Le langage informatique Python 3

Mathias RAGOT 1°6

Table des matières

[A ) Premiers pas avec Python : 2](#_Toc193969053)

[A-1 ) Calculer avec Python : 2](#_Toc193969054)

[A-2 ) Données et variables : 2](#_Toc193969055)

[A-3 ) Noms de variables et mots réservés : 2](#_Toc193969056)

[A-4 ) Affectation ou assignation : 3](#_Toc193969057)

[A-5 ) Afficher la valeur d'une variable : 3](#_Toc193969058)

[A-6 ) Typage des variables : 3](#_Toc193969059)

[A-7 ) Affectations multiples : 4](#_Toc193969060)

[2.1 4](#_Toc193969061)

[2.2 5](#_Toc193969062)

[A-8 ) Opérateurs et expressions : 5](#_Toc193969063)

[2.3 6](#_Toc193969064)

[A-9 ) Priorité des opérations : 6](#_Toc193969065)

[A-10 ) Composition : 6](#_Toc193969066)

[B ) Contrôle du flux d'exécution : 7](#_Toc193969067)

[B-1 ) Séquence d'instructions : 7](#_Toc193969068)

[B-2) Sélection ou exécution conditionnelle : 7](#_Toc193969069)

[B-3 ) Opérateurs de comparaison : 9](#_Toc193969070)

[B-4 ) Instructions composées – blocs d'instructions : 10](#_Toc193969071)

[B-5 ) Instructions imbriquées : 10](#_Toc193969072)

[B-6 ) Quelques règles de syntaxe Python : 11](#_Toc193969073)

[a ) Les limites des instructions et des blocs sont définies par la mise en page : 11](#_Toc193969074)

[b ) Instruction composée : en-tête, double point, bloc d’instructions indenté : 11](#_Toc193969075)

[c ) Les espaces et les commentaires sont normalement ignorés : 11](#_Toc193969076)

[C ) Instructions répétitives : 12](#_Toc193969077)

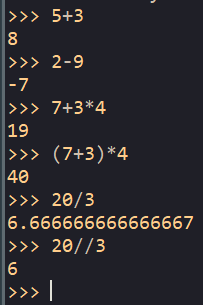
[C-1 ) Réaffectation : 12](#_Toc193969078)

Dans le cadre de la spécialité NSI, nous commençons l’apprentissage du langage de programmation python.

Le programme a été créé par un informaticien néerlandais Guido van Rossum. Le programme est sorti en 1991 et tien son nom de la troupe d’humoriste britanniques : les Monty python

# A ) Premiers pas avec Python :

## A-1 ) Calculer avec Python :



Python utilise les symbole +, – et le parenthèse () de la même facon que en mathématiques

Il utilise \* pour faire des multiplication, / pour les division et // pour obtenire le quotient

## A-2 ) Données et variables :

Les variable sont nommé avec a peut près n’importe quelle nom mais pour l’ordinateur cela correspond a une adresse ou le programme poura récupérer les information si code le lui demande

## A-3 ) Noms de variables et mots réservés :

Sous Python, les noms de variables doivent en outre obéir à quelques règles simples :

* Un nom de variable peut utiliser des lettres de (a → z , A → Z) et des chiffres (0 → 9), mais doit toujours commencer par une lettre.
* Les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que $, #, @, etc. sont interdits, à l’exception du caractère \_ (souligné).
* La casse est significative (les caractères majuscules et minuscules sont distingués). Attention : Joseph, joseph, JOSEPH sont donc des variables différentes.

, il faut encore ajouter que vous ne pouvez pas utiliser comme nom de variables les 33 « mots réservés » suivant

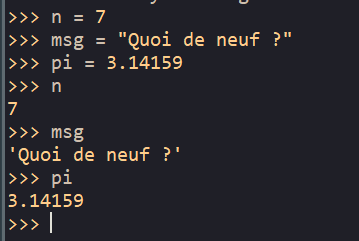
and del from None True as elif global nonlocal try assert else if not while break except import or with class False in pass yield continue finally is raise def for lambda return

## A-4 ) Affectation ou assignation :

Le signe = permet d’assigner une valeur a une variable

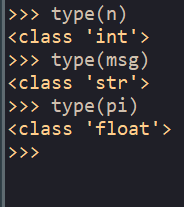
## A-5 ) Afficher la valeur d'une variable :

Le texte explique l’affectation des variables en Python, qui consiste à associer un nom à une valeur en mémoire.



