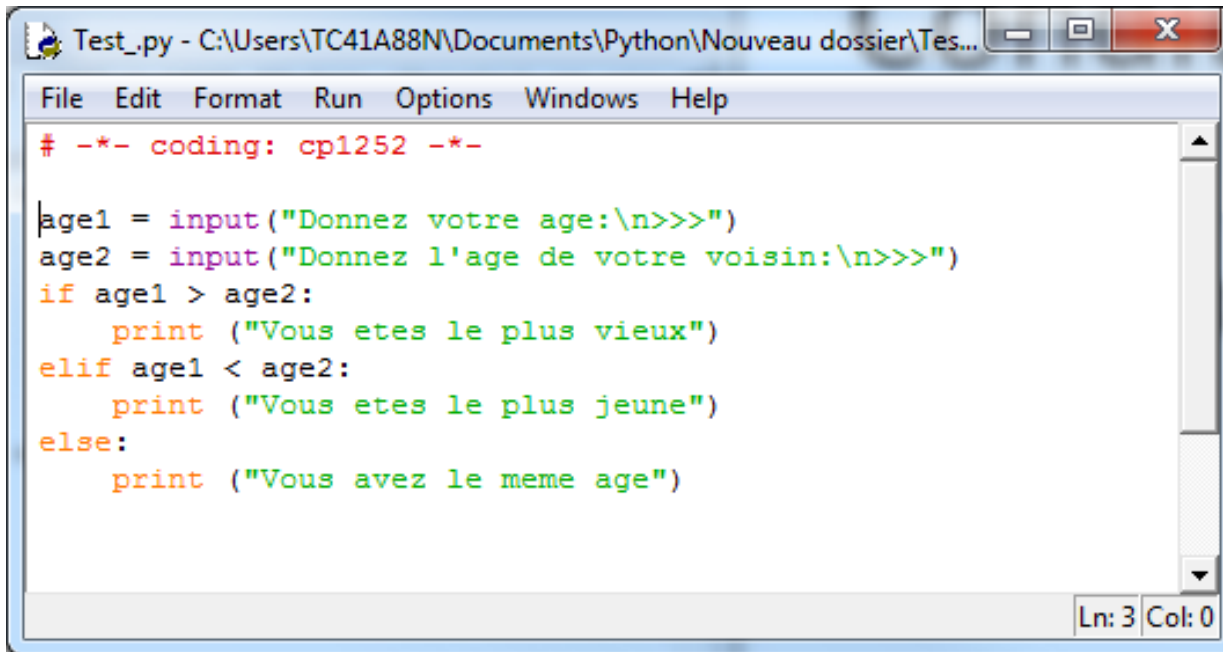
#Exemple : ce programme vous dis si vous etes majeur.
age = input("Quel age avez-vous?\n>>>")
if age >= 18:
    print ("Vous etes majeur!")
```

```
>>>
Quel age avez-vous?
>>>18
Vous etes majeur!
>>> =====
>>>
Quel age avez-vous?
>>>15
>>> |
```

*Attention, pour que les lignes soit considérées dans le « if », il est nécessaire de faire un alinéa!

Conditions

Il est également possible de gérer plusieurs conditions:



```
Test_.py - C:\Users\TC41A88N\Documents\Python\Nouveau dossier\Tes...
File Edit Format Run Options Windows Help
# -*- coding: cp1252 -*-

age1 = input("Donnez votre age:\n>>>")
age2 = input("Donnez l'age de votre voisin:\n>>>")
if age1 > age2:
    print ("Vous etes le plus vieux")
elif age1 < age2:
    print ("Vous etes le plus jeune")
else:
    print ("Vous avez le meme age")

Ln: 3 Col: 0
```

```
>>>
Donnez votre age:
>>>18
Donnez l'age de votre voisin:
>>>20
Vous etes le plus jeune
>>> =====
>>>
Donnez votre age:
>>>32
Donnez l'age de votre voisin:
>>>15
Vous etes le plus vieux
>>> =====
>>>
Donnez votre age:
>>>23
Donnez l'age de votre voisin:
>>>23
Vous avez le meme age
>>> |
```

Conditions

if : de l'anglais « si »

elif : de l'anglais (else if) « d'autre si »

else : de l'anglais « d'autre »

Le « if » doit toujours être placé en premier et le « else » (optionnel) à la fin!

Les alinéas sont **nécessaires** pour que l'ordinateur comprenne les ordres.

Conditions : Exercices

Les prix de l'électricité:

Puissance souscrite (kVA)	Abonnement (€TTC/mois)	Prix du kWh (€TTC/mois)
3	5,74	15,55
6	8,92	14,67
9	10,42	14,83
12	11,96	14,83
15	13,50	14,83

Le but de l'exercice est de créer un programme où vous devez renseigner la puissance souscrite. Le programme vous donnera le prix de l'abonnement et du kWh.

```
>>>  
Combien consommez-vous?:  
>>>5  
Le prix de l'abonnement sera de 8,92€/mois + 14,67€/kWh  
>>> |
```

Conditions : Exercices

Prix des places de ciné:

- | | |
|---------------------|--------|
| – Moins de 14 ans : | 5,00€ |
| – Moins de 18 ans : | 7,70€ |
| – Etudiant : | 8,70€ |
| – Adulte : | 12,10€ |
| | |
| • 3D | +2€ |
| • Lunettes | +1€ |

Faire un programme qui vous donne le prix de votre place de ciné.

```
>>>
Quel age avez-vous? >>> 28
Etes-vous etudiant? (Y/N) >>> "N"
Le film est-il en 3D? (Y/N) >>> "Y"
Avez-vous des lunettes 3D? (Y/N) >>> "Y"
Vous devez payer 15,10€.
>>> |
```

Les listes

Une liste est une variable pouvant contenir plusieurs informations. Ces listes peuvent contenir plusieurs type d'information.

```
liste_1 = ["Chat", 5, 30]  
liste_2 = ["Lapin", 23, 10]
```

Les listes peuvent également être ajoutées dans d'autres listes:

```
liste_3 = [liste_1, liste_2]  
print liste_3
```

```
>>>  
[['Chat', 5, 30], ['Lapin', 23, 10]]  
>>> |
```


Les listes

Pour manipuler les données des listes, nous devons faire appel à leurs coordonnées:

```
#Element n°      0      1      2      3      4      5
liste_test = ["Chat", "Chien", "Poule", "Vache", "Mouton", "Cochon"]
#Element n°     -6     -5     -4     -3     -2     -1

print (liste_test[1])
print (liste_test[3])
print (liste_test[-4])
print (liste_test[-1])
```

```
>>>
Chien
Vache
Poule
Cochon
>>> |
```

Il est possible de modifier ces données de la même manière:

```
liste_test[-1] = "Cheval"
print (liste_test[-1])
```

```
>>>
Cheval
>>>
```

Les listes


Pour repérer des éléments présents dans une liste incluse dans une liste, il faut faire comme suit:

```
liste_1 = ["Chat", 5, 30]
liste_2 = ["Lapin", 23, 10]
liste_3 = [liste_1, liste_2]
```

```
print (liste_3[1][0])
print (liste_3[0][2])
print (liste_3[1][1])
```

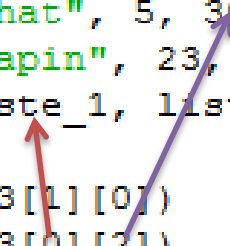
```
>>>
Lapin
30
23
>>> |
```

```
liste_1 = ["Chat", 5, 30]
liste_2 = ["Lapin", 23, 10]
liste_3 = [liste_1, liste_2]
```



```
print (liste_3[1][0])
```

```
liste_1 = ["Chat", 5, 30]
liste_2 = ["Lapin", 23, 10]
liste_3 = [liste_1, liste_2]
```



```
print (liste_3[1][0])
print (liste_3[0][2])
print (liste_3[1][1])
```

Les listes

Il est également possible d'utiliser des tranches qui nous permet de manipuler des bouts de listes:

