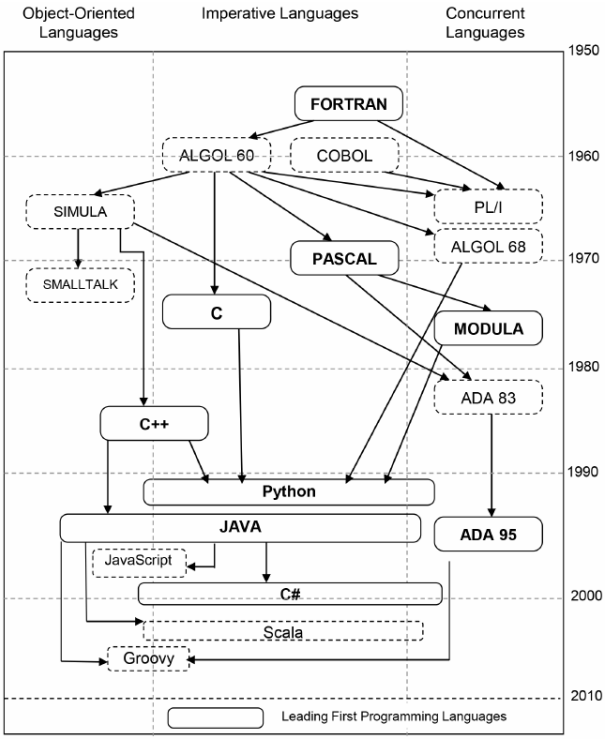
Lycée Les 3 Sources **Bourg-Lès-Valence**
NSI 1re – Thème 6 Année **2024-25**

Diversité des langages de programmation

Il n'y a pas que Python dans la vie ?

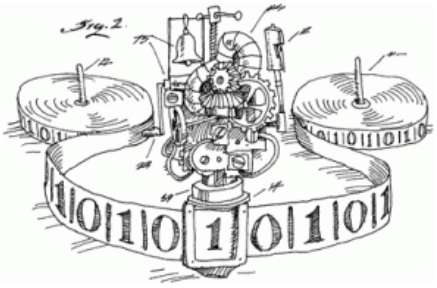
Même si Python est un langage très versatile permettant de faire énormément de choses, ce n'est pas le seul langage de programmation. Wikipedia en liste même des centaines.
Chacun de ces langages répond à des besoins différents (généraliste, résolution de contraintes, descriptif…), des paradigmes différents (orienté objet, fonctionnel, impératif, concurrents…), des fonctionnements différents (interprété, compilé…)…Certains langages sont fait spécifiquement pour résoudre certains types de problèmes. D'autres peuvent au contraire résoudre n'importe quel type de problème. Certains possèdent de nombreuses bibliothèques, d'autres n'en ont pas ou très peu.
On pourrait se demander quel est le langage le plus puissant, mais en fait il n'y en a pas.



La machine de Turing

En 1937, Alan Turing (1912–1954), un mathématicien anglais de génie a imaginé une machine universelle Machine de Turing. Cette machine théorique est considérée comme le fondement de l'ordinateur à mémoire, qui sera concrétisé par von Neumann.

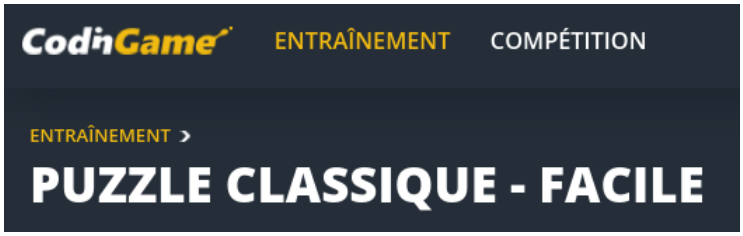
Cette machine est extrêmement simpliste, puisqu'uniquement composée d'un ruban infini sur lequel sont écrits des 0 et des 1 et d'une tête qui se déplace sur ce ruban en lisant et écrivant les valeurs. Un programme est juste une suite d'instructions finies du genre : "Si la tête est dans l'état 2 et lit un 1, alors elle écrit 0 et va à droite en passant à l'état 3". Lorsqu'il n'y a pas d'instruction à exécuter, le programme s'arrête.



