Bilan

- Jusqu'à maintenant, nous avons appris les notions de programmation suivantes :
  - registres, affectation (stocker une valeur dans un registre) et comparaison de registres
  - opérations arithmétiques
  - saisir un nombre, afficher une chaine de caractères
  - boucles **for** avec compteurs et boucles **while** avec condition.
- Il ne reste plus qu'une seule structure pour être complets : la structure de choix (**si** *condition* **alors** … **sinon** …)

Adieu les robots...

- Avant de présenter la structure de choix **si ... alors ... sinon**, il est temps de nous séparer de nos deux robots qui nous ont permis de comprendre l'intérêt d'écrire des algorithmes et de les détailler jusqu'au niveau des compétences du processeur : ce n'est qu'ensuite que nous avons pu coder ces algorithmes dans les langages des robots pour produire des programmes.

- Désormais, notre seul processeur sera le langage Python 3. En fait, le fichier **robot.py** que vous avez déjà utilisé était écrit en Python 3, tout comme les affectations, les opérateurs et les boucles **for** et **while** que vous avez déjà utilisés : on peut donc considérer que Python 3 est un processeur de plus bas niveau que le processeur de Peter.

Python 3 : présentation

- Le langage Python a été conçu par Guido Van Rossum, d'abord dans un but pédagogique : pour apprendre la programmation.

- Sa simplicité et sa puissance ont fait qu'il a été rapidement adopté dans quasiment toutes sortes de projets. C'est actuellement l'un des langages les plus utilisés, aussi bien pour l'enseignement que pour la création de logiciels.

- Il y a deux versions de Python : l'ancienne (Python 2) et la nouvelle (Python 3) qui ne sont, malheureusement, pas entièrement compatibles.

- Le nom Python a été choisi par GVR en hommage aux Monty Python, une troupe d'humoristes anglais.

Jouons un peu…

- Tout le monde connaît le jeu du « plus ou moins » :
  - le joueur A choisit un nombre compris entre 1 et une limite donnée (100, par exemple). Supposons qu'il choisisse 75.
  - le joueur B doit proposer un nombre (42, par exemple)
  - A répond à B en lui indiquant si son nombre est trop petit (« c'est plus ») ou trop grand (« c'est moins »). Dans notre exemple, il répondrait donc « c'est plus ».
  - Le jeu s'arrête si B a épuisé tous ses essais sans trouver (il a donc perdu) ou s'il a deviné le nombre (auquel cas, il a gagné).

Jouons un peu…

- Voici un exemple de jeu où il fallait trouver 31 et où l'on n'avait droit qu'à 5 essais :

```
Entrez un nombre compris entre 1 et 100 : 42
C'est moins...
Entrez un nombre compris entre 1 et 100 : 33
C'est moins...
Entrez un nombre compris entre 1 et 100 : 12
C'est plus...
Entrez un nombre compris entre 1 et 100 : 25
C'est plus...
Entrez un nombre compris entre 1 et 100 : 27
C'est plus...
Il fallait trouver 31
```

Jouons un peu…

- Voici un exemple de jeu où il fallait trouver 90 et où l'on n'avait droit qu'à 5 essais :

```
Entrez un nombre compris entre 1 et 100 : 50
C'est plus...
Entrez un nombre compris entre 1 et 100 : 75
C'est plus...
Entrez un nombre compris entre 1 et 100 : 95
C'est moins...
Entrez un nombre compris entre 1 et 100 : 90
Bravo, vous avez trouve en 4 coups
```

Jouons un peu

- Dans ces exemples, c'est le programme que nous allons écrire qui joue le rôle du joueur A.

- Pour l'instant, intéressons-nous simplement aux réponses « C'est plus » ou « C'est moins » : le programme doit produire la première si le nombre proposé par le joueur est plus petit que le nombre à deviner. Il doit produire la seconde si le nombre est plus grand.

- On voit donc qu'il y a un « aiguillage » du programme qui dépend d'une condition : *le nombre est-il plus grand ou plus petit que le nombre secret ?*

Si … alors … sinon …

- En français, on règle ce genre de choix par une phrase comme « ***Si*** *il a plus de 10 de moyenne,* ***alors*** *il a obtenu son UE.* ***Sinon****, il repasse la seconde session »*.

- En algorithmique, c'est exactement la même chose… On écrit : **SI** *condition* **ALORS** *suite1* **SINON** *suite2*. Où *suite1* et *suite2* sont les suites d'instructions qui seront réalisées dans chacun des cas et *condition* est vraie ou fausse, comme dans un tant que.