```
liste_test = ["Chat", "Chien", "Poule", "Canard", "Cheval", "Dinde"]
```

```
#La sélection se fait à l'aide du modèle suivant:
```

```
#liste [i, j] où:
```

```
# i est inclu
```

```
# j est exclu
```

```
print (liste_test[2:-2])
```

```
print (liste_test[:1])
```

```
print (liste_test[-3:])
```

```
>>>
```

```
['Poule', 'Canard']
```

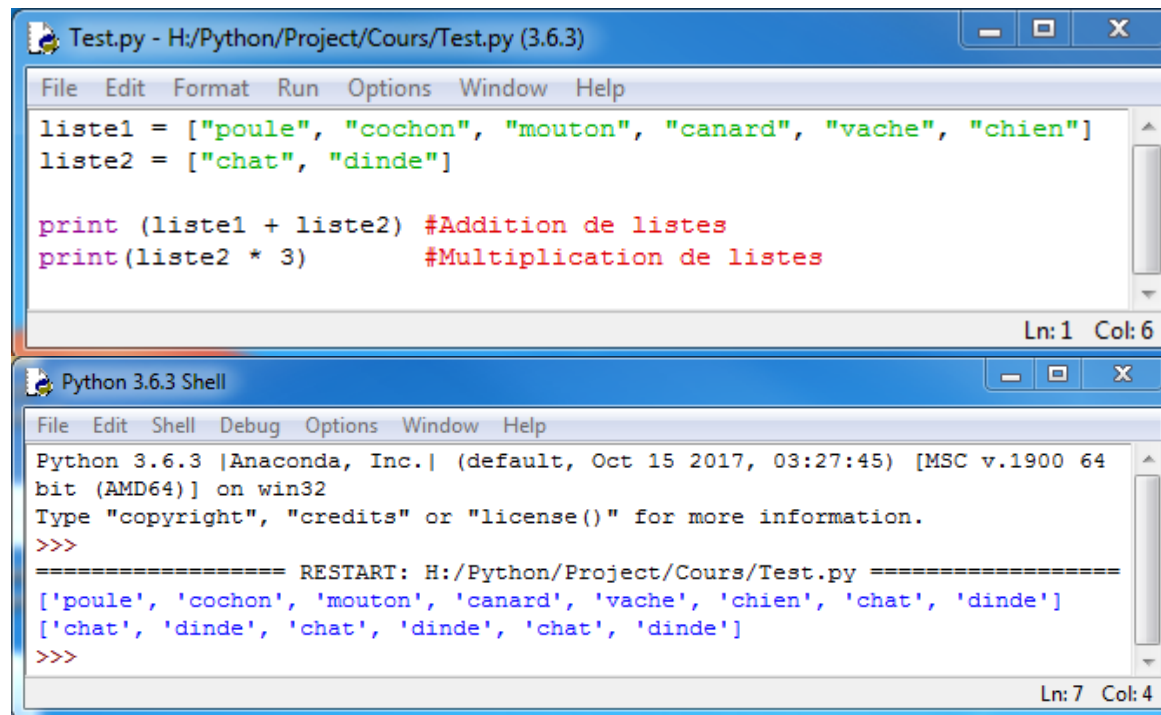
```
['Chat']
```

```
['Canard', 'Cheval', 'Dinde']
```

```
>>> |
```

Les listes

Les opérations possibles sur les listes sont les additions (+) et les multiplications (*) par des nombres entiers.



The screenshot displays two windows from a Python IDE. The top window, titled 'Test.py - H:/Python/Project/Cours/Test.py (3.6.3)', contains the following Python code:

```
liste1 = ["poule", "cochon", "mouton", "canard", "vache", "chien"]
liste2 = ["chat", "dinde"]

print (liste1 + liste2) #Addition de listes
print(liste2 * 3)       #Multiplication de listes
```

The bottom window, titled 'Python 3.6.3 Shell', shows the output of the code execution:

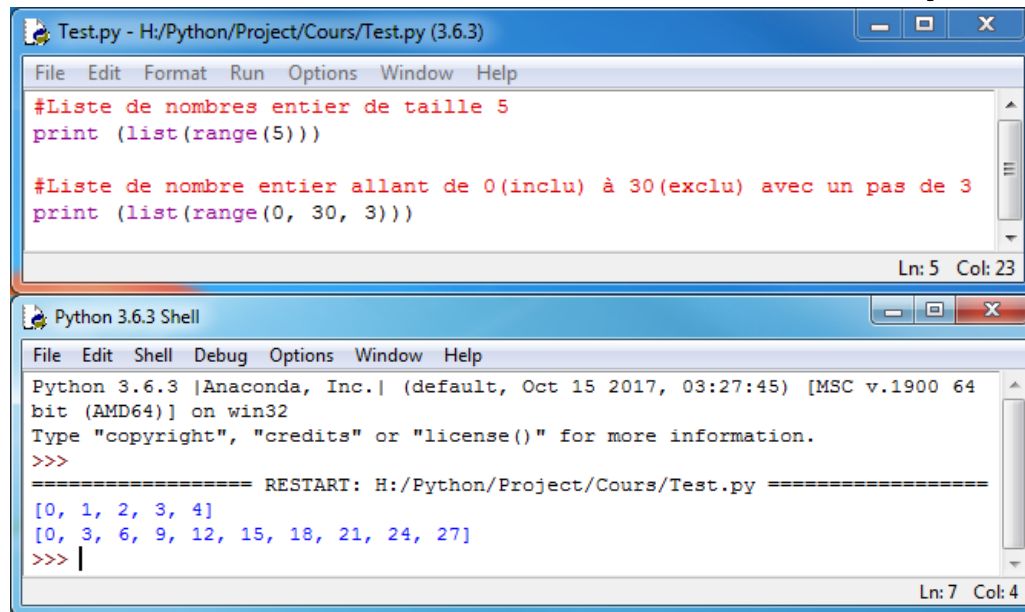
```
Python 3.6.3 [Anaconda, Inc.] (default, Oct 15 2017, 03:27:45) [MSC v.1900 64
bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: H:/Python/Project/Cours/Test.py =====
['poule', 'cochon', 'mouton', 'canard', 'vache', 'chien', 'chat', 'dinde']
['chat', 'dinde', 'chat', 'dinde', 'chat', 'dinde']
>>>
```

Les listes

L'instruction Range permet de créer des listes d'entiers. Elle fonctionne sur le modèle:

`range([début,] fin [, pas])`

Les arguments entre crochets sont optionnels.



The screenshot displays two windows from a Python IDE. The top window, titled 'Test.py - H:/Python/Project/Cours/Test.py (3.6.3)', contains the following Python code:

```
#Liste de nombres entier de taille 5
print (list(range(5)))

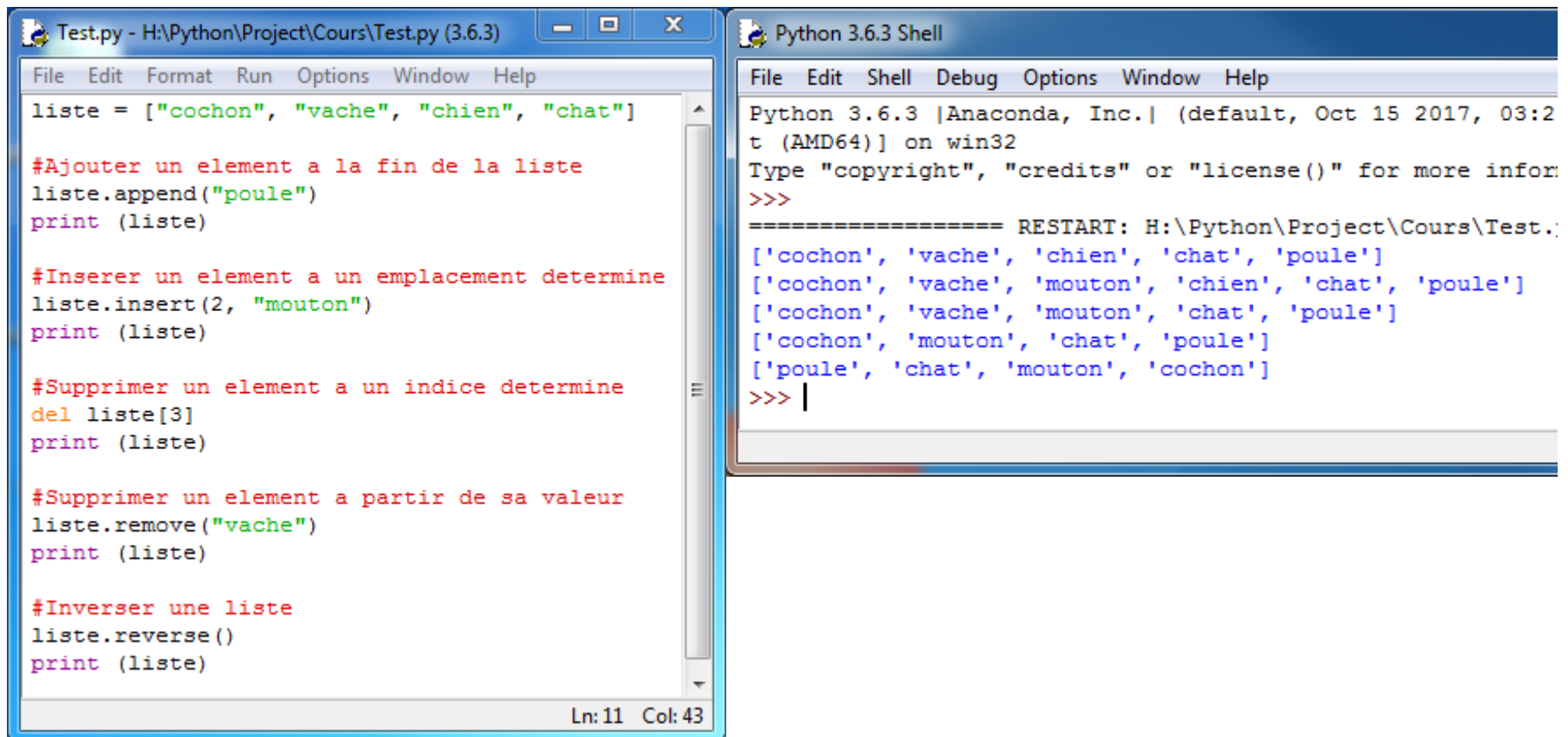
#Liste de nombre entier allant de 0(inclu) à 30(exclu) avec un pas de 3
print (list(range(0, 30, 3)))
```

The bottom window, titled 'Python 3.6.3 Shell', shows the execution output:

```
Python 3.6.3 |Anaconda, Inc.| (default, Oct 15 2017, 03:27:45) [MSC v.1900 64
bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: H:/Python/Project/Cours/Test.py =====
[0, 1, 2, 3, 4]
[0, 3, 6, 9, 12, 15, 18, 21, 24, 27]
>>> |
```

Les listes

Il y a plusieurs actions qui peuvent être utilisées sur les listes:



The image shows a Python IDE with two windows. The left window, titled 'Test.py - H:\Python\Project\Cours\Test.py (3.6.3)', contains a script with several list operations. The right window, titled 'Python 3.6.3 Shell', shows the output of these operations.