Ce modèle a l'air très limité, mais pourtant Turing et Alonzo Church (1903–1995), un mathématicien américain, ont démontré que tout ce qui pouvait être décrit par un algorithme pouvait être exécuté sur la machine de Turing. Lorsqu'un langage de programmation est capable de faire autant de choses que la machine de Turing, on dit qu'il est **Turing-complet**. La plupart des langages de programmation sont Turing-complet. Ce qui veut dure que tout ce que peut faire un langage de programmation particulier peut également être fait dans n'importe quel autre langage. Il n'y a donc pas de langage meilleur que les autres. Il y a juste des langages plus adaptés à certaines situations qu'à d'autres.

La tour de Babel

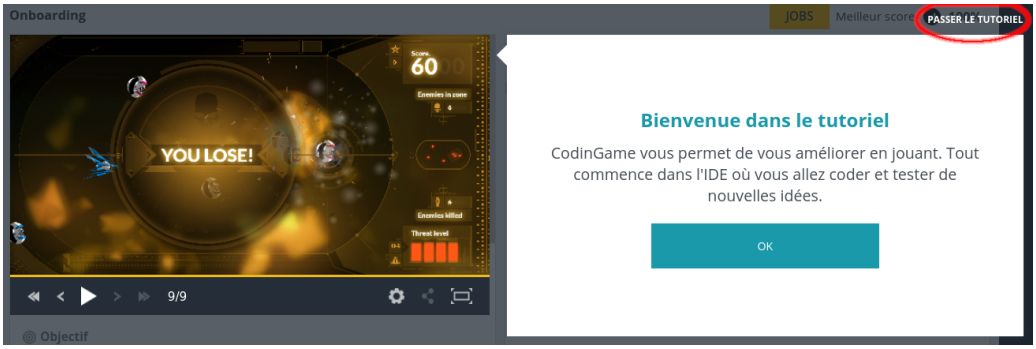
On dit parfois que le plus dur c'est d'apprendre un premier langage de programmation. Ensuite, en apprendre d'autres est bien plus simple. C'est globalement vrai. La grande difficulté pour apprendre à programmer, c'est l'algorithmique. Ensuite, à peu près tous les langages ont des structures assez similaires. Pour étudier cela, nous allons écrire le même programme dans 15 langages différents. Nous allons utiliser le site codingame.com. Si vous n'avez pas encore de compte, vous devrez vous en créer un. Il faut utiliser une adresse mail. Vous pouvez inventer une fausse adresse, comme `compte.bidon128@gmail.com` puisqu'il n'y a pas de mail de vérification d'envoyé. Par contre, il faudra vous souvenir de cette adresse pour vous re-connecter plus tard.
Nous allons utiliser le tutoriel du site car il donne une solution au problème dans tous les langages disponibles. Si vous n'êtes pas directement envoyé sur ce tutoriel, vous pouvez le trouver dans les exercices faciles de la partie entraînement. Il est probablement tout en bas de la page si vous l'avez résolu.



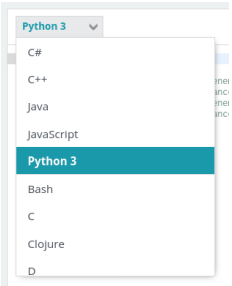
Chaque niveau de Codingame se présente de la manière suivante :

Il faut lire la présentation du problème pour savoir ce qu'il faut faire, à partir de quelles entrées et surtout les sorties à renvoyer. Une fois la solution tapée dans la zone prévue à cet effet, il faut la tester avec les jeux de tests proposés. Votre solution doit en valider le plus possible. Pour l'entraînement, il n'y a qu'un seul test à passer.
Une fois que votre solution vous convient, il faut appyer sur le bouton "SOUMETTRE" pour la valider.
Lors de votre première tentative le site vous propose un tutoriel qui vous guide sur la marche à suivre. Il est conseillé de le faire pour bien découvrir les différentes zones.

