|  |  |  |
| --- | --- | --- |
| Numérique et Sciences Informatiques | | |
| 2h | **Opérateurs et expressions booléennes** |  |
| **Objectif** : connaitre les opérateurs booléens et les expressions booléennes. | | |
| **Matériel**: Python | | |

**Historique**

A partir de 1847, le britannique **George Boole** propose un mode de calcul permettant de traduire des raisonnements logiques par des opérations algébriques (**algèbre de Boole**).

Il crée ainsi une branche des mathématiques qui définit des opérations dans un ensemble qui ne contient que deux éléments pouvant être notés :

* 1 et 0 : logique
* **True et False : booléen en programmation**
* Ouvert et Fermé : électronique

En 1938, l'américain **Claude Shannon** démontre que les circuits électriques peuvent réaliser tous les problèmes de l'algèbre de Boole.

En 1936, avec les travaux du britannique **Alan Turing**, sont posé les fondements de ce qui deviendra l'informatique.

**Les opérateurs booléens**

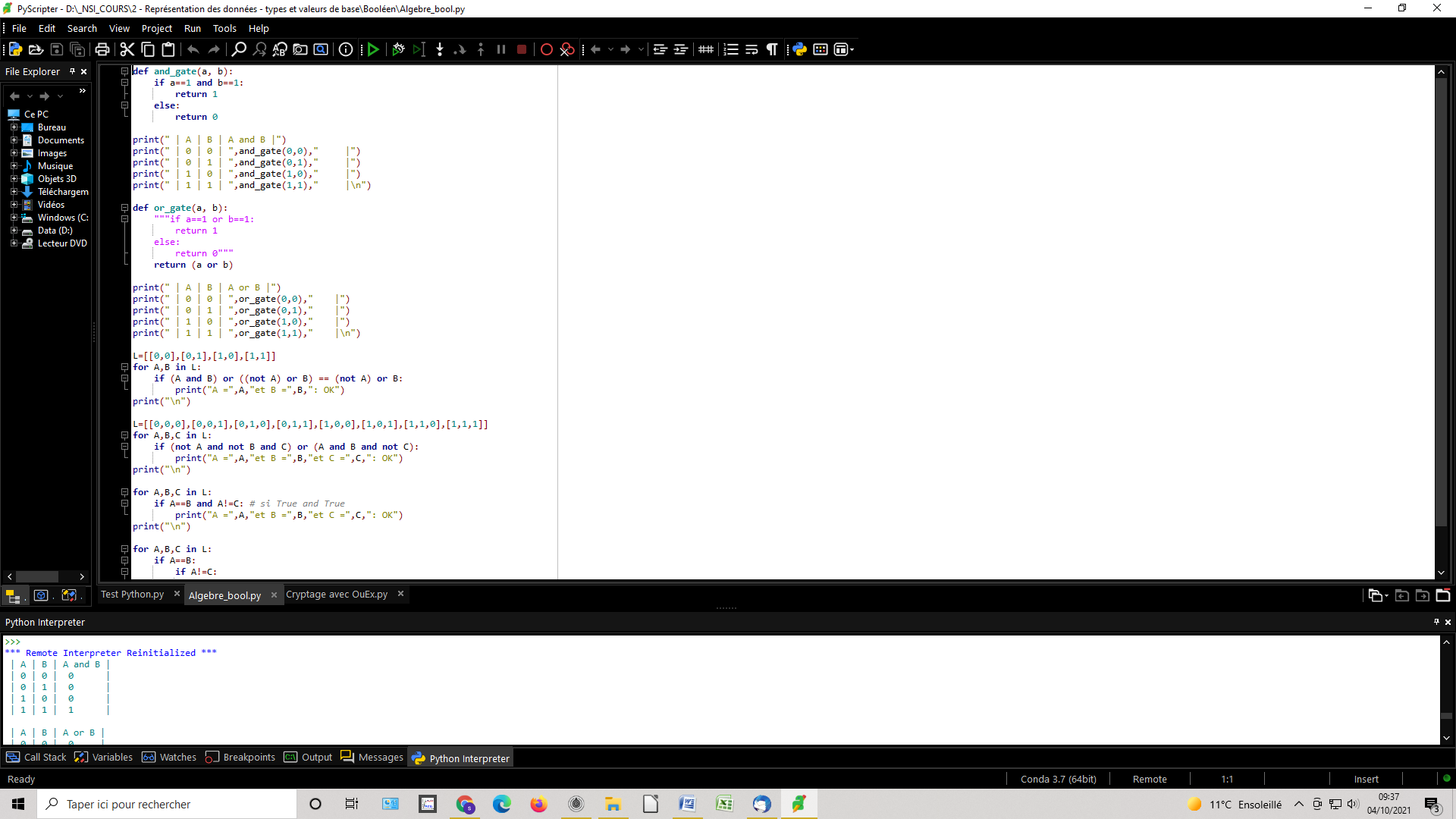
Vous connaissez les **opérateurs** (+, -, \*, /) ainsi que les **comparateurs** (==, !=, >, >=, <, <=) qu'on peut utiliser dans toutes les bases.

