# Initiation à la programmation avec Python (v3)

## Cours n°1

Morgan Rogers

Basé sur le cours de Jean-Vincent Loddo*



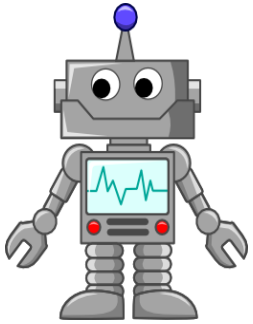
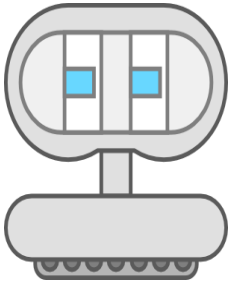
*y compris les blagues..!

# Sommaire du cours n°1

- Notion n°1 : programmer = **automatiser un service**
- Notion n°2 : les **valeurs**
- Notion n°3 : les **variables**
- Notion n°4 : la **conditionnelle** (prise de décision)

# Notion n°1 : programmer = automatiser un service

- On peut imaginer un **programme** comme un **robot**
- Même si il n'a pas un corps


- Et même si son ectoplasme est emprisonné dans une fenêtre graphique ou textuelle (**terminal**) d'un ordinateur

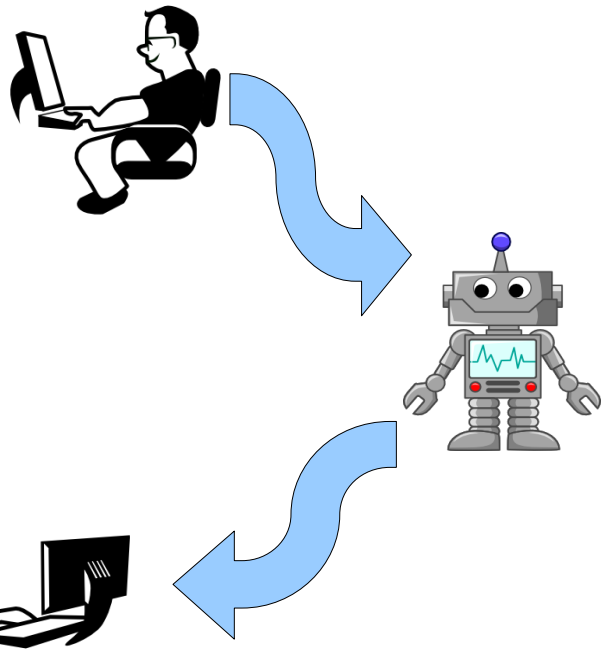
- Comme un robot : il fait un travail, il questionne l'utilisateur, il réagit aux réponses et autres stimuli (clavier, souris, réseau, etc)

- Comme un robot : quelqu'un le construit, quelqu'un l'utilise

# Développeur → Programme → Utilisateur

- Comme un robot : quelqu'un le construit, quelqu'un l'utilise

- Qui le construit ?
  - C'est le **programmeur** (ou **développeur**)
  - Comment : avec un **langage de programmation**
  - Combien de fois : une fois !
  - Pourquoi : parce qu'**il rendra** un service

