Semaine 2

Initiation à l'algorithmique et programmation

Revekka Kyriakoglou

Plan du cours

Operateurs

Opérateur

Les opérateurs sont des symboles qui permettent de manipuler des variables.
Il y a plusieurs types d'opérateurs :

- Opérateurs arithmétiques.
- Opérateurs d'assignation.
- Opérateur d'incrémentation
- Opérateurs de comparaison.
- Opérateurs logiques.

Opérateurs arithmétiques

Opérateur	Dénomination	Effet
+	Addition	Ajoute deux valeurs
-	Soustraction	Soustrait deux valeurs
*	Multiplication	Multiplie deux valeurs
/	Division	Divise deux valeurs
//	Division entière	Divise et arrondit
=	Affectation	Affecte une valeur à une variable

Exemple

- *2+3=5*

Opérateurs d'assignation et opérateurs d'incrémentation

Opérateur d'assignation	Effet
+=	additionne deux valeurs et stocke le résultat dans la variable (à gauche)
-=	soustrait deux valeurs et stocke le résultat
*=	multiplie deux valeurs et stocke le résultat
/=	divise deux valeurs et stocke le résultat

Exemple

Soit int x=5 ;

- *x+=1 ; donne comme résultat 6*
- *x-=1 ; donne comme résultat 4*

Ordre des Opérations

Python suit l'ordre standard des opérations (PEMDAS) :

- Parenthèses.
- Exposants.
- Multiplication et Division (de gauche à droite).
- Addition et Soustraction (de gauche à droite).

Ordre des Opérations

Python suit l'ordre standard des opérations (PEMDAS) :

- Parenthèses.
- Exposants.
- Multiplication et Division (de gauche à droite).
- Addition et Soustraction (de gauche à droite).

Exercise 1

Quelle est la valeur de x dans les exemples donnés ci-dessous ?

- x = (2 + 3) * 2 + 3
- x = 2
  x + = 3

Opérateurs de comparaison

Opérateur	Dénomination	Effet
==	égalité	vérifie l'égalité entre deux valeurs
<	infériorité stricte	vérifie qu'une variable est strictement inférieure à une valeur
<=	infériorité	vérifie qu'une variable est inférieure ou égale à une valeur
>	supériorité stricte	vérifie qu'une variable est strictement supérieure à une valeur
>=	supériorité	vérifie qu'une variable est supérieure ou égale à une valeur
!=	différence	vérifie qu'une variable est différente d'une valeur

Exemple

- *x==3 retourne* `True` *si x est égal à* 3, *sinon* `False`.

Opérateurs logiques

Opérateur	Dénomination	Effet
`or`	OU logique	vérifie qu'une des conditions est réalisée
`and`	ET logique	vérifie que toutes les conditions sont réalisées
`not`	NON logique	inverse l'état d'une variable booléenne

Opérateurs logiques

Opérateur	Dénomination	Effet
`or`	OU logique	vérifie qu'une des conditions est réalisée
`and`	ET logique	vérifie que toutes les conditions sont réalisées
`not`	NON logique	inverse l'état d'une variable booléenne

Exercise 2

Essayer de prévoir le résultat et de l'expliquer.

```
print("1 == 30 vaut ", 1 == 30)
print("1 != 30 vaut", 1 != 30)
print("1 < 30 vaut", 1 < 30)
print("1 > 30 vaut", 1 > 30)
```