Il existe également des **opérateurs booléens** (not, and, or, xor) qu'on ne peut utiliser qu'en base 2 (sur des booléens).

Tables de vérité des **opérateurs booléens** :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **E1** | **E2** | **S = E1 and E2** | |  |  |  | |  |  |  | |  |  |  | |  |  |  | | |  |  |  | | --- | --- | --- | | **E1** | **E2** | **S = E1 or E2** | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |
| |  |  | | --- | --- | | **E** | **S = not E** | |  |  | |  |  | | |  |  |  | | --- | --- | --- | | **E1** | **E2** | **S = E1 xor E2**  xor (e**x**clusive **or**) =  ou exclusif | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |

* Réalisez un programme qui affiche les tables de vérité du **and** et du **or** sous la forme suivante (utilisez les opérateurs **And** et **or**) :



Remarque : dans la plupart des langages de programmation, il est possible d’effectuer des opérations logiques "bit à bit" sur les données.

* Tester les instructions suivantes :

>>> bin(0b0101 & 0b1111)

>>> bin(0b0101 | 0b1111)

>>> bin(0b0101 ^ 0b1111)

**Expressions booléennes**

Une expression booléenne est une [expression](https://fr.wikipedia.org/wiki/Expression_(informatique)) qui donne une valeur [booléenne](https://fr.wikipedia.org/wiki/Booléen) (**True** ou **False**).

Par exemple, la valeur de (5 > 3) est **True**et la valeur de (5 < 4) est **False**.

* Parmi les expressions suivantes, laquelle est **True** ?
* False and (True and False)
* (False or True) and False
* True and (True and False)
* True or (True and False)
* Écrivez les tables de vérité pour les expressions suivantes :
* S1 = ((a > b) and (c == d)) or (e != f)
* S2 = ((a > b) or (c == d)) and (e != f)
* S3 = (a > b) and ((c == d) or (e != f))
* S4 = (a > b) and (c == d) and (e != f)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **a > b** | **c == d** | **e != f** | **S1** | **S2** | **S3** | **S4** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Remarque : python évalue les expressions de la gauche vers la droite mais il est conseillé de mettre des parenthèses pour définir l’expression prioritaire.**

**Exemple :**

* (True or True) and False = False
* True or True and False = True
* Soient a=5, b=4, c=7, d=8, e=3 et f=4. Donnez la valeur de S pour les expressions suivantes :
* S = (a > b) and ((c == d) or (e != f)) → True
* S = ((a > b) or (c == d)) and (e != f)
* S = (a > b) or (c == d) or (e != f)
* S = (a > b) and (c == d) and (e != f)
* Quelle est l’expression booléenne équivalente à S = **a and (a or b)**:
* S = a or b
* S = and b
* S = a
* S = b

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **A or b** | **S** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* Quelle est l’expression booléenne équivalente à S = **a or ((not a) and b)**:
* S = a or b
* S = a and b
* S = a
* s = b

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **a** | **b** | **not a** | **(not a) and b** | **S** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Remplissez le tableau suivant en fonction des expressions suivantes à tester dans la console :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 < 3 | 1 > 3 | 3 == (2 + 1) | 3 == "3" | True or (True and False) | 2022%4==0 | type(True) |
|  |  |  |  |  |  |  |

**Code 1\*\*** : Ecrivez un programme qui indique si l'année entrée par l’utilisateur est bissextile (366 jours).

Une année bissextile respecte les conditions suivantes :

* L’année doit être divisible par 4 mais pas par 100 ou alors divisible par 400.
* L’année doit être postérieure à 1582.