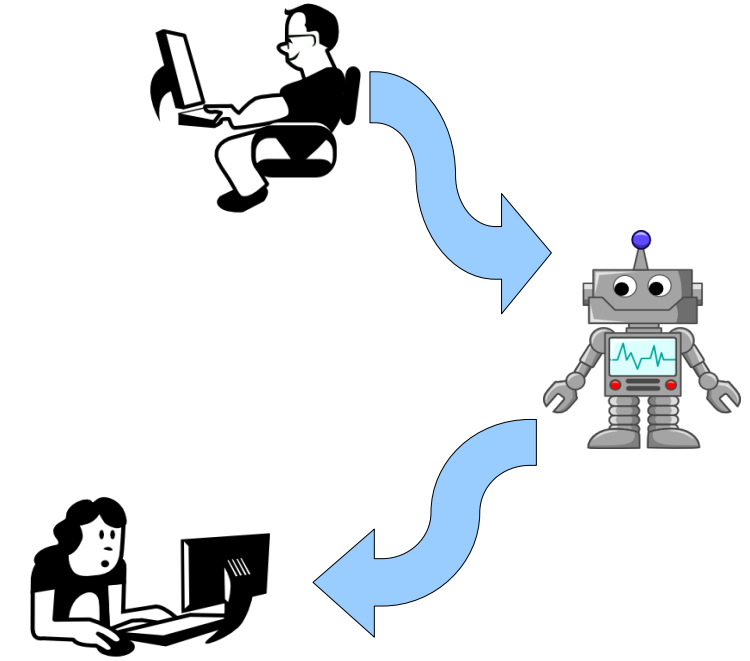


- Qui l'utilise ?
  - C'est l'**utilisateur**
  - Comment : avec une interface (graphique ou textuelle)
  - Combien de fois : autant qu'il le souhaite
  - Pourquoi : parce qu'**il rend** un service

# Développeur → Programme → Utilisateur

- Mais alors, apprendre à programmer c'est apprendre à **automatiser un service** ?

- OUI



# Développeur → Programme → Utilisateur

- Est-ce que le développeur peut être aussi l'utilisateur ?

- Ce n'est pas souvent le cas, mais c'est possible.

- Mais... le développeur doit **tester** si le programme rend effectivement le service qu'on attend de lui (avant de livrer).

- Un peu de terminologie à propos :
  - si le comportement est erroné on dit que le programme a un **bug** (ou **bogue**)
  - l'activité qui consiste à corriger un programme s'appelle **debugging** (ou **debogage**)

# Premier programme (1)

- **Service à rendre** : afficher des citations ou proverbes qui stimulent la réflexion

```
#!/usr/bin/python3
# coding: utf-8
print("Lorsqu'on se cogne la tête contre un pot et que cela sonne creux,")
print("ce n'est pas forcément le pot qui est vide.")
print("Confucius, philosophe.")
print("---")
print("Qui veut faire quelque chose trouve un moyen,")
print("qui ne veut rien faire trouve une excuse.")
print("Proverbe Arabe.")
```

# Premier programme (2)

- **Service à rendre** : afficher des citations ou proverbes qui stimulent la réflexion

*shebang* : en-tête du programme (ou "script").
Ne pas oublier non plus de rendre le fichier exécutable (chmod +x)

```
#!/usr/bin/python3
# coding: utf-8
print("Lorsqu'on se cogne la tête contre un pot et que cela sonne creux,")
print("ce n'est pas forcément le pot qui est vide.")
print("Confucius, philosophe.")
print("---")
print("Qui veut faire quelque chose trouve un moyen,")
print("qui ne veut rien faire trouve une excuse.")
print("Proverbe Arabe.")
```

pour utiliser des caractères accentués

**Remarque** :
ce programme fait appel à l'outil (fonction) **print** sept fois :
le service rendu est donc un
**assemblage** de sous-services rendus
par des outils (sous-programmes) pré-existants

# Premier programme (3)

- **Service à rendre** : afficher des citations ou proverbes qui stimulent la réflexion

- Nous avons rangé la liste des instructions dans le fichier citations.py et nous l'avons rendu exécutable (chmod +x).
Exécution :

```
$ ./citations.py
Lorsqu'on se cogne la tête contre un pot et que cela sonne creux,
ce n'est pas forcément le pot qui est vide.
Confucius, philosophe.
---
Qui veut faire quelque chose trouve un moyen,
qui ne veut rien faire trouve une excuse.
Proverbe Arabe.
```

# Premier programme (4)

- **Service à rendre** : afficher des citations ou proverbes qui stimulent la réflexion

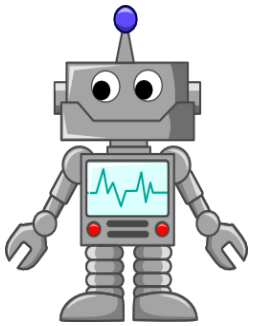
- Nous avons rangé la liste des instructions dans le fichier citations.py et nous l'avons rendu exécutable (chmod +x).
Exécution :