```
File Edit Format Run Options Window Help
liste = ["cochon", "vache", "chien", "chat"]

#Ajouter un element a la fin de la liste
liste.append("poule")
print (liste)

#Insérer un element a un emplacement determine
liste.insert(2, "mouton")
print (liste)

#Supprimer un element a un indice determine
del liste[3]
print (liste)

#Supprimer un element a partir de sa valeur
liste.remove("vache")
print (liste)

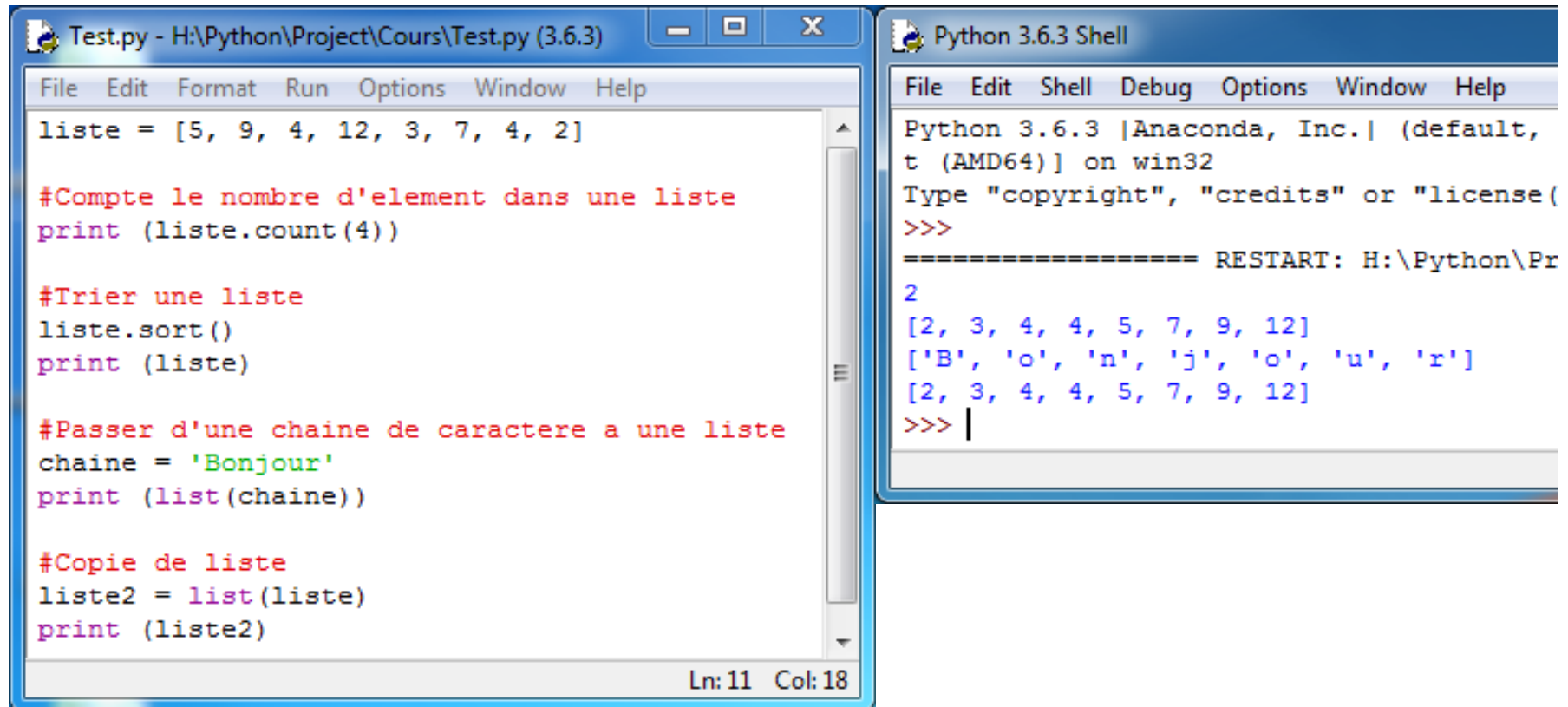
#Inverser une liste
liste.reverse()
print (liste)
```

Python 3.6.3 Shell

```
File Edit Shell Debug Options Window Help
Python 3.6.3 |Anaconda, Inc.| (default, Oct 15 2017, 03:2
t (AMD64)] on win32
Type "copyright", "credits" or "license()" for more infor
>>>
===== RESTART: H:\Python\Project\Cours\Test.:
['cochon', 'vache', 'chien', 'chat', 'poule']
['cochon', 'vache', 'mouton', 'chien', 'chat', 'poule']
['cochon', 'vache', 'mouton', 'chat', 'poule']
['cochon', 'mouton', 'chat', 'poule']
['poule', 'chat', 'mouton', 'cochon']
>>> |
```

Ln: 11 Col: 43

Les listes



The image shows a Python IDE with two windows. The left window, titled 'Test.py - H:\Python\Project\Cours\Test.py (3.6.3)', contains a Python script. The right window, titled 'Python 3.6.3 Shell', shows the output of the script's execution.

```
Test.py - H:\Python\Project\Cours\Test.py (3.6.3)
File Edit Format Run Options Window Help
liste = [5, 9, 4, 12, 3, 7, 4, 2]

#Compte le nombre d'element dans une liste
print (liste.count(4))

#Trier une liste
liste.sort()
print (liste)

#Passer d'une chaine de caractere a une liste
chaine = 'Bonjour'
print (list(chaine))

#Copie de liste
liste2 = list(liste)
print (liste2)
Ln: 11 Col: 18
```

```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 [Anaconda, Inc.] (default,
t (AMD64)] on win32
Type "copyright", "credits" or "license(
>>>
===== RESTART: H:\Python\Pr
2
[2, 3, 4, 4, 5, 7, 9, 12]
['B', 'o', 'n', 'j', 'o', 'u', 'r']
[2, 3, 4, 4, 5, 7, 9, 12]
>>> |
```

Listes : Exercices

La légende raconte qu'un jour, un roi découvrit un nouveau jeu: le jeu d'échec. Le concepteur du jeu voulu que le roi pose un grain de riz sur la première case du jeu et double le nombre de grains de riz à chaque case.

Créez une liste contenant autant d'éléments que de case dans le jeu d'échec. Chaque élément sera le nombre de grain de riz présent sur la case.

Le programme devra alors sommer le nombre de grains de riz de chaque élément et donner le nombre de tonnes de riz que le concepteur a gagné.

Un grain de riz $\approx 0.02g = 2.10^{-8}$ tonnes

Les boucles

Les boucles sont très importantes en programmation. Elles permettent de répéter des instructions un nombre de fois défini à l'avance.