Pour info, la Terre tourne autour du Soleil en 365 jours, 5 heures et 46 minutes.

Le jeudi 4 octobre 1582 a été suivi du vendredi 15 octobre 1582, c’est le passage du calendrier Julien au calendrier Grégorien.

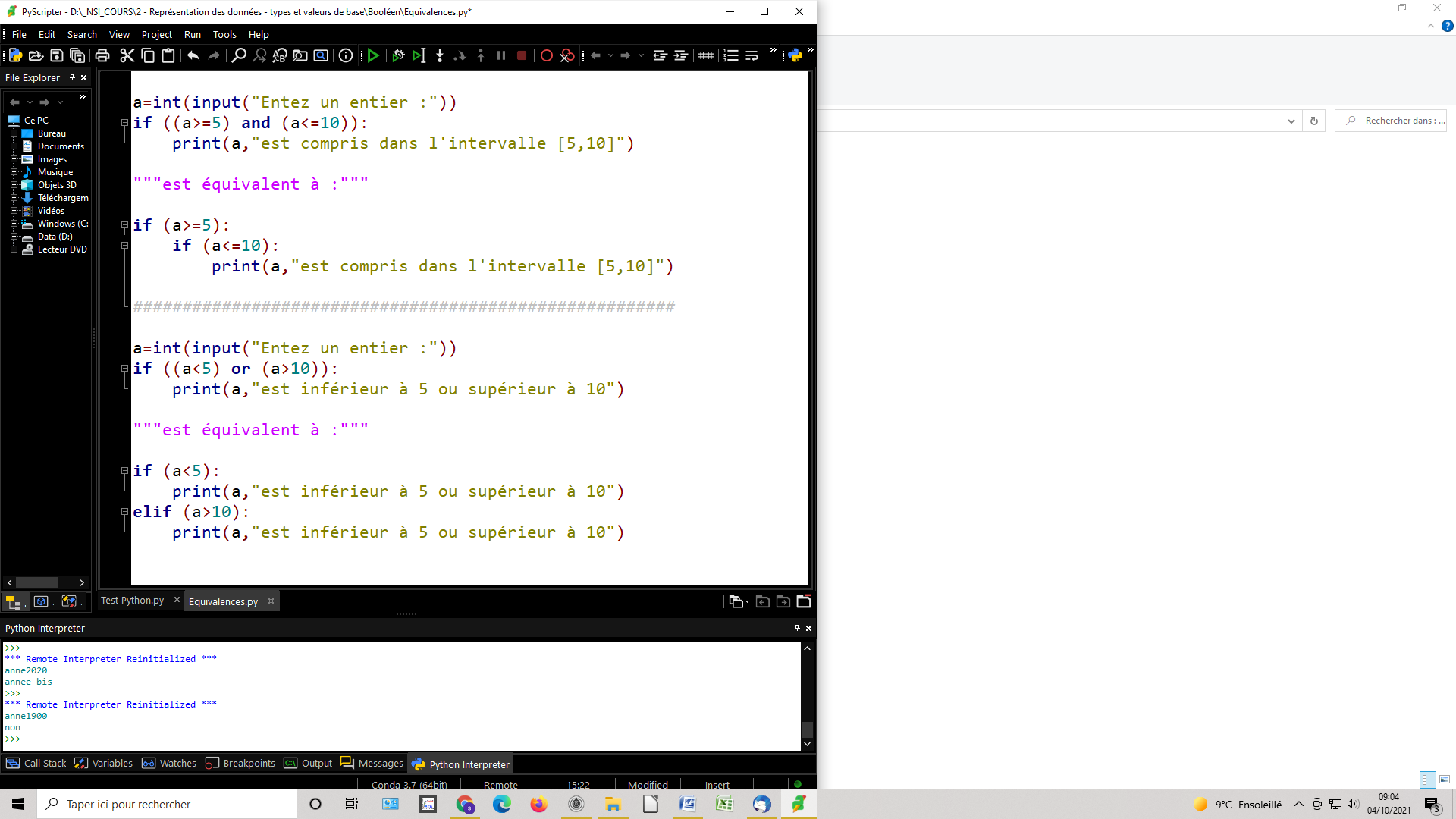
Pour savoir si une année est bissextile : https://www.calendriergratuit.fr/annees-bissextiles.html