$ ./citations.py

| | (appels à) |
|---|---|
| Lorsqu'on se cogne la tête contre un pot et que cela sonne creux, | **print** n°1 |
| ce n'est pas forcément le pot qui est vide. | **print** n°2 |
| Confucius, philosophe. | **print** n°3 |
| --- | **print** n°4 |
| Qui veut faire quelque chose trouve un moyen, | **print** n°5 |
| qui ne veut rien faire trouve une excuse. | **print** n°6 |
| Proverbe Arabe. | **print** n°7 |

# Notion n°1 : programmer = automatiser un service

- On peut imaginer un **programme** comme un **robot**



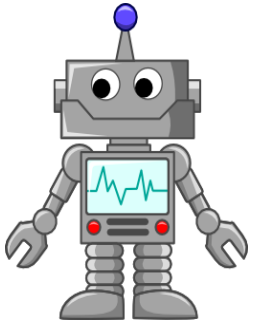
- Ok, ok, mais emprisonné dans la fenêtre d'un ordinateur il ne pourra pas faire le **ménage** ! Ni **repasser le linge** !

- Alors, que peut-il rendre comme service intéressant ?
  - récupérer des informations (sur Internet, dans des fichiers, données par l'utilisateur…),
  - présenter des informations (des proverbes, la date…),
  - calculer des informations

- Autrement dit : un programme **élabore des informations**

# Notion n°1 : programmer = automatiser un service

- Admettons : un programme **élabore des informations**



- Quels types d'information sont traitées ?

  – Textes ? Nombres ? Dates ? Noms de fichiers ? Adresses Internet ? Autre chose ?

  – C'est la notion de **valeur**

  – Et ça dépend du langage de programmation...

# Notion n°2, les informations ou **valeurs** élaborées

- Les **valeurs** d'un langage de programmation sont les informations que les programmes sont capables de manipuler

- Il y en a de plusieurs **types** :
  - Nombres entiers (0 42 -16 100)
  - Nombres flottants (3.14159 2.71828)
  - Booléens (True False)
  - Caractères (a z A Z 0 9 #)
  - Chaînes de caractères (salut HeLLo Confucius, philosophe.)
  - Tableaux, listes, tuples, arbres, dictionnaires, fonctions, objets, ...

# Les **valeurs** en **Python**

- Les **valeurs** d'un langage de programmation sont les informations que les programmes sont capables de manipuler

- Il y en a de plusieurs **types** :
  - Nombres entiers (0 42 -16 100)
  - Nombres flottants (3.14159 2.71828)
  - Booléens (True False)
  - Caractères (a z A Z 0 9 #)
  - Chaînes de caractères (salut HeLLo Confucius, philosophe.)
  - Tableaux, listes, tuples, arbres, dictionnaires, fonctions, objets, ...

type « **int** »

type « **float** »

type « **bool** », attention à la syntaxe : première lettre de True et False en **majuscule**

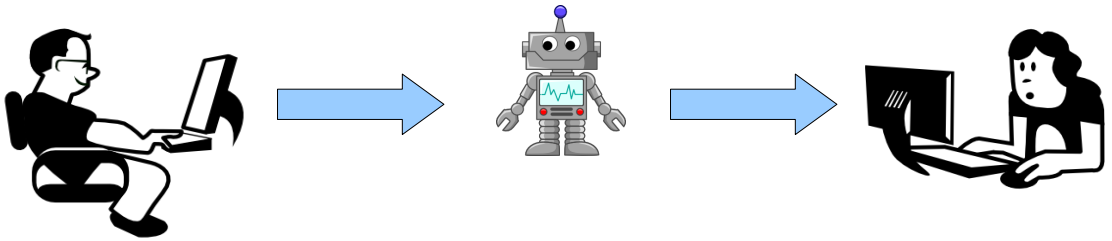
ce type de base n'existe pas en Python

type « **str** »

tout y est en Python !