## A-6 ) Typage des variables :

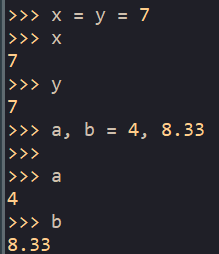
Python utilise un typage dynamique, ce qui signifie que les variables adoptent automatiquement le type correspondant à leur valeur sans déclaration préalable



## A-7 ) Affectations multiples :

Sous Python, on peut assigner une valeur à plusieurs variables simultanément.

On peut aussi effectuer des affectations parallèles à l’aide d’un seul opérateur



Dans cet exemple, les variables a et b prennent simultanément les nouvelles valeurs 4 et 8,33.

### 2.1

>>> largeur = 20

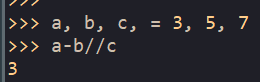
>>> hauteur = 5 \* 9.3

>>> largeur \* hauteur

930

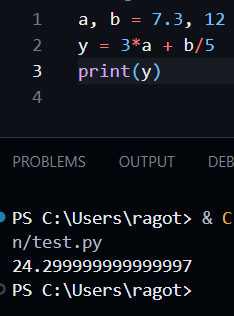
Dans cette exeple on associ 20 a la valeur argeur puis on associe la multiplication 5 multiplier par 9.3 puis on multiplie les deux variable soit 20 \* (5\*9.3)=930

### 2.2



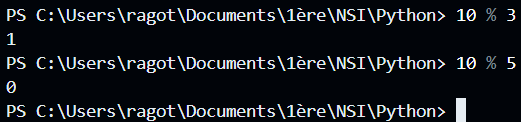
On associe chaque chiffre a une letre donc ca done la calcule 5 division 7 se qui done 0 puis on

## A-8 ) Opérateurs et expressions :



Dans cet exemple, nous commençons par affecter aux variables a et b les valeurs 7,3 et 12. Comme déjà expliqué précédemment, Python assigne automatiquement le type « réel » à la variable a, et le type « entier » à la variable b

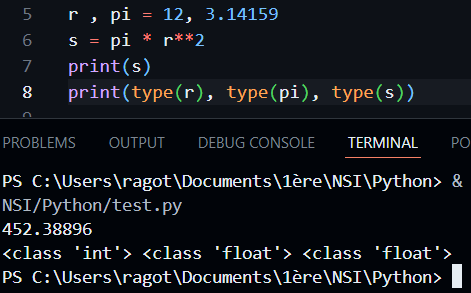
La seconde ligne de l’exemple consiste à affecter à une nouvelle variable y le résultat d’une expression qui combine les opérateurs \* , + et / avec les opérandes a, b, 3 et 5.



Signalons au passage la disponibilité de l’opérateur modulo, représenté par le caractère typographique %. Cet opérateur fournit le reste de la division entière d’un nombre par un autre. Essayez par exemple :

Cet opérateur vous sera très utile plus loin, notamment pour tester si un nombre a est divisible par un nombre b. Il suffira en effet de vérifier que a % b donne un résultat égal à zéro

### 2.3



Dans cette exercice on défini r a 12 et pi a sa valeur (3.14159), puis on multipli pi par r au carré se qui done 3.12159 \* 12² = 452.38896

La fonction type peret de doner le type d’un variable ici la variable r est un variable int pi est float et s est loat aussi

## A-9 ) Priorité des opérations :

Cours de math

## A-10 ) Composition :

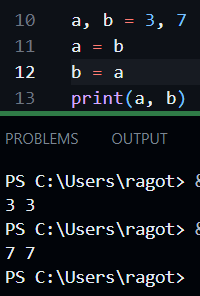
En résumer python permet de donner plusieurs instruction en une seul et le signe = est un signe d’affectation et non dégalité