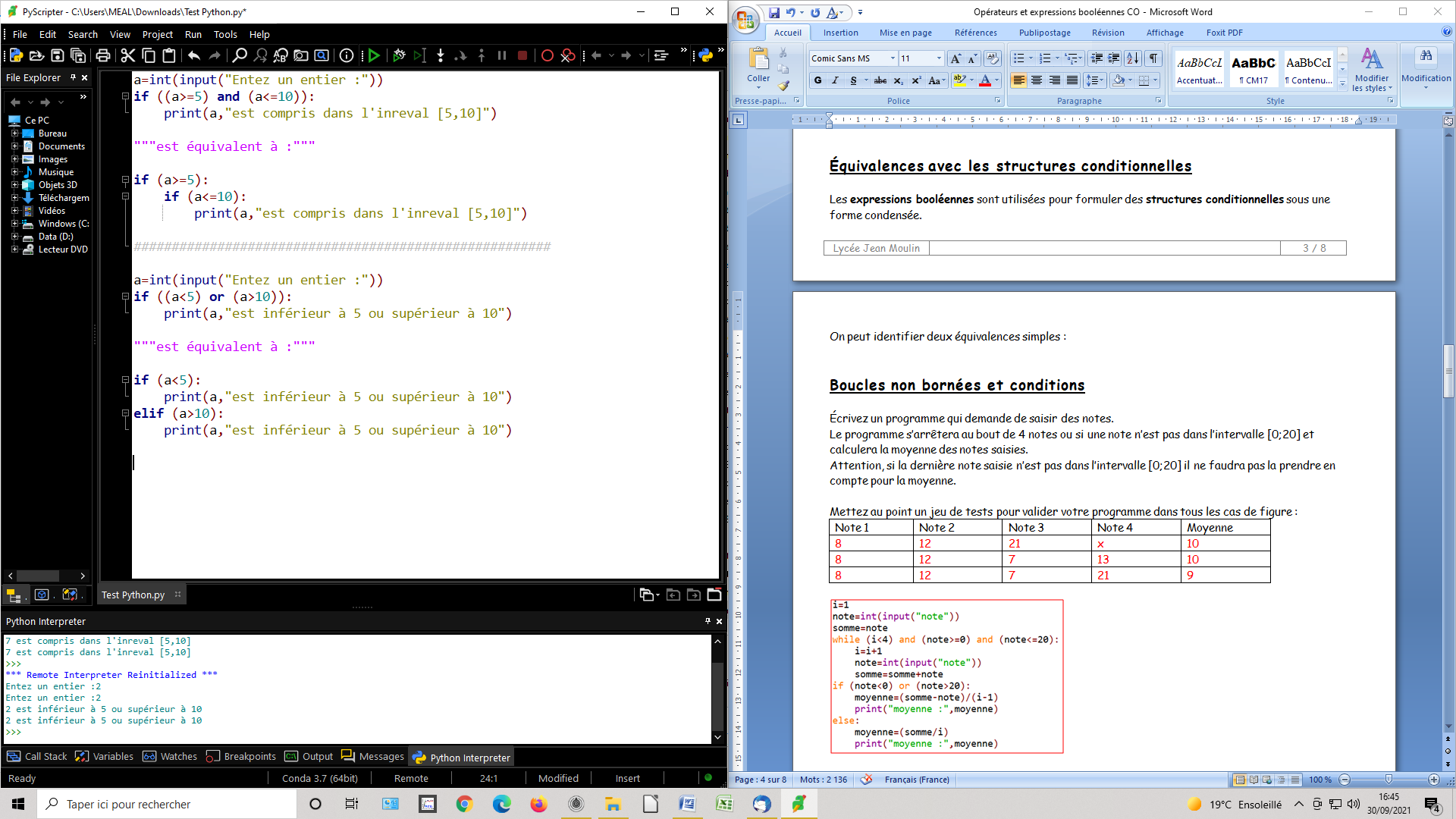
Jeu de test : 1500, 1900, 2016, 2020, 2021.