```
i=1
while i<10000:#Tant que i est inférieur à 10 000
    #On ajoute 1 à i
    i=i+1
```

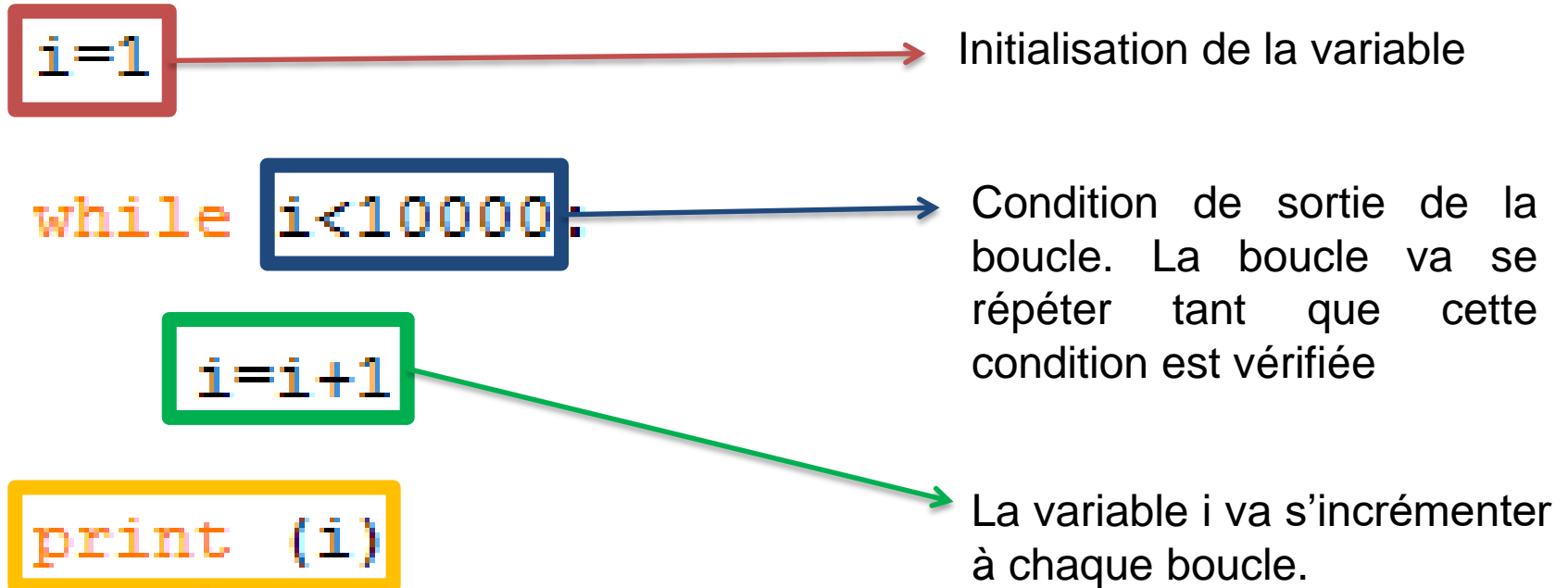
Les boucles : While

Une boucle **while** répète les instruction tant que la condition d'entrée est vérifiée (**True**).

Les instructions présentes dans la boucle doivent être décalées d'un alinéa de plus que l'initialisation de la boucle.

Attention, ce genre de boucle peut entrainer des boucles infinies!

Les boucles : While



Cette instruction n'est pas dans la boucle. Le résultat affiché dans la console sera la valeur de la variable i à la fin du programme (i = 10 000)

Les boucles : While

Exemple de boucle infinie:

```
i=2
```

```
while i>1:
```

```
    i=i+1
```

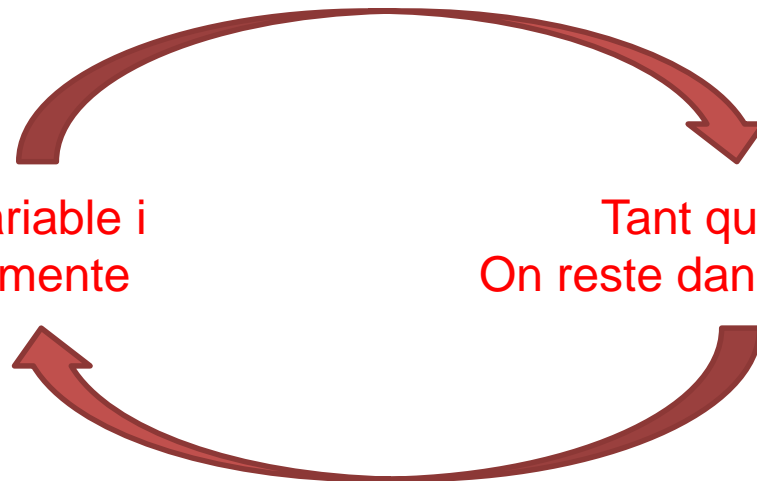
```
print (i)
```

→ Ici, les instructions se répéteront tant que i sera supérieur à 1

→ La variable i est initialisée à la valeur 2 et continue d'augmenter.

La variable i
augmente

Tant que i>1
On reste dans la boucle



Les boucle : For

La boucle **for** va parcourir une liste et renvoyer toutes les valeurs de la liste dans une variable définie dans la boucle.

```
liste = ['Chien', 'Chat', 'Poule', 'Vache', 'Poule', 'Lapin']
```

```
for i in liste:  
    print (i)
```

```
>>>
```

```
Chien
```

```
Chat
```

```
Poule
```

```
Vache
```

```
Poule
```

```
Lapin
```

```
>>> |
```



La variable *i* va prendre alternativement toutes les valeurs de la liste.

Les boucles : For

On peut utiliser l'outil `range` avec la boucle `for`. Cela permet de parcourir une liste d'entier définie

```
for i in range(10):  
    print (i)
```

```
>>>
```

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

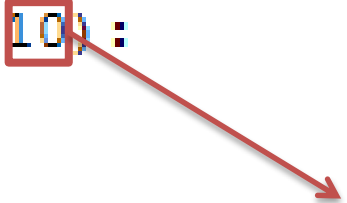
```
6
```

```
7
```

```
8
```

```
9
```

```
>>>
```



**La fonction `range` permet de déterminer à l'avance combien de fois on veut répéter les instructions (dans cet exemple, les instructions seront répétées 10 fois)*

Les boucles : Exercices

Héron (1er siècle après JC) a inventé une technique pour déterminer les racine carré des nombre supérieurs à 1.

Créer un programme qui détermine la racine d'un nombre avec cette méthode.

$$a_{n+1} = \frac{1}{2} \left(a_n + \frac{N}{a_n} \right)$$

N = Nombre initial

$a_0 = N$

Les boucles : Exercices

La conjecture de Syracuse est une suite d'entiers naturels définie de la manière suivante:

- On part d'un entier plus grand que zéro.
- Si l'entier est pair, on le divise par 2
- Si l'entier est impair, on le multiplie par 3 et on ajoute 1
- La suite devient infinie une fois qu'elle a atteint 1 (1,4,2,1,4,2,1...)

Créer un programme qui affiche les nombres dans la console.

Les boucles : Exercices

Le cycle de Syracuse est un cycle qui fini toujours sur les mêmes chiffres (1 ou 4).

- Si la suite tombe sur 4, elle va boucler à l'infinie. On dit que le chiffre est malheureux.
- Si la suite tombe sur 1, il n'y a pas de boucle donc le chiffre est heureux.

Pour déterminer le nombre suivant, il faut faire la somme des carrés des chiffres du nombre. Par exemple:

$$\underline{42} : \quad 4^2 + 2^2 \quad = 16 + 4 \quad = 20$$

$$\underline{127} : \quad 1^2 + 2^2 + 7^2 \quad = 1 + 4 + 49 \quad = 54$$

Créer un programme qui affiche les nombres dans la console.

Les fonctions

Une fonction est un morceau de code que l'on peut appeler à tout moment dans une autre partie de son code. Lorsque l'on a un nombre important de fonctions, il est possible de les regrouper dans des modules.

```
#Le but de cette fonction est d'écrire dans la console
def Hello_world():
    print ("Hello World!")
```

Les fonctions

Le but d'une fonction est de pouvoir être appelée à différents endroits sans pour autant devoir répéter les instructions:

```
#Le but de cette fonction est d'écrire dans la console
def no_conduite():
    print ("Vous ne pouvez pas conduire seul!")

#Le programme suivant va poser des questions et déterminer
#si la personne peut conduire
if input("Etes-vous majeur? (1/0)\n>>>") == 1:
    if input("Avez-vous le permis? (1/0)\n>>>") == 1:
        if input("Etes-vous sous forte medication? (1/0)\n>>>") == 1:
            no_conduite()
        else:
            print ("Vous pouvez conduire seul!")
    else:
        no_conduite()
else:
    no_conduite()
|
```

Les fonctions

Une fonction peut également utiliser des arguments (données d'entrée) et renvoyer des résultats:

```
#Trouve les coefficients directeurs d'une droite
def coef_droite(x1, x2, y1, y2):
    a = (y2 - y1) / (x2 - x1)
    b = y1 - a * x1
    return a, b

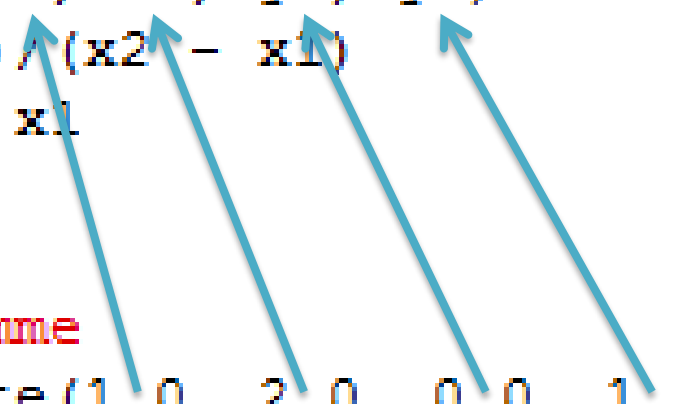
#Debut du programme
x, y = coef_droite(1.0, 2.0, 0.0, 1.0)
print (x, y)
```

Les fonctions

Une fonction peut également utiliser des arguments (données d'entrée) et renvoyer des résultats:

```
#Trouve les coefficients directeurs d'une droite
def coef_droite(x1, x2, y1, y2):
    a = (y2 - y1)/(x2 - x1)
    b = y1 - a * x1
    return a, b

#Debut du programme
x, y = coef_droite(1.0, 2.0, 0.0, 1.0)
print (x, y)
```


A diagram consisting of four blue arrows pointing from the arguments of a function call to the parameters of a function definition. The first arrow points from '1.0' in the function call to 'x1' in the function definition. The second arrow points from '2.0' to 'x2'. The third arrow points from '0.0' to 'y1'. The fourth arrow points from '1.0' to 'y2'.

Les fonctions

Une fonction peut également utiliser des arguments (données d'entrée) et renvoyer des résultats:

```
#Trouve les coefficients directeurs d'une droite
def coef_droite(x1, x2, y1, y2):
    a = (y2 - y1) / (x2 - x1)
    b = y1 - a * x1
    return a, b

#Debut du programme
x, y = coef_droite(1.0, 2.0, 0.0, 1.0)
print (x, y)
```



Les fonctions

Il est possible de donner des valeurs aux arguments, ceux-ci prendront alors la valeur indiquée si aucune valeur ne leurs sont appliqué.

```
#Trouve les coefficients directeurs d'une droite
def coef_droite(x1, y1, x2=0.0, y2=0.0):
    a = (y2 - y1) / (x2 - x1)
    b = y1 - a * x1
    return a, b

#Debut du programme
x, y = coef_droite(1.0, 2.0)
print (x, y)
```

Fonctions : Exercices

Créer une fonction qui prend un argument (un nombre) et qui renvoie:

- **True** si le nombre est premier
- **False** si le nombre n'est pas premier

Fonctions : Exercices

Créer une fonction qui prend deux arguments (deux nombres) et qui renvoie:

- **True** si les nombres sont premier entre eux
- **False** si les nombres ne sont pas premier entre eux

Fonctions : Exercices

Créer une fonction qui prend deux arguments (deux nombres) et qui renvoie:

- **True** si les nombres sont amicaux
- **False** si les nombres ne sont pas amicaux

Deux nombres sont dits amicaux si la somme des diviseurs propres de l'un (diviseurs autre que lui-même) égale l'autre.

Ex:

220 et 284

<u>Diviseurs de 220:</u>	1+2+4+5+10+11+20+22+44+55+110	=284
<u>Diviseurs de 284:</u>	1+2+4+71+142	=220

Exercice : Triangle de Sierpiński

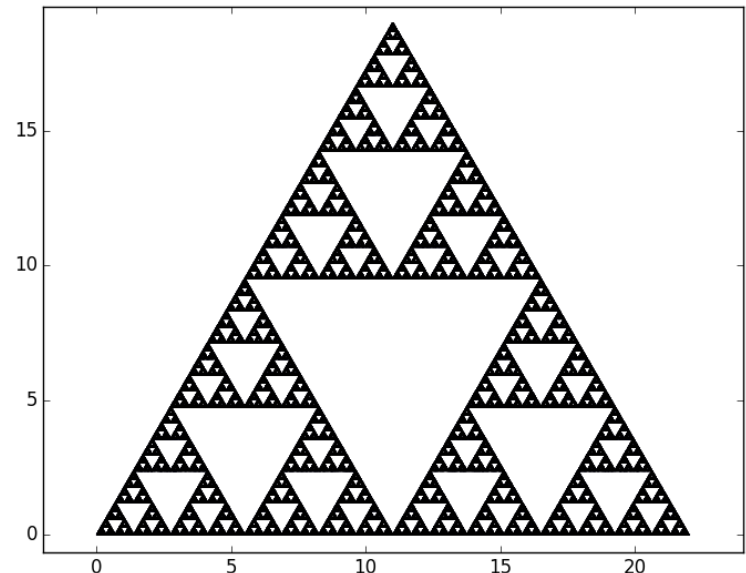
Les règles pour la création de ce triangle sont simples:

- On initialise la position d'un point au hasard (ou choisi)
- On crée 3 points (A, B, C) n'étant pas sur la même ligne.
- Pour trouver la position du point suivant, on prend un nombre aléatoire (entre 1 et 3 inclus):

➤ Si $n=1$, alors le nouveau point se trouve à mi-distance entre le point précédent et le point A

➤ Si $n=2$, alors le nouveau point se trouve à mi-distance entre le point précédent et le point B

➤ Si $n=3$, alors le nouveau point se trouve à mi-distance entre le point précédent et le point C



Exercice : La fougère de Barnsley

Les règles pour la création de cette forme sont simples:

- On initialise la position d'un point au hasard (ex: $P_i [0, 0]$)
- Pour trouver la position du point suivant, on prend un nombre aléatoire n entre 0 et 1 et on suit le système suivant:

$$\left\{ \begin{array}{lll} \forall n < 0.01 & x_i = 0 & y_i = 0.16 * y_{i-1} \\ \forall 0.01 \leq n < 0.86 & x_i = 0.85 * x_{i-1} + 0.04 y_{i-1} & y_i = 1.6 - 0.04 * x_{i-1} + 0.85 * y_{i-1} \\ \forall 0.86 \leq n < 0.93 & x_i = 0.2 * x_{i-1} - 0.26 * y_{i-1} & y_i = 1.6 + 0.23 * x_{i-1} + 0.22 * y_{i-1} \\ \forall n \geq 0.93 & x_i = -0.15 * x_{i-1} + 0.28 * y_{i-1} & y_i = 0.44 + 0.26 * x_{i-1} + 0.24 * y_{i-1} \end{array} \right.$$



Exercices : Déterminer Pi au hasard

Nous allons tirer au sort des coordonnées d'un point dans un carré de taille 1x1.

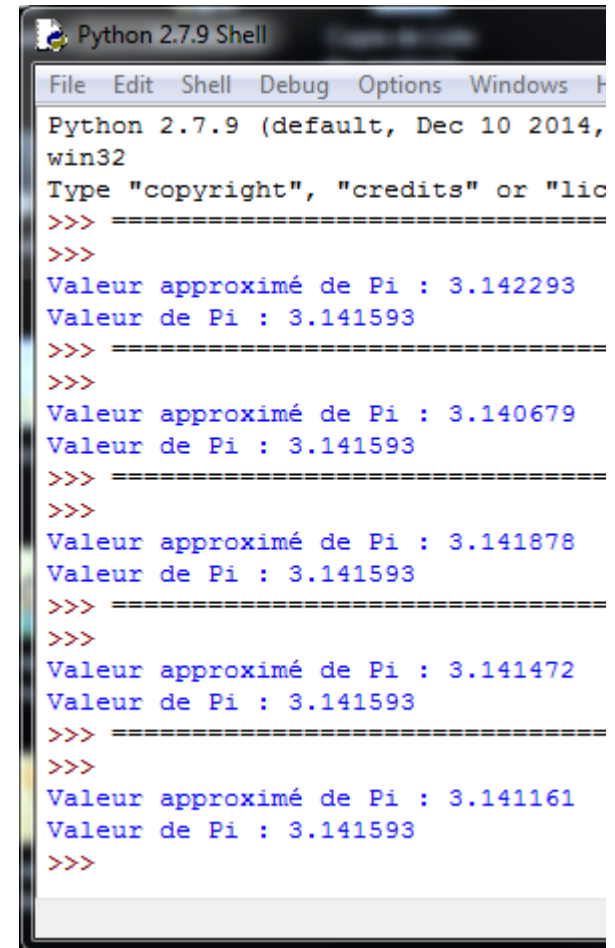
Soit le point $P_n [x, y]$ avec x et $y = \text{Random}$.

Si le point est compris dans le disque de centre O [0, 0] et de rayon 1, alors on incrémente la valeur correspondant au nombre de tirs réussis.

On incrémente si $x^2 + y^2 \leq 1$

On détermine la valeur de Pi avec ces deux valeurs.

$$Pi = 4 * Nbr_tirs_réussis / Nbr_tirs$$



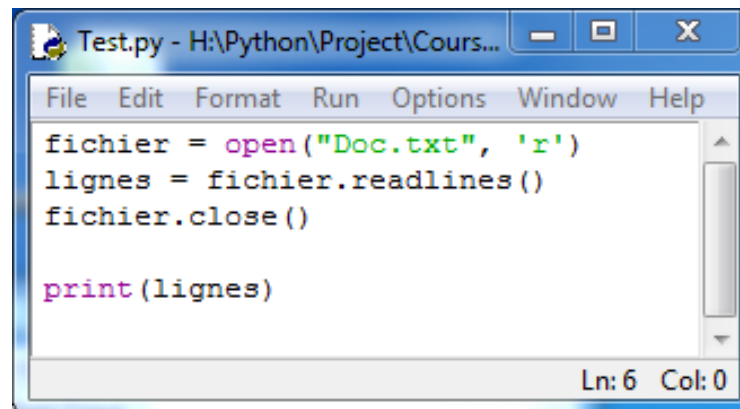
```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows
Python 2.7.9 (default, Dec 10 2014, win32)
Type "copyright", "credits" or "lic
>>> =====
>>>
Valeur approximé de Pi : 3.142293
Valeur de Pi : 3.141593
>>> =====
>>>
Valeur approximé de Pi : 3.140679
Valeur de Pi : 3.141593
>>> =====
>>>
Valeur approximé de Pi : 3.141878
Valeur de Pi : 3.141593
>>> =====
>>>
Valeur approximé de Pi : 3.141472
Valeur de Pi : 3.141593
>>> =====
>>>
Valeur approximé de Pi : 3.141161
Valeur de Pi : 3.141593
>>>
```

Ecriture/Lecture fichier .txt

La prise en compte ou l'export d'un nombre important de données nécessite l'utilisation de fichier pour stocker les informations. Elles pourront alors être utilisées dans un autre logiciel.

Pour cela, nous utilisons la commande `open` pour ouvrir un document. La lecture se fait en utilisant l'argument `'r'` (read).

L'instruction `.readlines()` retourne une liste des différentes lignes du fichier.

A screenshot of a Python IDE window titled 'Test.py - H:\Python\Project\Cours...'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The main text area contains the following Python code:

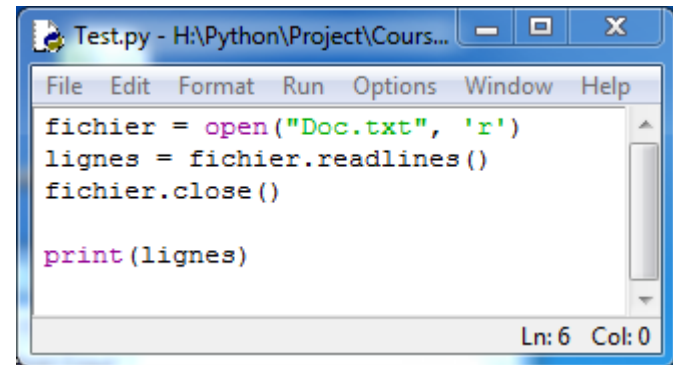
```
fichier = open("Doc.txt", 'r')
lignes = fichier.readlines()
fichier.close()

