Exercice 6.

Écrire en Python la définition de la classe IMC (indice de masse corporelle) et son constructeur avec quatre attributs pour caractériser une personne :

a) nom (str),
b) age (int),
c) taille (float) en mètres,
d) poids (float) en kg

Écrire la méthode de classe **imc()** qui traite l'IMC de la personne. L'IMC est calculé à partir du poids de la personne divisé par sa taille puissance 2 :

$IMC = masse/taille^2$

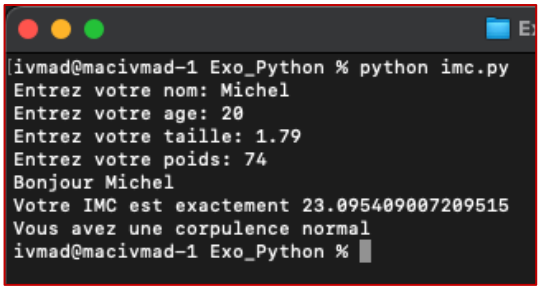
Interprétation de l'IMC

IMC (kg·m^{-2})	Interprétation
moins de 16,5	dénutrition ou famine
16,5 à 18,5	maigreur
18,5 à 25	corpulence normale
25 à 30	surpoids
30 à 35	obésité modérée
35 à 40	obésité sévère
plus de 40	obésité morbide ou massive

Écrire la méthode de classe **interpretation()** qui affiche le verdict selon le calcul obtenu de la méthode **imc()** en accord du tableau d'interprétation (voir la figure jointe).

Les données pour la personne sont saisies au clavier. Le résultat du calcul est affiché avec le contenu suivant :

```
Bonjour (son nom)
Votre IMC est exactement : (son imc)
Vous avez (une corpulence normale ou bien Vous êtes en surpoids, ou bien Vous êtes etc.…)
```



Effectuer des tests avec un jeux de données au minimum trois.

Exercice 7.

Daniel Gabriel Fahrenheit est l'inventeur des thermomètres en graduation Fahrenheit. Au début, ses thermomètres sont à l'alcool (1709), mais il remplace rapidement l'alcool par du mercure (1715), permettant à ses outils de mesure de fournir des données comparables. En 1742, un autre scientifique propose une nouvelle graduation au thermomètre. Anders Celsius crée une échelle thermométrique où ses degrés représentent :

`°C=(°F-32)/1.8` on peut obtenir la correspondance par `°F=(°C*9/5)+32`

Enregistrer une classe Python "Temperature" dans le fichier **TempCelFar.py** avec les attributs **celsius** et **fahrenheit** de type numérique et un attribut **temp** de type string permettant de déterminer le choix de la méthode de classe **cel-far** pour convertir des températures de Celsius en Fahrenheit ou le choix de la méthode de classe **far-cel** pour convertir des températures de Fahrenheit en Celsius. Mettre les attributs de la classe en mode privé. Utiliser les **accesseurs** et **mutateurs** pour accéder et manipuler les données de température. Exécuter l'application Python depuis le fichier **ApplTempFarCel.py** en demandant les données au clavier. Afficher le résultat accompagné de l'indication `°C` ou `°F` suivant le cas.

Écrire une application Python qui instancie la classe "Temperature". La classe est importée avec l'instruction :

```
from <nom de fichier> import <nom de la classe>
```

Effectuer des tests avec un jeux de données au minimum trois.

Exercice 8.

a) Créer une classe Python nommée **CompteBancaire** qui représente un compte bancaire, ayant pour attributs
   a. **numeroCompte** (type numérique),
   b. **nom** (nom du propriétaire du compte de type chaîne de caractères),
   c. **solde** (type numérique).

b) Créer un **constructeur** ayant comme paramètres : *`numeroCompte, nom, solde`*.

c) Créer une méthode **`Versement()`** qui gère les versements.

d) Créer une méthode **`Retrait()`** qui gère les retraits.

e) Créer une méthode **`afficher()`** permettant d'afficher le relevé du compte

f) Écrire la méthode principale dans un fichier à part qui effectue des versements et des retraits en boucle en précisant le code d'opération V/R ou F pour fin du programme.

```
Entrez le numéro de compte: 135798642
Entrez le nom du titulaire: Michel
Entrez le solde intial: 23000
Entrez opération (V ou R ou F pour fin): V
Versez €: 4500
Compte numéro :  135798642
Nom & Prénom :  Michel
 Solde  :  27500  DH
Sauf erreur ou omisssion !
Entrez opération (V ou R ou F pour fin): R
Retirez €: 5896
Compte numéro :  135798642
Nom & Prénom :  Michel
 Solde  :  21604  DH
Sauf erreur ou omisssion !
Entrez opération (V ou R ou F pour fin): F
Compte numéro :  135798642
Nom & Prénom :  Michel
 Solde  :  21604  DH
Sauf erreur ou omisssion !
ivmad@macivmad-1 Exo_Python %
```

