Consignes. Avant toute chose, prenez conscience que les corrections sont effectuées à l'aide d'un automate, par conséquent toute violation des règles conduira inévitablement à la délivrance d'une note nulle.

- Comment livrer mon code ?
  Si mon nom de famille est **yunès** et que je souhaite rendre le travail pour l'exercice 12 du tp 3 alors je dois construire une archive au format ZIP de nom `yunes-tp3-ex12.zip`, le nom ayant été converti en minuscules, sans accent et sans espace (*ie* : **de Gaulle** doit être traduit en `degaulle`). Si vous avez un homonyme rajoutez des initiales, par exemple `yunesjb-tp1-ex1`. Respectez votre nommage pour tout le semestre.
- Que mettre dans l'archive ?
  Le code source (et uniquement le code source) correspondant à l'exercice.

De plus, suivez bien les instructions complémentaires des livrables.

1 Exercice 1 (input/output/arithmetic)

Écrire un programme python qui :

- saisit un nombre représentant une distance exprimée en mètres,
- saisit un nombre représentant un temps exprimé en secondes,
- affiche la vitesse résultante exprimée en km/h,
- affiche la vitesse résultante exprimée en km/h avec une précision de 2 chiffres après la virgule.

```
La vitesse est de 14.399999999999999 km/h
La vitesse est de 14.40 km/h
```

Livrable : l'archive contenant le programme `vitesse.py`.

2 Exercice 2 (function)

Écrire un programme python qui :

- définit une fonction permettant de calculer la valeur du polynôme $3x^2 + 5x - 10$
- affiche la valeur du polynôme au point 0
- affiche la valeur du polynôme en tous les points $-5 \leq x \leq 5$ (par pas de 0.5) sous la forme `P(valeur)=valeur`

```
-10
P(-5.0)=40.0
P(-4.5)=28.25
P(-4.0)=18.0
P(-3.5)=9.25
...
```

Livrable : l'archive contenant le programme `polynome.py`.

3 Exercice 3 (exceptions)

Écrire un programme qui :

1. définit une fonction permettant de tenter de calculer la valeur de $\frac{sin(x)}{x}$ pour toutes les valeurs $-3 \leq x \leq 3$ (par pas de 0.5) et lorsque ce n'est pas possible d'afficher le message `Pas possible avec` *valeur*

```
...
0.6649966577360363
0.8414709848078965
0.958851077208406
>>>>>>>>>>>Pas possible avec 0.0
0.958851077208406
...
```

Livrable : l'archive contenant le programme `sintest.py`.

Exercice 4 (list)

Écrire un programme Python qui :

1. construit et affiche la liste :
   `[9, 7, 3, 2, 7, 8, 3, 8, 4, 2, 7, 0, 5, 3, 2, 0, 9, 6, 0, 5, 6, 2, 2, 4, 5, 2, 6, 3, 5, 2]`
2. affiche le nombre d'éléments de la liste
3. affiche le nombre d'éléments égaux à 7
4. ajoute le nombre 9 en fin de liste puis affiche la liste
5. supprime le premier élément égal à 7 et affiche la liste
6. affiche le nombre d'éléments égaux à 7
7. renverse la liste et l'affiche
8. trie la liste et l'affiche
9. affiche la sous-liste des 5 derniers éléments
10. affiche la liste des 5 plus grand éléments de la liste (à la duplication près)

Livrable : l'archive contenant le programme `liste.py`.

Exercice 5 (comprehensive lists and sets)

Écrire un programme Python qui :

1. construit et affiche la liste :
   `[9, 7, 3, 2, 7, 8, 3, 8, 4, 2, 7, 0, 5, 3, 2, 0, 9, 6, 0, 5, 6, 2, 2, 4, 5, 2, 6, 3, 5, 2]`
2. construit la liste des éléments impairs et l'affiche
3. construit la liste des éléments pairs et l'affiche
4. construit l'ensemble des éléments pairs et strictement plus grands que 4 et l'affiche

Livrable : l'archive contenant le programme `comprehensive.py`.

Exercice 6 (Quelques petits packages...)

Écrire un programme Python qui (à l'aide des packages `random` et `statistics` :

1. construit et affiche une liste de 20 éléments tirés au hasard dans l'intervalle $[10, 20[$
2. afficher la moyenne de la liste des éléments
3. construire et afficher la liste des éléments plus petits ou égaux à la moyenne,
4. construire et afficher la liste des éléments plus grands strictement que la moyenne,
5. afficher la médiane de la liste des éléments
6. construire et afficher la liste des éléments plus petits ou égaux à la médiane,
7. construire et afficher la liste des éléments plus grands strictement que la médiane,

Livrable : l'archive contenant le programme `modules.py`.

Exercice 7 (tuples)

Écrire un programme Python qui :

1. définit une fonction `minmax` prenant en paramètre une liste (non vide) et renvoie la paire constituée du minimum (respectivement maximum) des valeurs de la liste,
2. utiliser cette fonction pour calculer la paire du minimum et du maximum de la liste :
   `[9, 7, 3, 2, 7, 8, 3, 8, 4, 2, 7, 0, 5, 3, 2, 0, 9, 6, 0, 5, 6, 2, 2, 4, 5, 2, 6, 3, 5, 2]`
3. afficher la moyenne de la plus grande et la plus petite des valeurs

Livrable : l'archive contenant le programme `tuples.py`.

Exercice 8 (fonctions)

On définit un nombre parfait comme un nombre qui est égal à la somme de ses diviseurs.
Écrire un programme Python qui :

1. définit une fonction `is_parfait` prenant en paramètre un entier et renvoie vrai si l'entier est parfait et faux sinon
2. utiliser cette fonction pour afficher tous les nombres parfaits plus petits que 1000

Livrable : l'archive contenant le programme `perfect.py`.

Exercice 9 (fonctions)

Le triplet (x, y, z) est pythagoricien si $x < y < z$ et $x^2 + y^2 = z^2$.
Écrire un programme Python qui :

1. définit une fonction `is_pythagoricien` prenant en paramètre un triple de entiers et renvoie vrai si le triple est pythagoricien et faux sinon
2. utiliser cette fonction pour afficher tous les triplets pythagoriciens constitués de nombres plus petits que 100

Livrable : l'archive contenant le programme `pythagore.py`.