

## Algorithme n°1 : l'indice IMC

### Écrire un programme sous python

qui demande la taille (en mètres) et la masse (en kg) d'un individu et lui renvoie sa masse IMC avec un petit commentaire :

Si l'indice IMC est inférieur à 25, le commentaire pourra être : « vous n'êtes pas en surpoids »

sinon le commentaire pourra être : « vous êtes en surpoids ».

Cet algorithme utilisera 3 variables :

- masse (à transformer en "float"),
- taille (à transformer en "float"),
- $IMC = \frac{masse}{taille \times taille}$

### Amélioration n°1 :

Initialiser la masse à -1 et la taille à -1 et intégrer les questions sur la masse et la taille dans une boucle non bornée qui répète la question tant que la réponse est incohérente.

### Amélioration n°2

si l'indice IMC est inférieur à 18,5, l'individu est considéré comme « maigre ». Prendre en compte cette information dans les messages.

## Algorithme n°2 : calculer une moyenne

### Écrire un programme sous python

qui, à chaque fois qu'on saisit en entrée 5 notes sur 20, affiche en sortie la moyenne de ces cinq notes.

Cet algorithme utilisera au moins les deux variables suivantes :

- la variable note
- la variable moy

### Amélioration n°1 :

il est possible de n'utiliser que 2 variables en utilisant au bon endroit l'instruction :

```
note=note+float(input("Entrer une note : "))
```

### Amélioration n°2 :

on souhaite maintenant rentrer autant de notes que l'on veut. Comment faire ?

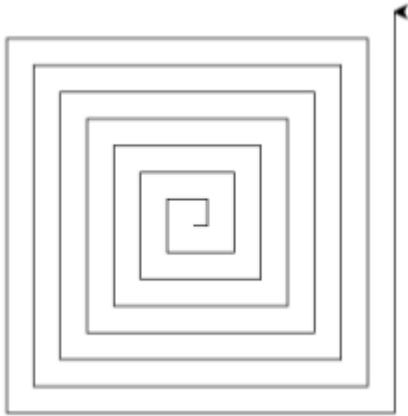
### Amélioration n°3 :

on souhaite que l'utilisateur rentre les notes à partir du message suivant : "Taper une note ou -1 pour terminer"

### Algorithme n°3 : la spirale.

#### Écrire un programme sous python

permettant de créer la spirale ci-dessous.



Elle est composée de 30 segments.

Le premier segment mesure 10 pixels.

A chaque fois que l'on tourne à gauche, le côté du segment augmente de 10 pixels.

Remarque : la première ligne de votre programme intégrera le module « turtle » permettant le dessin :

```
from turtle import *
```

Les instructions utiles :

`forward(longueur)` : pour avancer de la longueur souhaitée en pixels.

`left(90)` : pour tourner à angle droit sur la gauche.

### Algorithme n°4 : Tirage de dé

#### Exercice :

L'algorithme ci-dessous permet de lancer un dé à 6 faces 50 fois et d'afficher les résultats :

```
from random import *  
for lancer in range (50) :  
    print("lancer n°",lancer,":",randint(1,6))
```

Modifier cet algorithme (en gardant le même dé et le même nombre de lancers) pour qu'il affiche le nombre de fois où le 1 est sorti.

#### Amélioration :

Faire varier le nombre n de tirages et vérifier si la fréquence de sortie du 1 reste dans l'intervalle de

fluctuation  $\left[ p - \frac{1}{\sqrt{(n)}}; p + \frac{1}{\sqrt{(n)}} \right]$