Exercice 9.

a) Reprendre l'exercice précédent avec le **CompteBancaire**.

b) Sécuriser les attributs de la classe **CompteBancaire** en les rendant privés.

c) Ajouter les **accesseurs** (getters) et les **mutateurs** (setters) pour assurer du bon fonctionnement du compte bancaire.

d) Note sur le fonctionnement d'un compte bancaire :

   Dans son fonctionnement normal, un compte bancaire est censé être en positif. Il arrive toutefois qu'il passe en position débitrice, qu'il soit "à découvert". Les banques peuvent autoriser ce découvert, mais en contrepartie, elles facturent à leurs clients des intérêts débiteurs, appelés **agios**.

e) Créer une méthode **`Agios()`** permettant d'appliquer les agios à un pourcentage de 5 % du solde.

f) A chaque retrait vérifier la solvabilité du compte bancaire.

g) Utiliser les accesseurs (*`get_solde()`*)et les mutateurs (*`set_solde()`*) pour la gestion de la solvabilité du compte.

```
Entrez le numéro de compte: 135798642
Entrez le nom du titulaire: Jacques
Entrez le solde intial: 1500
Entrez opération (V ou R ou F pour fin): V
Versez €: 100
Compte numéro :  135798642
Nom :  Jacques
 Solde :  1600 €
Entrez opération (V ou R ou F pour fin): R
Retirez €: 1000
Compte numéro :  135798642
Nom :  Jacques
 Solde :  600 €
Entrez opération (V ou R ou F pour fin): R
Retirez €: 700
Solde insuffisant ! La règle AGIOS sera appliquée
Compte numéro :  135798642
Nom :  Jacques
 Solde :  -105.0 €
Entrez opération (V ou R ou F pour fin): F
Compte numéro :  135798642
Nom :  Jacques
 Solde :  -105.0 €
ivmad@macivmad-1 Exo_Python %
```

Exercice 10.

1. Définir une classe Python nommé "**Time**" avec les attributs de la classe pour mémoriser les **heures**, **minutes** et **secondes**.
2. Créer une méthode **affiche_heure(self)** qui serve à retourner en chaîne de caractères les attributs de l'objet de la classe Time.
3. Créer un objet **t** de type **Time** et tester la méthode sur cet objet.

```
k-IvMad/IvMad/Cours-IvMad/BUT_2_R3.08_POO/Exo_Python/time_2022.py
Présentation du temps: 1662881314.882584
Aujourd'hui Sun Sep 11 09:28:34 2022
resultat: time.struct_time(tm_year=2022, tm_mon=9, tm_mday=11, tm_hour=9, tm_min=28, tm_sec=34, tm_wday=6, tm_yday=254, tm_isdst=1)
année: 2022
heure: 9
minute: 28
seconde: 34
ivmad@macivmad-1 Exo_Python %
```

Exercice 11.

1. Écrire une classe Python pour le calcul des racines réelles d'un trinôme du second degré : `ax2 + bx + c`
2. Les attributs de la classe sont des paramètres réels du trinôme : `a`, `b` et `c`.
3. Une méthode `affiche(self)` est appelée si le trinôme n'a pas de racine réelle en affichant une phrase indiquant qu'il n'y a pas de racine réelle.
4. Écrire la méthode de classe pour la saisie au clavier des coefficients du trinôme sous forme de 3 nombres flottants.
5. Écrire la méthode qui retourne `delta` après avoir calculer la valeur discriminant du trinôme.
6. Écrire une méthode qui teste la valeur du discriminant. En fonction du signe du discriminant la méthode retourne une phrase donnant le nombre de racines réelles du trinôme, puis la valeur de ces racines réelles (ou de cette racine réelle s'il s'agit d'une racine double).
7. N'oubliez pas les commentaires.
8. Indication : pour utiliser la fonction `sqrt` du module `math`, il faut importer le module juste après l'en-tête du fichier, la ligne : `from math import sqrt` (mais on peut aussi utiliser **0.5 (puissance 1/2) ce qui évite l'importation la fonction `sqrt`).

9. Écrire la partite test de la classe dans un fichier à part.
10. Effectuer des tests avec un jeux de données couvrant les cas donnant à des solutions différentes.