# Résumé pour l'instant et prochaine question...

- Un programme est comme un robot sauf que son job est de manipuler des informations, par exemple des chaînes de caractères
- Un programmeur le construit, un utilisateur l'utilise

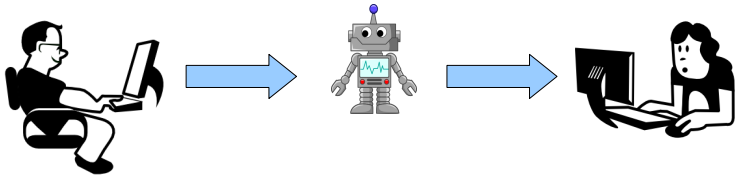


- Parce que cela rend service
- Très bien mais on entrevoit un petit problème :
  - comment programmer le comportement du robot
    **sans savoir ni quand**,
    ni dans **quelle circonstances**,
    ni par **qui**, ni **comment** il sera utilisé ?

# Programmer c'est aussi prévoir sans tout savoir

- Question posée :

Comment programmer le comportement du robot **sans savoir ni quand**, ni dans **quelle circonstances**, ni par **qui**, ni **comment** il sera utilisé ?



- Exemples :

  - Programmer un **navigateur Internet** : quelles pages seront visitées ? Combien en même temps ? À quelles dates ? Avec quel OS ? Avec quelles technologies (Html, Javascript, Java, ...) ?

  - Programmer un **traitement de texte** : quel texte ? Quelle langue ? Quels caractères ? Combien de pages ? Quels effets (gras, italique, souligné,...) ?

  - (plus simple) Programmer un robot qui **calcule le double** : de quel nombre ?

  - (plus simple) Programmer l'**affichage** d'un message d'anniversaire :
    « Joyeux Anniversaire mon cher Emmanuel, felicitations pour vos 42 ans! »
    Quel sera le nom (à la place de Emmanuel) ? Et si c'était Brigitte ?
    Quel sera l'age (à la place de 42) ?

# Notion n°3, les **variables**

- Pour traiter l'information que le programmeur connaît mais surtout celle qu'il **ne connaît pas**, les langages de programmation proposent les « variables »

- Les variables sont des boîtes qui ont un **nom** et un **contenu**


- Le contenu est une **valeur**, c'est-à-dire une information traitée par le langage de programmation

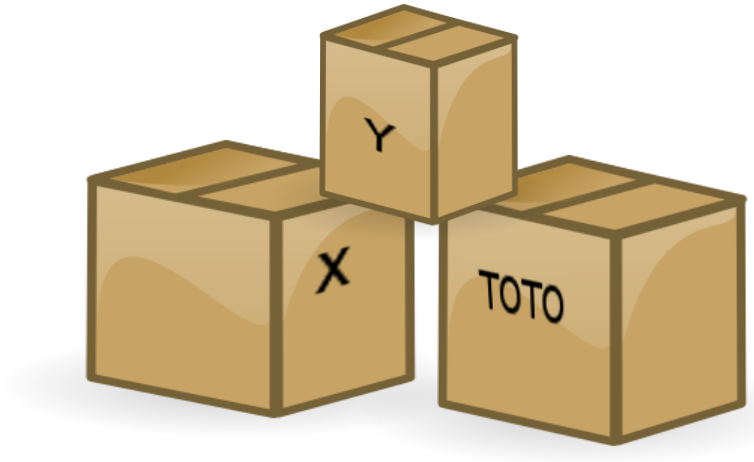
- Comment on stocke une information (valeur) dans une boîte ?

- Comment on la récupère ?

# **Variables** : comment on stocker de l'information dans une variable ?

- Par **affectation** du contenu :

```
TOTO = "salut le monde"
Y = 16
X = 3.14159
```



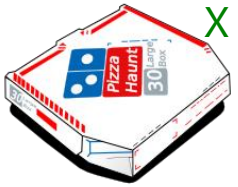

- Par lecture des caractères saisis au clavier par l'utilisateur.

En C ou Java cela se fait avec **scanf**, en Bash avec **read**, en Python (v3) cela se fait avec **input** et toujours par l'affectation :

```
X = input("Votre nom ? ")
Y = input("Votre age ? ")
```

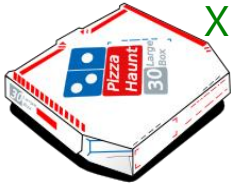
# **Variables** : comment on récupère l'information stockée dans une variable ?