A=a+1

M+1=b

# B ) Contrôle du flux d'exécution :

## B-1 ) Séquence d'instructions :

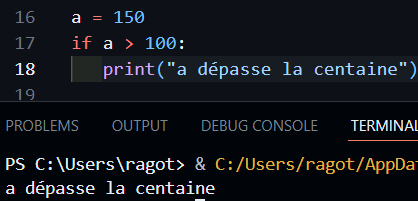


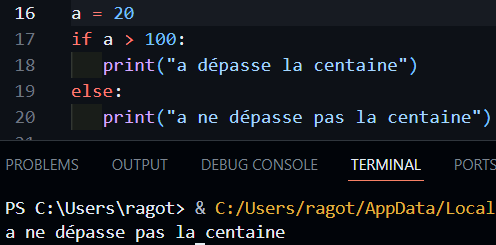
il faut être extrêmement attentif à l’ordre dans lequel vous placez vos instructions les unes derrière les autres.

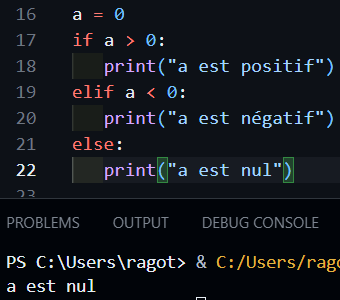
## B-2) Sélection ou exécution conditionnelle :

En programmation, les instructions conditionnelles permettent de modifier le déroulement d’un programme en fonction de certaines conditions. En Python, l’instruction if est utilisée pour exécuter un bloc de code uniquement si une condition est vraie. Si la condition est fausse, on peut utiliser else pour définir une alternative. L’instruction elif (abréviation de "else if") permet de gérer plusieurs cas successifs.

Les opérateurs de comparaison, comme == (égalité), != (différent de), > (supérieur) et < (inférieur), sont essentiels pour formuler ces conditions. Il est important de distinguer = (affectation) de == (comparaison). Par exemple, un programme peut vérifier si un nombre est positif, négatif ou nul et afficher un message approprié en fonction du résultat.

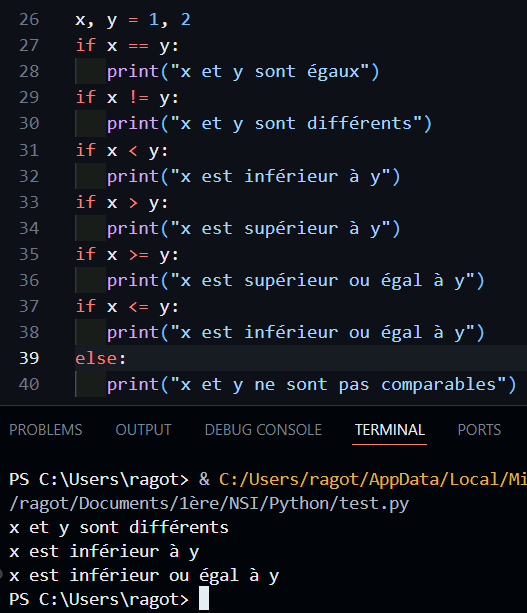


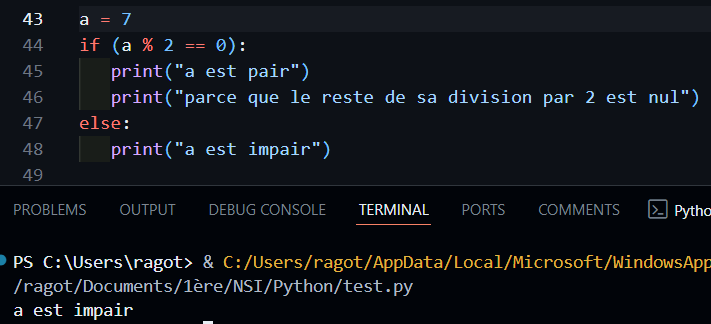




## B-3 ) Opérateurs de comparaison :

La condition évaluée après l’instruction if peut contenir les opérateurs de comparaison suivants :

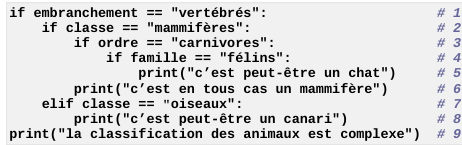




## B-4 ) Instructions composées – blocs d'instructions :

La construction que vous avez utilisée avec l’instruction if est votre premier exemple d’instruction composée. Vous en rencontrerez bientôt d’autres. Sous Python, les instructions composées ont toujours la même structure : une ligne d’en-tête terminée par un double point, suivie d’une ou de plusieurs instructions indentées sous cette ligne d’en-tête.