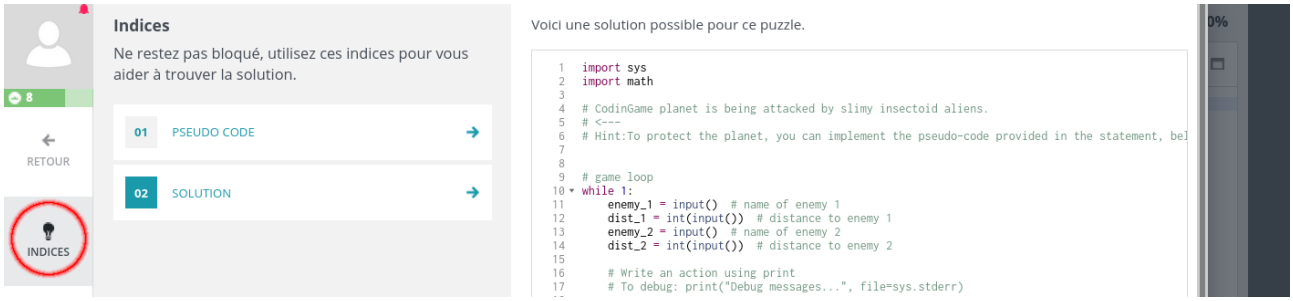
Par contre, vous n'avez pas à suivre le tutoriel à chaque fois. Il suffit de cliquer en haut à droite une fois que vous l'avez fait au moins une fois.



Une fois dans le niveau, il faut choisir son langage. Vous pouvez prendre celui que vous voulez.

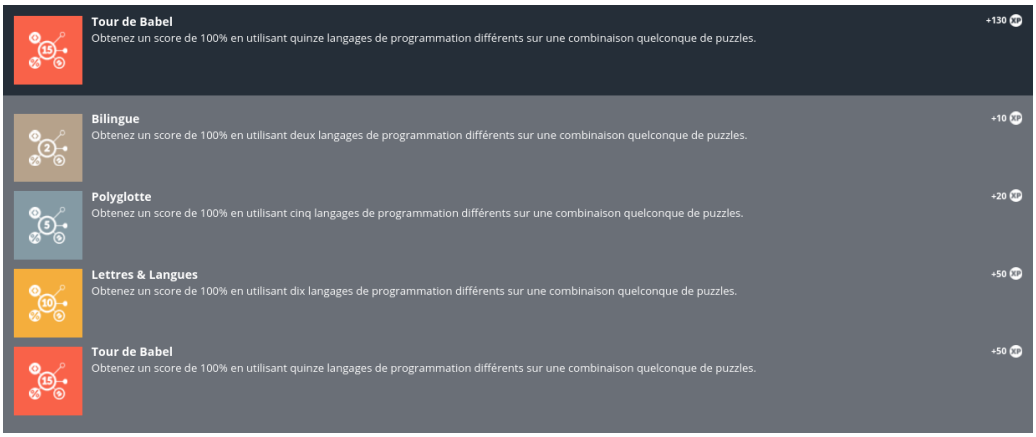


Il faut ensuite aller sur les indices et prendre le second pour avoir la solution. Copiez cette solution et collez-là à la place du programme de base.



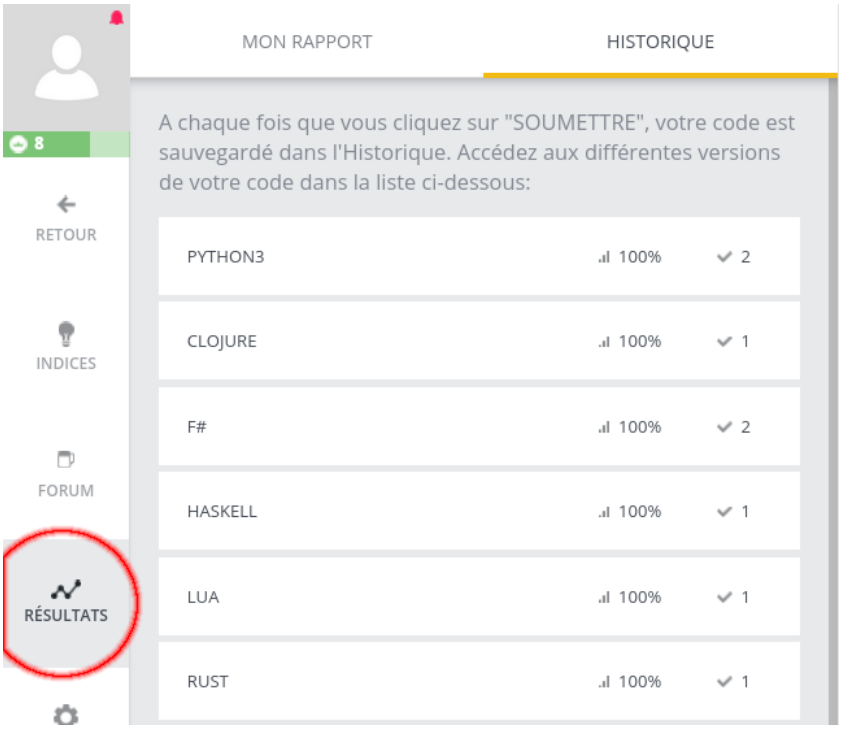
Il faut ensuite appuyer sur le bouton soumettre (en bas à droite) pour valider votre réponse.
Vous devez répéter l'opération avec 15 langages différents. Vous pouvez prendre ceux que vous voulez. Ne vous limitez pas aux 15 premiers de la liste.