print(lignes)
```

The status bar at the bottom right indicates 'Ln: 6 Col: 0'.

Ecriture/Lecture fichier .txt

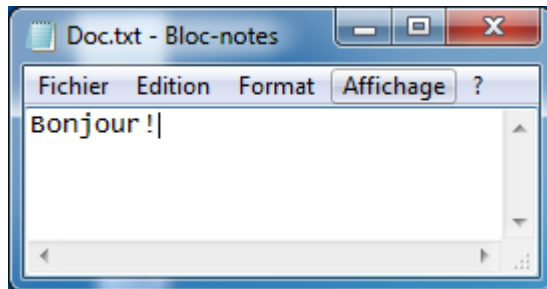
Lecture d'un fichier texte.



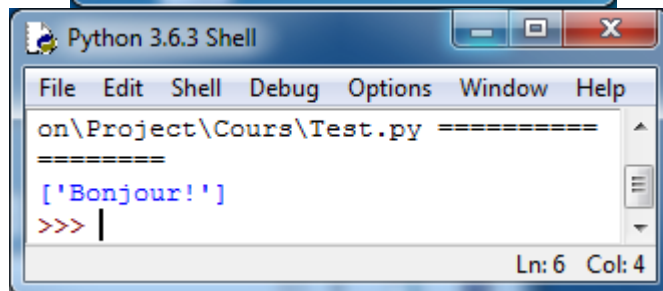
```
Test.py - H:\Python\Project\Cours...
File Edit Format Run Options Window Help
fichier = open("Doc.txt", 'r')
lignes = fichier.readlines()
fichier.close()

print(lignes)
```

Ln: 6 Col: 0

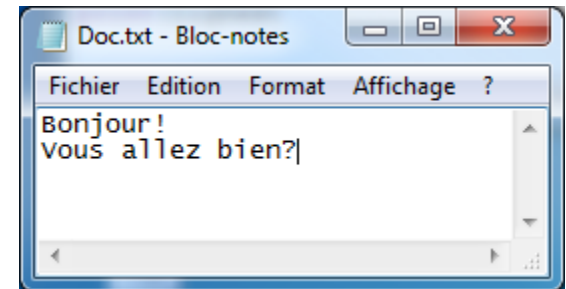


```
Doc.txt - Bloc-notes
Fichier Edition Format Affichage ?
Bonjour!|
```

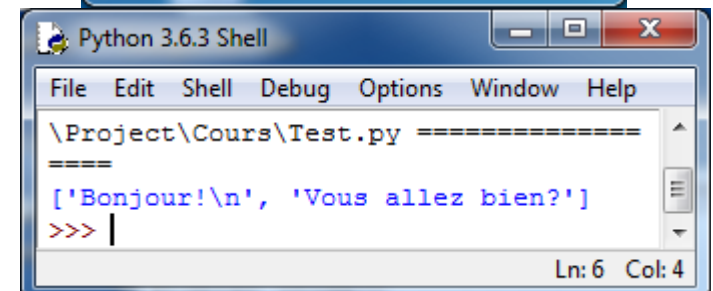


```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
on\Project\Cours\Test.py =====
=====
['Bonjour!']
>>> |
```

Ln: 6 Col: 4



```
Doc.txt - Bloc-notes
Fichier Edition Format Affichage ?
Bonjour!
Vous allez bien?|
```



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
\Project\Cours\Test.py =====
=====
['Bonjour!\n', 'Vous allez bien?']
>>> |
```

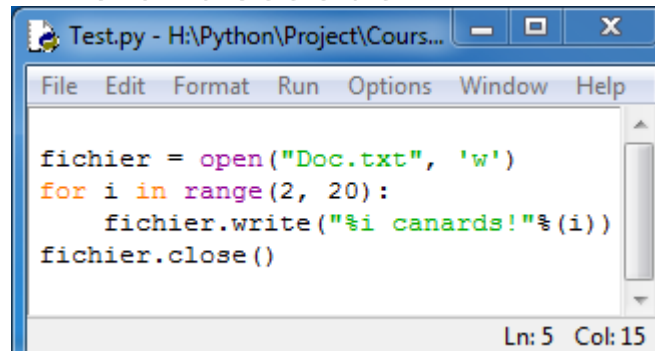
Ln: 6 Col: 4

Ecriture/Lecture fichier .txt

Pour écrire dans un fichier, nous utilisons la même commande (**open**) mais nous utilisons l'argument **'w'** (write).

Attention, lorsque l'on ouvre un fichier de cette manière, on efface toutes les données du fichier.

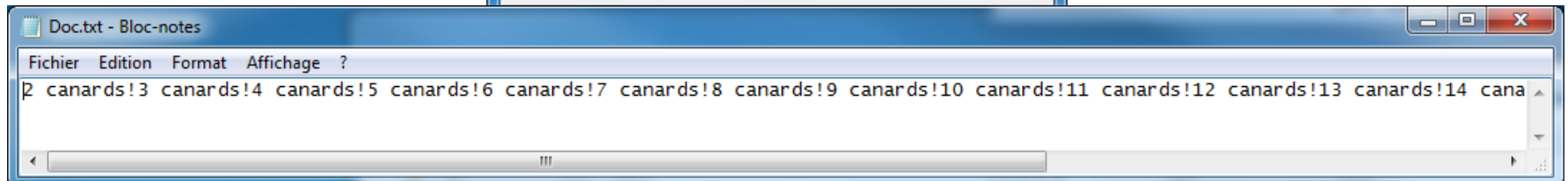
Exemple, le programme ci-dessous:



```
Test.py - H:\Python\Project\Cours...
File Edit Format Run Options Window Help

fichier = open("Doc.txt", 'w')
for i in range(2, 20):
    fichier.write("%i canards!"%(i))
fichier.close()

Ln: 5 Col: 15
```

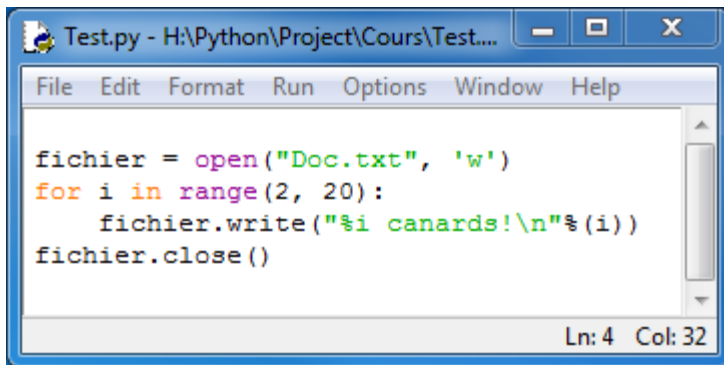


```
Doc.txt - Bloc-notes
Fichier Edition Format Affichage ?

2 canards!3 canards!4 canards!5 canards!6 canards!7 canards!8 canards!9 canards!10 canards!11 canards!12 canards!13 canards!14 cana
```