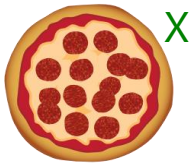
- Dans certains langages le contenu de la boîte **X** est indiqué par **$X** (Bash, PHP,...) :





- En Python, comme en math et comme dans beaucoup d'autres langages, le **nom** de la boîte peut indiquer aussi le **contenu**, cela dépend du contexte.
Par exemple:



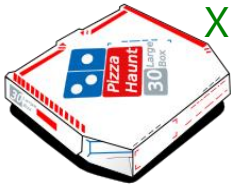

```
TOTO = "La réponse"
X = 12
Y = 30 + X
print(TOTO,"à la question ultime est",Y)
```

– Ce programme affiche :

La réponse à la question ultime est 42

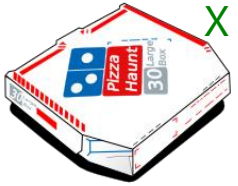
# **Variables** : comment on récupère l'information stockée dans une variable ?

- Dans certains langages le contenu de la boîte **X** est indiqué par **$X** (Bash, PHP,...) :

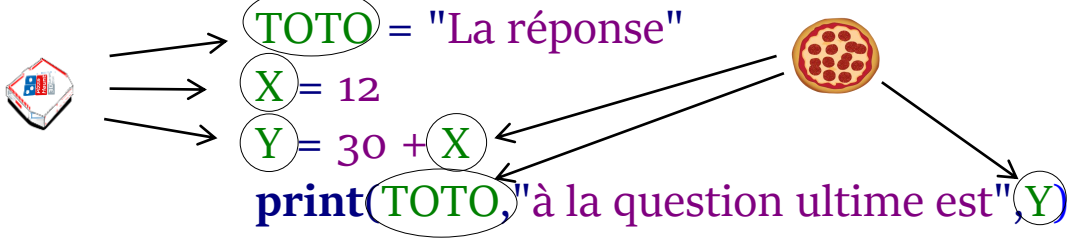




- En Python, comme en math et comme dans beaucoup d'autres langages, le **nom** de la boîte peut indiquer aussi le **contenu**, cela dépend du contexte.
Par exemple:








– Ce programme affiche :

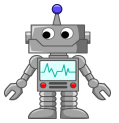
La réponse à la question ultime est 42

# **Variables** : remarque sur les informations manipulées

- Le **développeur** programme (écrit) cette ligne en **1999** :

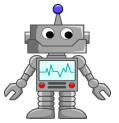
```
X = input("Votre nom ? ")
```

- Un **robot** (une instance du programme) est exécuté en **2017**



- L'**utilisateur** saisi une chaîne de caractères en **2017**

- Le **robot** stocke cette chaîne dans la variable X en **2017**



- Le **développeur**, toujours en **1999**, ne connaît pas la chaîne saisie
  - mais il sait qu'elle se trouve dans X
  - il peut donc y accéder en écrivant X dans la suite du programme
- Le développeur planifie le traitement des informations qu'il **connaît** et des informations qu'il **ne connaît pas**

# Le petit robot bien gentil

- **Service à rendre** : le robot doit demander le nom et l'âge de l'utilisateur et lui écrire ensuite une gentillesse du style

  « Joyeux Anniversaire mon cher ..., felicitations pour vos ... ans! »

- Comment le programmeur peut « remplir » les pointillés, c'est-à-dire manipuler l'information qu'il ne connaît pas ?
- C'est simple, il peut citer cette information sans la connaître :

```
NOM = input("Votre nom ?")
AGE = input("Votre âge ?")
print(" Joyeux Anniversaire mon cher ",NOM,", felicitations pour vos ",AGE," ans!")
```

- Le programmeur sait qu'une certaine information est dans une certaine boîte.
Cela suffit pour faire des calculs, pour afficher ou pour prendre des décisions !

