On vous conseille de commencer par créer un dossier TP3, dans lequel vous placerez les programmes que vous écrirez au cours de ce TP.

Exercice III.1 *(Recherche)*
Imaginons le jeu suivant entre deux personnes Alice et Bob :

- Alice commence par choisir un nombre n compris entre 1 et 100, que Bob va chercher à deviner.
- Ensuite, Bob propose un premier nombre. Alice lui indique s'il a deviné le bon nombre. Si c'est le cas, le jeu s'arrête. Si ce n'est pas le cas, alors Alice lui indique si le nombre proposé est strictement inférieur, ou strictement supérieur, au nombre n à deviner. Bob peut alors proposer une nouvelle valeur. Le jeu se poursuit tant que Bob n'a pas trouvé n.

Ecrivez un programme Python, intitulé `recherche.py`, qui permet de jouer à ce jeu. La valeur de n sera demandée à l'utilisateur (à ce moment-là, Alice saisit le nombre, et Bob n'a pas le droit de regarder...). Ensuite, c'est Bob qui saisit au clavier le nombre qu'il veut proposer.

Votre programme testera également que le nombre choisi par Alice est bien compris entre 1 et 100.

Pour finir, améliorez votre programme en ajoutant une variable supplémentaire, qui servira à compter le nombre de tentatives qu'il a fallu à Bob pour trouver le nombre n.

Faites quelques tentatives avec un autre étudiant, vous devez observer qu'on parvient à trouver le nombre en 7 coups environ. Savez-vous pourquoi ?

Exercice III.2 *(Recherche vs Computer)*
Dans un programme, recopiez le code ci-contre. La fonction `randrange` permet de générer aléatoirement des entiers dans un certain intervalle (syntaxe similaire à celle de la fonction `range`).
Utilisez ce code pour modifier le jeu de recherche, en jouant à présent à deviner un nombre choisi aléatoirement par l'ordinateur.

```
from random import *

for i in range(10):
   print(randrange(20))
```

Exercice III.3 *(Doubles boucles)*
Dans un programme, recopiez le code ci-contre. L'ajout de la virgule à la fin de l'appel à la fonction print permet d'éviter le saut de ligne. Le saut de ligne est ensuite forcé avec une instruction print sans argument.
Modifiez ce code pour obtenir l'affichage ci-dessous à gauche.
De même avec l'affichage ci-dessous à droite.

```
for i in range(10):
    for j in range(10):
        print(j," ",end='')
    print()
```

```
0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 0
2 3 4 5 6 7 8 9 0 1
3 4 5 6 7 8 9 0 1 2
4 5 6 7 8 9 0 1 2 3
5 6 7 8 9 0 1 2 3 4
6 7 8 9 0 1 2 3 4 5
7 8 9 0 1 2 3 4 5 6
8 9 0 1 2 3 4 5 6 7
9 0 1 2 3 4 5 6 7 8
```

```
0 1 2 3 4 5 6 7 8 9
0 0 1 2 3 4 5 6 7 8
0 0 0 1 2 3 4 5 6 7
0 0 0 0 1 2 3 4 5 6
0 0 0 0 0 1 2 3 4 5
0 0 0 0 0 0 1 2 3 4
0 0 0 0 0 0 0 1 2 3
0 0 0 0 0 0 0 0 1 2
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
```

Exercice III.4 *(Calcul du PGCD)*

1. Ecrire un *programme* Python qui demande à l'utilisateur de saisir deux entiers naturels strictement positifs n et m et affiche tous leurs diviseurs communs, du plus petit au plus grand.
2. Dans un nouveau programme, écrire une *fonction* qui, étant donnés deux entiers naturels strictement positifs n et m, retourne leur plus grand diviseur commun (pgcd), en faisant une recherche "ascendante". En effet, vous pouvez remarquer que leur pgcd est le dernier diviseur affiché dans la question précédente.

Votre fonction parcourra donc les entiers naturels par ordre croissant à partir de 1. Cette fonction sera nommée `pgcd1`.

3. Ecrire une nouvelle *fonction* qui permet de calculer le pgcd de deux nombre n et m en faisant une recherche "descendante" à partir de la valeur $\min(n, m)$ (la fonction `min` existe en Python, vous pouvez l'utiliser). Cette fonction sera nommée `pgcd2`.
4. Vérifiez sur quelques exemples que vos deux fonctions retournent bien la même valeur.
5. Selon vous laquelle des deux fonctions est la plus rapide ? Pour le démontrer formellement, exprimer mathématiquement, en fonction de la valeur du pgcd de m et n, le nombre d'entiers naturels énumérés par chacune de vos deux fonctions. Dans quel cas précis les deux fonctions considèrent-elles le même nombre d'entiers naturels ?
6. On souhaite à présent vérifier cette propriété de façon expérimentale. Modifiez vos deux fonctions pour qu'elles retournent non plus la valeur du pgcd, mais le nombre d'entiers naturels énumérés (utilisez une variable supplémentaire servant de compteur). Vous les nommerez respectivement `pgcd1compte` et `pgcd2compte`.

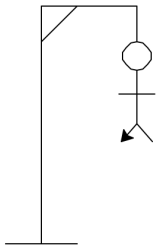
   En utilisant deux boucles `for` imbriquées, la première permettant de faire varier la valeur de m entre 1 et 50, la seconde faisant varier n entre 1 et 50, affichez le nombre de fois où :
   — `pgcd1compte(m,n)` est égal à `pgcd2compte(m,n)`,
   — `pgcd1compte(m,n)` est strictement supérieur à `pgcd2compte(m,n)`,
   — `pgcd1compte(m,n)` est strictement inférieur à `pgcd2compte(m,n)`.
   La somme de ces trois nombres doit être égale à 50*50=2500. Commentez les proportions.

Exercice III.5 *(Calendrier perpétuel)*
A partir de la fonction testant si une année est bissextile, écrire un programme permettant de réaliser un calendrier perpétuel. Le programme demande à l'utilisateur de saisir trois entiers correspondant à une date (jour, mois, année) postérieure au 1er janvier 2000. Le programme doit ensuite indiquer de quel jour de la semaine il s'agit (lundi, mardi...).

On indique que le 1er janvier 2000 était un samedi, qu'une année bissextile compte 366 jours, tandis qu'une année non bissextile compte 365 jours. Quel jour est le 23 septembre 2016 ?

Exercice III.6 *(Pour tuer le temps)*

Vous connaissez sans doute le jeu du pendu. Un premier joueur choisit un mot, et le second joueur doit deviner ce mot. Pour cela, le second joueur propose des lettres, une par une, et à chaque lettre proposée, le premier joueur répond de la façon suivante :
— si la lettre appartient au mot, alors il doit révéler cette lettre dans le mot, à toutes les positions où elle apparaît.
— si la lettre n'appartient pas au mot, alors il ajoute un élément au dessin qui représente le pendu.



Proposez un programme réalisant ce jeu. Le programme demandera aux deux joueurs leurs choix et se chargera de l'affichage du pendu, du mot (avec des traits pour les lettres inconnues), et d'indiquer au cours de la partie si le deuxième joueur a perdu (il est pendu) ou gagné (il a deviné le mot). On pourra considérer que le dessin du pendu se réalise en 10 étapes, et il pourra être réalisé à l'aide du module turtle.