Ecriture/Lecture fichier .txt

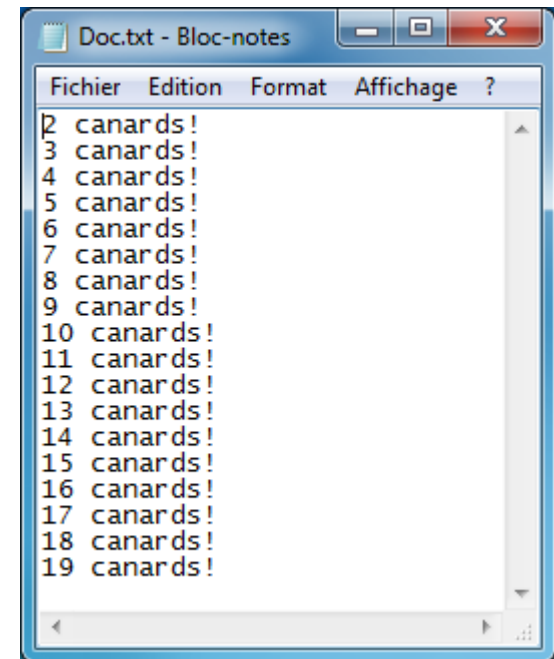
Pour écrire dans un fichier, il est parfois nécessaire d'utiliser les alinéas (`'\t'` [tabulation]) ou les retours à la ligne (`'\n'` [new line]).



The screenshot shows a window titled 'Test.py - H:\Python\Project\Cours\Test...'. The code inside is as follows:

```
fichier = open("Doc.txt", 'w')
for i in range(2, 20):
    fichier.write("%i canards!\n"%i)
fichier.close()
```

The status bar at the bottom indicates 'Ln: 4 Col: 32'.



The screenshot shows a window titled 'Doc.txt - Bloc-notes'. The menu bar includes 'Fichier', 'Edition', 'Format', 'Affichage', and '?'. The content of the file is a list of 18 lines, each containing a number followed by 'canards!':

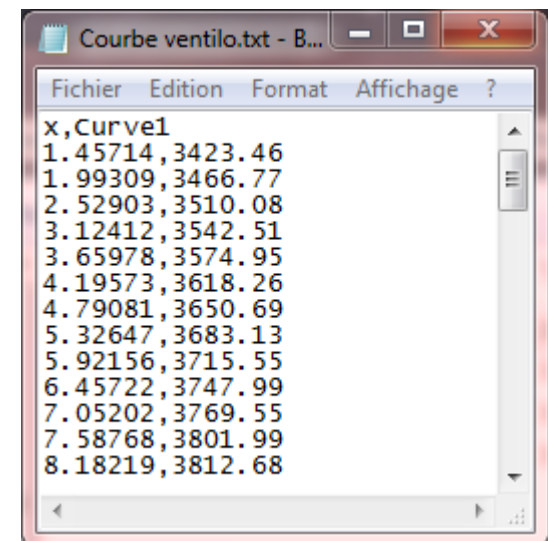
```
2 canards!
3 canards!
4 canards!
5 canards!
6 canards!
7 canards!
8 canards!
9 canards!
10 canards!
11 canards!
12 canards!
13 canards!
14 canards!
15 canards!
16 canards!
17 canards!
18 canards!
19 canards!
```

Exercice – Lecture de fichier

Allez chercher dans les données des exercices le fichier texte « Courbe ventilo ». Vous devez importer les données dans 2 listes (débit et pression).

Le programme devra alors renvoyer la différence de pression aux bornes du ventilateur pour un débit donné.

```
Quel debit voulez-vous tester?
>>>1
Il est impossible de repondre!
>>> ===== RESTART =====
>>>
Quel debit voulez-vous tester?
>>>15
La difference de pression devrait etre entre 3821.6Pa et 3799.7Pa
>>> ===== RESTART =====
>>>
Quel debit voulez-vous tester|
>>>20
La difference de pression devrait etre entre 3385.1Pa et 3352.4Pa
```



Exercice – Lecture de fichier

Allez chercher dans les données des exercices le fichier texte « Fichier_pi » sur le site et créer un programme qui:

- Recherche une suite de chiffre dans pi
- Retourne la/les positions où ce trouve cette suite

```
>>>
Quelle suite de chiffres voulez-vous trouver?
145887
>>> ===== RESTART =
>>>
Quelle suite de chiffres voulez-vous trouver?
12345
Present a la position n°49703
```

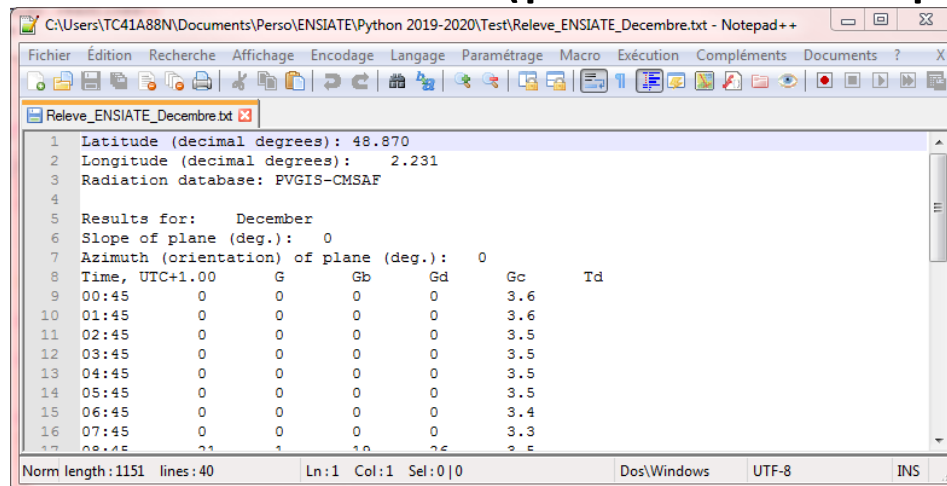
```
>>>
Quelle suite de chiffres voulez-vous trouver?
4321
Present a la position n°23053
Present a la position n°25226
Present a la position n°55569
Present a la position n°64226
Present a la position n°67662
Present a la position n°73162
Present a la position n°73598
Present a la position n°79100
```

Exercice – Lecture de fichier

Allez chercher dans les données des exercices les fichiers texte « Releve_XXX_Decembre ».

A partir de la ligne n°9 et jusqu'à la ligne n°32, vous avez des relevés d'irradiance globale (colonne n°2).

Vous devez réaliser un programme qui compare les irradiances des 3 fichiers (plus haute à plus basse).



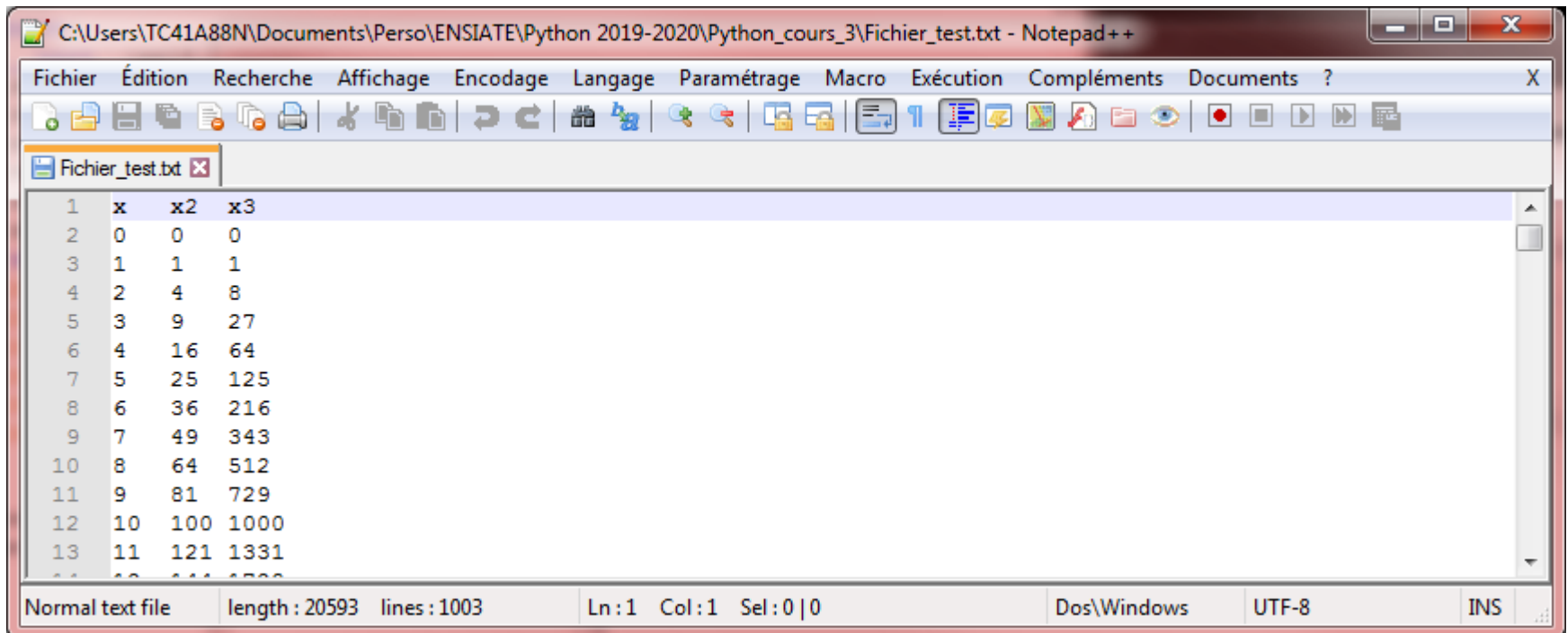
```
C:\Users\TC41A88N\Documents\Perso\ENSIATE\Python 2019-2020\Test\Releve_ENSIATE_Decembre.txt - Notepad++
Fichier  Édition  Recherche  Affichage  Encodage  Langage  Paramétrage  Macro  Exécution  Compléments  Documents  ?  X

Releve_ENSIATE_Decembre.txt
1 Latitude (decimal degrees): 48.870
2 Longitude (decimal degrees): 2.231
3 Radiation database: FVGIS-CMSAF
4
5 Results for: December
6 Slope of plane (deg.): 0
7 Azimuth (orientation) of plane (deg.): 0
8 Time, UTC+1.00  G  Gb  Gd  Gc  Td
9 00:45 0 0 0 0 3.6
10 01:45 0 0 0 0 3.6
11 02:45 0 0 0 0 3.5
12 03:45 0 0 0 0 3.5
13 04:45 0 0 0 0 3.5
14 05:45 0 0 0 0 3.5
15 06:45 0 0 0 0 3.4
16 07:45 0 0 0 0 3.3
17 08:45 0 0 0 0 3.5

Norm length: 1151 lines: 40 Ln: 1 Col: 1 Sel: 0 | 0 Dos\Windows UTF-8 INS
```