# Un robot pour tout âge

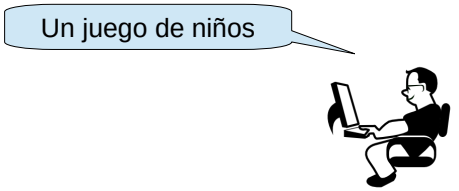
- Supposons à présent de vouloir écrire une phrase différente en fonction de l'âge.

- **Service à rendre** :
  - Si l'utilisateur a moins de 13 ans le robot devra écrire « Alors vous jouez à Call of Duty »
  - Sinon il écrira « Alors vous jouez à Pokemon »

- Comment faire ?

Avec la célèbre construction **if-then-else** ! (en Python **if-elif-else**) :

```
AGE = input("Votre âge ? ")    # AGE a type str
AGE = int(AGE)                 # AGE a type int
if (AGE <= 13):
        print("Alors vous jouez à Call of Duty")
else:
        print("Alors vous jouez à Pokemon")
```

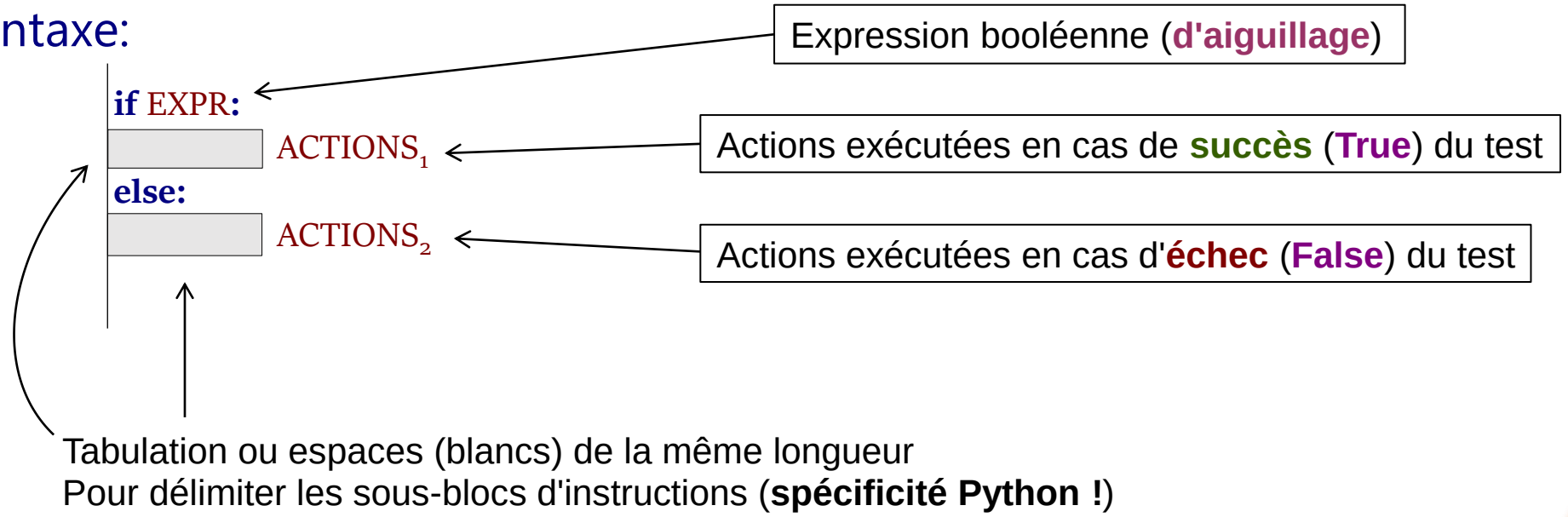


# Notion n°4, la « conditionnelle »

- Il est possible de planifier des actions **conditionnelles** en utilisant un test, qui est une **expression** dont le résultat est **booléen** (vrai ou faux) :

  - Si le test a **succès** (résultat **True**), le robot exécutera certaines actions
  - Si le test **échoue** (résultat **False**), le robot exécutera d'autres actions

Syntaxe:



# Un robot pour tout âge

- Syntaxe:

```
if EXPR:
        ACTIONS1
else:
        ACTIONS2
```

- Par rapport à notre exemple:

```
AGE = int(input("Votre âge ?"))
if (AGE <= 13):
        print("Alors vous jouez à Call of Duty")
else:
        print("Alors vous jouez à Pokemon")
```

# Un robot pour tout âge

- Syntaxe:

```
if EXPR:
        ACTIONS1
else:
        ACTIONS2
```

- Par rapport à notre exemple:

**Remarque** :
on compose les deux fonctions **input** et **int**
pour avoir directement un entier dans AGE

```
AGE = int(input("Votre âge ?"))
if (AGE <= 13):
        print("Alors vous jouez à Call of Duty")
else:
        print("Alors vous jouez à Pokemon")
```

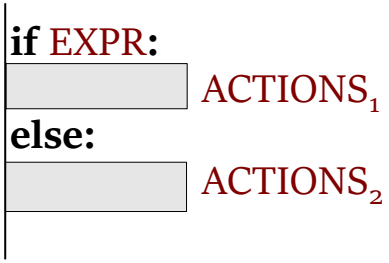
EXPR

$ACTIONS_1$

$ACTIONS_2$

# L’indentation en Python

- **En informatique** : l'indentation consiste en l'ajout de tabulations ou d'espaces (blancs) dans un fichier, pour une meilleure lecture et compréhension du code. C’est une fonction **aesthétique**.




- **En Python** : l'indentation fait partie de la **syntaxe** du langage pour délimiter les **blocs** de code. Par exemple, pour délimiter le bloc « then » (alors) et le bloc « else » (sinon). C’est une fonction **syntaxique**.

# Un robot pour tout âge
# Remarque du programmeur

```
AGE = int(input("Votre âge ?"))
if (AGE <= 13):
        print("Alors vous jouez à Call of Duty")
else:
        print("Alors vous jouez à Pokemon")
```

Non seulement on peut stocker (dans une variable)
une information **qu'on ne connaît pas**,
mais on peut aussi **traiter** cette information
en adaptant le comportement du robot aux différentes possibilités
par la construction **conditionnelle**

# Un robot pour footballeurs professionnels (v0)

```
#!/usr/bin/python3
# coding: utf-8
# Version 0 : basée sur l'âge

PRENOM = input("Votre prénom ?")
NOM = input("Votre nom ?")
AGE = int(input("Votre âge ?"))
if (AGE > 35):
        print("Aucune chance de gagner le Ballon d'Or !")
else:
        print("Vous avez de fortes chances de gagner le Ballon d'Or ! ")
```

# Un robot pour footballeurs professionnels (v1)

```
#!/usr/bin/python3
# coding: utf-8
# Version 1 : basée sur l'âge et sur l'équipe

PRENOM = input("Votre prénom ?")
NOM = input("Votre nom ?")
AGE = int(input("Votre âge ?"))
if (AGE > 35):
        print("Cher",PRENOM,NOM,"désolé de vous dire que")
        print("vous n'avez aucune chance de gagner le Ballon d'Or !")
else:
        CLUB = input("Quel est votre club ?")
        if (CLUB == "Barcelone" or CLUB == "PSG" or CLUB == "Juventus"):
                print("Vous avez de fortes chances de gagner le Ballon d'Or ! ")
        else:
                print("Ce n'est pas de notre faute si vous avez les pieds carrés !")
```

# Adresse des images utilisées

- Boite fermée https://openclipart.org/detail/15872/closed-box-by-mcol
- Robot sympa https://openclipart.org/detail/170101/cartoon-robot-by-sirrob01
- Robot chenille https://openclipart.org/detail/168755/cartoon-robot-by-qubodup
- Laptop https://openclipart.org/detail/24817/-by--24817
- Développeur https://openclipart.org/detail/37129/personnage_ordinateur-by-antoine
- Utilisateur https://openclipart.org/detail/37135/personnage_ordinateur-by-antoine-37135
- Pizza box https://openclipart.org/detail/171767/pizza-haunt-by-jakoriginal-171767
- Pizza https://openclipart.org/detail/189439/pepperoni-pizza-by-toons4biz-189439