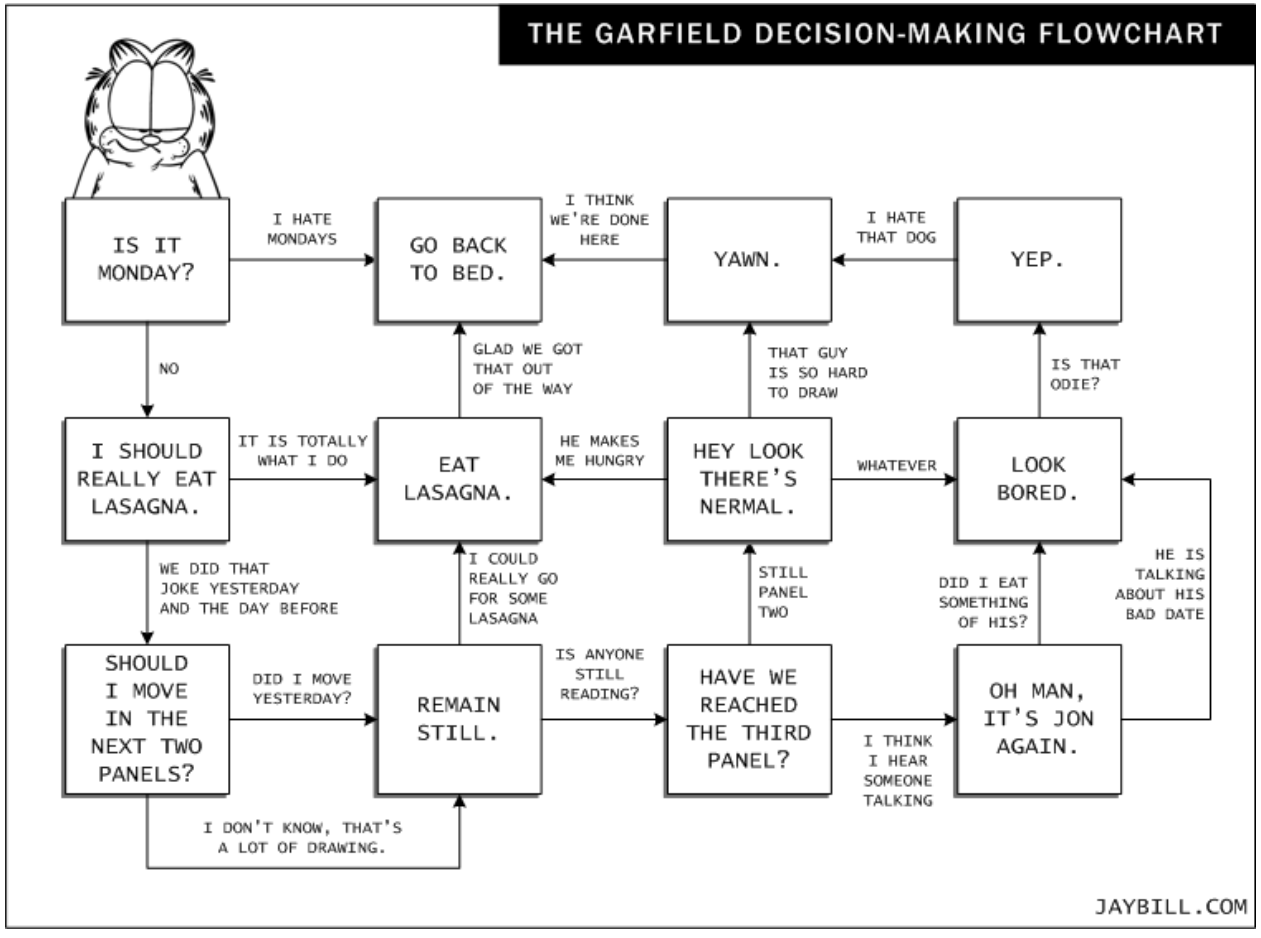
Exercise 3

Essayer de prévoir le résultat et de l'expliquer.

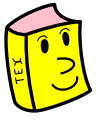
```
is_sunny = True
is_warm = True

is_good_weather = is_sunny and is_warm
is_rainy_day = not is_sunny

print(is_good_weather)
print(is_warm)
```

Structure conditionnelle

On appelle **structure conditionnelle** les instructions qui permettent de tester si une condition est vraie ou non.



Structure conditionnelle



Les structures conditionnelles en Python permettent d'exécuter différentes parties de code en fonction de conditions spécifiques.

La structure conditionnelle de base utilise l'instruction `if` pour tester une condition, et éventuellement `else` et `elif` (abréviation de "else if") pour définir des alternatives.

if-else

L'instruction **if-else** permet d'exprimer des prises de décision.

```
if Condition1:
    # Instruction 1

elif Condition2:
    # Instruction 2

else:
    # Instruction 3
```

La **Condition 1** est évaluée

- Si elle est **VRAI**, l'instruction 1 s'éxecute.
- Si elle est **FAUSSE**, la **Condition 2** est évaluée.
  - Si elle est **VRAI**, l'instruction 2 s'éxecute.
  - Si elle est **FAUSSE**, c'est l'instruction 3 qui s'éxecute.

Chaque **else** s'associe automatiquement au plus proche des if précédents qui n'ont pas encore de else.

```
if ( n > 0 ):
    if ( a > b):
        z = a
    else:
        z = b
```

Sinon il faut :

```
if ( n > 0 ):
    if ( a > b):
        z = a
else
    z = b
```

Exemple

Ce code prend une chaîne de caractères de l'utilisateur et vérifie s'il s'agit du mot "banane".

```
word = input("Entrez un mot: ")

# Vrai si le mot est "banana"
if  (word == "banana"):
    printf("Le mot est banana")
# Sinon
else:
    print("Le mot n'est pas banana")
```