## B-5 ) Instructions imbriquées :



Il est parfaitement possible d’imbriquer les unes dans les autres plusieurs instructions composées, de manière à réaliser des structures de décision complexes.

Pour que la phrase « c’est en tous cas un mammifère » soit affichée, il faut et il suffit que les deux premières conditions soient vraies. L’instruction d’affichage de cette phrase (ligne 4) se trouve en effet au même niveau d’indentation que l’instruction : if ordre == "carnivores": (ligne 3). Les deux font donc partie d’un même bloc, lequel est entièrement exécuté si les conditions testées aux lignes 1 et 2 sont vraies. Pour que la phrase « c’est peut-être un canari » soit affichée, il faut que la variable embranchement contienne « vertébrés », et que la variable classe contienne « oiseaux ».

## B-6 ) Quelques règles de syntaxe Python :

### a ) Les limites des instructions et des blocs sont définies par la mise en page :

En Python, chaque instruction se termine automatiquement par un saut de ligne, contrairement à d'autres langages où un caractère spécifique, comme le point-virgule, est requis. Les commentaires débutent par # et sont ignorés lors de l’exécution.

### b ) Instruction composée : en-tête, double point, bloc d’instructions indenté :

Nous aurons de nombreuses occasions d’approfondir le concept de « bloc d’instructions » et de faire des exercices à ce sujet dès le chapitre suivant. Le schéma ci-contre en résume le principe.

• Les blocs d’instructions sont toujours associés à une ligne d’en-tête contenant une instruction bien spécifique (if, elif, else, while, def, etc.) se terminant par un double point.

• Les blocs sont délimités par l’indentation : toutes les lignes d’un même bloc doivent être indentées exactement de la même manière (c’est-à-dire décalées vers la droite d’un même nombre d’espaces). Le nombre d’espaces à utiliser pour l’in dentation est quelconque, mais la plupart des programmeurs utilisent des multiples de 4.

• Notez que le code du bloc le plus externe (bloc 1) ne peut pas lui-même être écarté de la marge de gauche (il n’est imbriqué dans rien).

### c ) Les espaces et les commentaires sont normalement ignorés :

À part ceux qui servent à l’indentation, en début de ligne, les espaces placés à l’intérieur des instructions et des expressions sont presque toujours ignorés (sauf s’ils font partie d’une chaîne de caractères). Il en va de même pour les commentaires : ceux-ci commencent toujours par un caractère dièse (#) et s’étendent jusqu’à la fin de la ligne courante.

L’une des tâches que les machines font le mieux est la répétition sans erreur de tâches identiques. Il existe bien des méthodes pour programmer ces tâches répétitives. Nous allons commencer par l’une des plus fondamentales : la boucle de répétition construite autour de l’instruction while.

Dans le programme suivant, les commentaires après le symbole # ne sont pas lues.  
# Programme de Test  
a= 3 # Variable a  
if a == 3 : # Condition de test  
print("NSI")  
else ;  
print("Informatique")  
# permet d'écrire un commentaire sur une seule ligne. Pour écrire un commentaire sur  
plusieurs lignes, il faut l'encadrer par """ et """.  
Exemple d'utilisation d'un commentaire sur plusieurs lignes avec """ :  
""" Programme de Test  
sur la variable a """  
a= 3  
""" == correspond en informatique  
à l'égalité mathématique = """  
if a == 3 :   
print("NSI")  
else ;  
print("Informatique")

# C ) Instructions répétitives :

## C-1 ) Réaffectation :

il est permis de réaffecter une nouvelle valeur à une même variable, autant de fois qu’on le souhaite.