**Équivalences avec les structures conditionnelles**

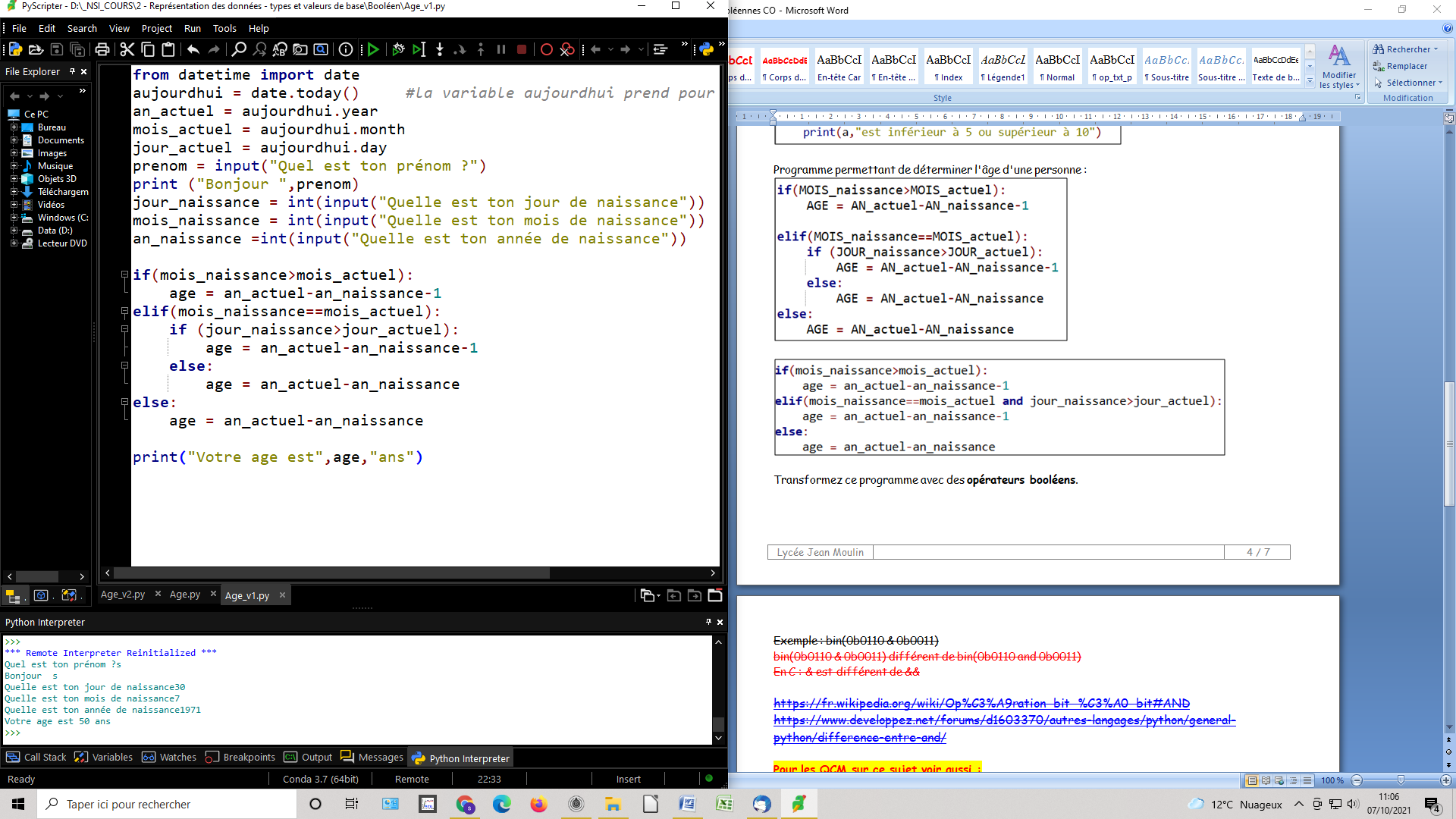
Les **expressions booléennes** sont utilisées pour formuler des **structures conditionnelles** sous une forme condensée.

Voici deux équivalences simples :





Programme permettant de déterminer l'âge d'une personne :



**Code 2\*** : Transformez ce programme avec des **opérateurs booléens**.