- En programmation, la syntaxe peut varier en fonction des langages mais elle est toujours plus ou moins de la forme **if** *condition* … **else** …

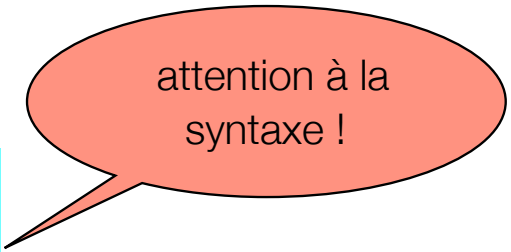
Si … alors … sinon …

- Pour notre jeu, si l'on suppose que le nombre à deviner est stocké dans le registre **secret** et que le nombre que l'utilisateur a entré est stocké dans le registre **nombre**, cela donnerait, en algorithmique :

```
SI nombre > secret ALORS
    afficher("C'est moins")
SINON
    afficher("C'est plus")
FSI
```

- Et, en Python :

```
if nombre > secret:
   afficher_ligne("C'est moins")
else:
   afficher_ligne("C'est plus")
```



Sinon si…

- En réalité, il n'y a pas que deux choix possibles, il y en a 3 :

```
SI nombre > secret ALORS
    afficher("C'est moins")
SINON SI nombre < secret ALORS
    afficher("C'est plus")
SINON
    afficher("Bravo, vous avez trouvé")
FSI
```

- Ce qui se traduit, en Python, par :

```
if nombre > secret:
    afficher_ligne("C'est moins")
elif nombre < secret:
    afficher_ligne("C'est plus")
else:
    afficher_ligne("Bravo, vous avez trouvé")
```

Un jeu complet

- Récapitulons :
  - On suppose que **secret** a été initialisé par un nombre pris au hasard entre 1 et 100.
  - L'utilisateur saisit un **nombre** et le programme affiche l'une des trois phrases *jusqu'à ce que l'utilisateur ait trouvé ou qu'il ait épuisé son* ***nombre d'essais*** (on suppose qu'il est fixé à 5).
- Réfléchissez à partir des éléments d'algorithmique qu'on a déjà vus et du bout d'algorithme qu'on a déjà écrit : comment écrire l'algorithme du jeu complet ?

Un jeu complet

- Réfléchissez à partir des éléments d'algorithmique qu'on a déjà vus et du bout d'algorithme qu'on a déjà écrit : comment écrire l'algorithme du jeu complet ?

```
SI nombre > secret ALORS
    afficher("C'est moins")
SINON SI nombre < secret ALORS
    afficher("C'est plus")
SINON
    afficher("Bravo, vous avez trouvé")
FSI
```

- On a besoin de 3 registres numériques : **secret**, **nombre**, **nb_essais** et d'un registre indiquant si, oui ou non, on a trouvé…

Un jeu…

- Étudiez ce code :

```
from outils import *                    # pour aleatoire(), afficher_ligne(), lire_lombre()…

secret = aleatoire(1, 101)              # initialise secret avec un nbre au hasard
nb_essais = 0                           # initialise le nombre d'essais déjà effectués
trouve = False                          # on n'a pas encore trouvé…

while nb_essais < 5 and not trouve:
  afficher("Entrez une valeur comprise entre 1 et 100 : ")
  nombre = lire_nombre()
  nb_essais = nb_essais + 1             # on a joué un essai de plus
  if nombre > secret:
    afficher_ligne("C'est moins…")
  elif nombre < secret:
    afficher_ligne("C'est plus…")
  else:
    afficher_ligne("Bravo, vous avez trouvé en", nb_essais, "coups")
    trouve = True

if not trouve:                          # c'est parce qu'on a épuisé tous nos essais
  afficher_ligne("Désolé, il fallait trouver", secret)
```

Remarque sur la structure d'un programme Python

la structure d'un programme est définie par l'indentation des lignes

```
from outils import *                    # pour aleatoire(), afficher_ligne(), lire_lombre()…

secret = aleatoire(1, 101)              # initialise secret avec un nbre au hasard
nb_essais = 0                           # initialise le nombre d'essais déjà effectués
trouve = False                          # on n'a pas encore trouvé…

while nb_essais < 5 and not trouve:
  afficher("Entrez une valeur comprise entre 1 et 100 : ")
  nombre = lire_nombre()
  nb_essais = nb_essais + 1             # on a joué un essai de plus
  if nombre > secret:
    afficher_ligne("C'est moins…")
  elif nombre < secret:
    afficher_ligne("C'est plus…")
  else:
    afficher_ligne("Bravo, vous avez trouvé en", nb_essais, "coups")
    trouve = True

if not trouve:                          # c'est parce qu'on a épuisé tous nos essais
  afficher_ligne("Désolé, il fallait trouver", secret)
```

Un jeu…

- Utilisez votre éditeur favori pour saisir ce programme.
- Lancez-le et jouez… (en vérifiant qu'il fonctionne correctement !!!)

Améliorations possibles

- Demander à l'utilisateur un niveau de difficulté (0, 1 ou 2) qui correspondrait à un nombre d'essai maximum => utilisation d'un registre **niveau** et d'un registre **nb_essais_max**.

- Permettre de rejouer ou non => registre **encore** et boucle **while**

Bilan

- Cet exemple a fait apparaître de nouvelles valeurs (**True**, **False**) et de nouveaux opérateurs (**and**, **not**)…

- Si on résume, on utilise donc 3 types de données dans ce jeu : des *nombres*, des *chaînes de caractères* et ce nouveau type, appelé **booléen**.

- Nous allons voir qu'en programmation, tous les objets que l'on manipule (registres/variables) sont "typés".

- C'est le type d'une donnée qui définira les valeurs qu'elle pourra contenir et les opérations que l'on pourra lui appliquer.