Exercice – Ecriture de fichier

Créer un fichier .txt dans lequel vous imprimerez les valeurs de X , X^2 et X^3 séparées par des tabulations.



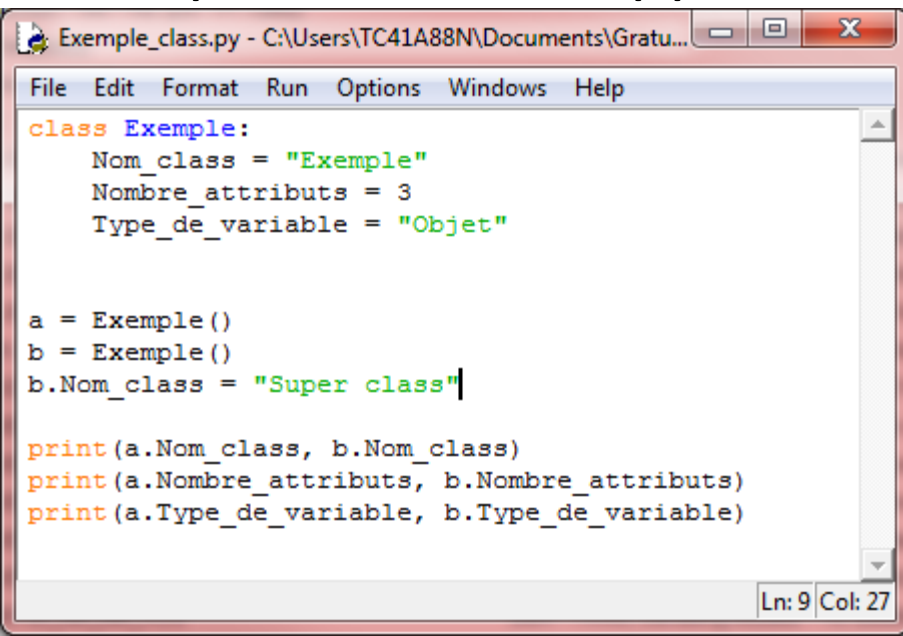
The screenshot shows a Notepad++ window titled "C:\Users\TC41A88N\Documents\Perso\ENSIATE\Python 2019-2020\Python_cours_3\Fichier_test.txt - Notepad++". The menu bar includes Fichier, Édition, Recherche, Affichage, Encodage, Langage, Paramétrage, Macro, Exécution, Compléments, Documents, and ?. The toolbar contains various icons for file operations. The text area displays a table with 13 rows and 4 columns. The first column contains line numbers (1-13), the second column contains values of X (0-11), the third column contains values of X squared (0-121), and the fourth column contains values of X cubed (0-1331). The status bar at the bottom shows "Normal text file", "length : 20593 lines : 1003", "Ln : 1 Col : 1 Sel : 0 | 0", "Dos\Windows", "UTF-8", and "INS".

	x	x2	x3
1	0	0	0
2	1	1	1
3	2	4	8
4	3	9	27
5	4	16	64
6	5	25	125
7	6	36	216
8	7	49	343
9	8	64	512
10	9	81	729
11	10	100	1000
12	11	121	1331
13			

Class – Création d'objets

Les class permettent de générer des objets indépendants et pouvant interagir avec le programme principal.

Une class peut comprendre plusieurs attributs qui peuvent être appelés dans le programme principal.



```
Exemple_class.py - C:\Users\TC41A88N\Documents\Gratu...
File Edit Format Run Options Windows Help

class Exemple:
    Nom_class = "Exemple"
    Nombre_attributs = 3
    Type_de_variable = "Objet"

a = Exemple()
b = Exemple()
b.Nom_class = "Super class"

print(a.Nom_class, b.Nom_class)
print(a.Nombre_attributs, b.Nombre_attributs)
print(a.Type_de_variable, b.Type_de_variable)
```

Ln: 9 Col: 27

```
>>>
('Exemple', 'Super class')
(3, 3)
('Objet', 'Objet')
>>> |
```

La modification d'un attribut d'un objet ne modifie pas les attributs des autres.

Class – Initialisation

Lorsqu'un objet est créé, il est possible d'initialiser certains attributs:

```
class Cube:
    def __init__(self, arete):
        self.surface_face = arete**2
        self.surface_totale = self.surface_face * 6
        self.volume = arete**3

Cube1 = Cube(10)
Cube2 = Cube(5)

print(Cube1.surface_face, Cube2.surface_face)
print(Cube1.surface_totale, Cube2.surface_totale)
print(Cube1.volume, Cube2.volume)
```

```
>>>
(100, 25)
(600, 150)
(1000, 125)
>>>
```

Pour initialiser la class, vous devez nommer votre première fonction « `__init__` ». Le « `self` » correspond à votre objet (ici, « `Cube` ») et vous pouvez ajouter des arguments.

Class – Fonctions

Vous pouvez ajouter des fonctions à votre class:

```
class Cube:  
    nom = "Cube"
```

```
    def __init__(self, arete):  
        self.surface_face = arete**2  
        self.surface_totale = self.surface_face * 6  
        self.volume = arete**3
```

```
    def help_me(self):  
        print("Il y a 4 arguments:\nnom\nsurface_face\nsurface_totale\nvolume")
```

```
    def resumer(self):  
        print("Le nom de l'element est %s"%(self.nom))  
        print("La surface d'une face est %.2f"%(self.surface_face))  
        print("La surface totale est %.2f"%(self.surface_totale))  
        print("Le volume est %.2f"%(self.volume))
```

```
Cube1 = Cube(10)  
Cube1.nom = "Cube1"  
Cube2 = Cube(5)  
Cube2.nom = "Cube2"
```

```
Cube1.help_me()  
Cube1.resumer()  
Cube2.resumer()
```

```
>>>  
Il y a 4 arguments:  
nom  
surface_face  
surface_totale  
volume  
Le nom de l'element est Cube1  
La surface d'une face est 100.00  
La surface totale est 600.00  
Le volume est 1000.00  
Le nom de l'element est Cube2  
La surface d'une face est 25.00  
La surface totale est 150.00  
Le volume est 125.00  
>>>
```

Class : Exercice

Créer une « class » déterminant les propriétés d'une sphère à l'aide du rayon.

Les attributs devront être le rayon et:

- Surface ($S = 4 \cdot \pi \cdot R^2$)
- Volume ($V = \frac{4}{3} \pi \cdot R^3$)
- Périmètre ($P = \pi \cdot R^2$)
- Arrête du cube inscrit ($A = \frac{2 \cdot R}{\sqrt{3}}$)
- Volume du cube inscrit ($V_c = A^3$)

```
>>>
Les rayon est : 2.50
La surface est : 78.54
Le volume est : 65.45
Le perimetre est : 19.63
Le rayon du cube inscrit est : 2.89
Le volume du cube inscrit est : 24.06
>>>
```

Une fonction devra résumer les valeurs des attributs de la class.