Après avoir soumis votre réponse, il est possible de rester pour modifier votre code et ainsi changer directement de langage et de recommencer. Par contre au bout d'un certain nombre de soumissions, vous risquez de devoir remplir de Captcha. Si c'est le cas, vous pouvez faire une petite pause et revenir après.
Pour suivre votre progression, vous pouvez aller voir dans votre profil, l'évolution du succès "Tour de Babel".



Ce que vous devez faire

Une fois que vous avez utilisez 15 langages, vous allez pouvoir commencer à comparer les langages. Pour retrouver toutes vos soumissions, vous pouvez aller dans l'historique :



Exercice : Créez un fichier `diversite-langages-NOM.txt` que vous éditerez avec Notepad++ ou un autre éditeur de texte. Évitez juste Word ou LibreOffice, ou alors choisissez une police à taille fixe pour les bouts de code.

1) Pour chaque langage utilisé :
   a) Indiquez le nom du langage.
   b) Collez le code de la solution.
   c) Identifiez pour les langages, autres que Python, la syntaxe utilisée pour séparer les instructions, les bloces d'instructions, les affectations, les boucles "tant que", les tests "si alors sinon", les entrées (inputs), les sorties (affichages) et les commentaires. Si vous n'arrivez pas à les identifier, dites juste que vous n'avez pas trouvé pour ce langage.
   La page suivante vous montre un exemple pour la solution en Python.
2) À votre avis (et c'est totalement subjectif) :
   a) Quel est le langage le plus simple ?
   b) Quel est le langage le plus compliqué ?
   c) Quel est le langage le plus original ?
   d) Quel est le langage le plus concis ?
   e) Quel est le langage avec la syntaxe la plus pénible ?

Attention, certains programmes ne sont pas totalement représentatifs de la simplicité, ou la complexité, des langages concernés. Les solutions sont générées en partie automatiquement et ne correspondent pas forcément à la syntaxe la plus simple ou la plus élégante. Il y a aussi le système utilisé pour la gestion des entrées et des sorties qui fait que les affichages sont parfois plus compliqués que de simples "`print`", mais ce qui ne veut pas dire qu'ils n'existent pas dans tous les langages.

Exemple pour Python :

```
Langage Python :

Solution :
while 1:
    enemy_1 = input()   # name of enemy 1
    dist_1 = int(input())   # distance to enemy 1
    enemy_2 = input()   # name of enemy 2
    dist_2 = int(input())   # distance to enemy 2
    if dist_1 < dist_2:
        print(enemy_1)
    else:
        print(enemy_2)

Instrutions séparées par un saut à la ligne :
INSTR1
INSTR2

Les blocs d'instructions sont séparés par des changements d'indentation.

Affectations : VAL = EXPR

Boucles tant que :
while TEST:
    INSTR1
    INSTR2

Tests :
if TEST:
    INSTR_SI
else:
    INSTR_SINON

Entrées : VAL = input()
Sorties : print(EXPR)

Commentaires : # un commentaire sur une ligne
"""
Un commentaire sur plusieurs lignes
"""
```