Exo 1 : Corriger le code suivant

```
msg = ""
def repet(n: int) -> str:
    for i in range(n)
        msg += "ba"
    return msg
repet(3)
```

```
File "/le/chemin/du/fichier.py", line 6
repet(3)
File "/le/chemin/du/fichier.py", line 4, in repet
msg += "ba"
UnboundLocalError: local variable 'msg' referenced
before assignment
```

Exo 2 : idem

```
def f() -> None:
    g(a)
def g(x) -> int:
    return x + 1
a = "1"
f()
```

```
File "/le/chemin/du/fichier.py", line 6
f()
File "/le/chemin/du/fichier.py", line 2, in f
g(a)
File "/le/chemin/du/fichier.py", line 4, in g
return x + 1
TypeError: can only concatenate str (not "int") to str
```

Exo 3 : idem

```
def incremente-n(n: int) -> int:
   return n+1
```

```
File "/le/chemin/du/fichier.py", line 1
def incremente-n(n: int) -> int:
                    ^
SyntaxError: invalid syntax
```

Exo 4 : Lister toutes les erreurs qui seront levées dans l'ordre d'apparition, en précisant le message qui sera affiché (vérifier vos réponses à l'aide de `python`).

```
deF ma-fonction() -> None
   print( "mauvais nom de fonction' )
```

Exo 5 : idem

```
for in range( 4. ;
print( i )
```

Exo 6 :

a) Recopier le code suivant et exécuter-le.

```
print "Entrer une valeur entière : "
n = input
for i in range( n, -1, -1 )
   if i % 2 = 0 and i % 3
      print "{i} est divisible par 6"
   else
      print "{i} n'est pas divisible par 6"
```

b) Une `SyntaxError` est levée, quelle est la ligne incriminée ? Pourquoi ? Corriger cette erreur.

c) Corriger ainsi l'ensemble du code. Une dernière `TypeError` peut survenir si vous n'avez pas tout corrigé ; rappelez-vous que toute fonction requiert des parenthèses pour être appelée.

d) Que fait ce code ?

Exo 7 : Recopier le code suivant et corriger ses erreurs. Que fait-il ?

```
print "Quel est ton nom ?"
nom = input
0 = nbVoy
for i in range( len(nom) )
   print f"La {i+1}è lettre est { nom[i] }"
   if nom[i] in 'aeiouy'
      nbVoy + 1 = nbVoy
print Il y a {nbVoy} voyelles
```

Exo 8 : Proposer une procédure `verifTab(tab: list) -> None`, qui parcourt un tableau d'entiers en vérifiant que tous les champs soient compris entre 0 et 100. Si l'un des champs ne vérifie pas cette condition, une erreur **AssertionError** est levée avec la première valeur et son indice qui a posé problème.

Exo 9 : Compléter le code suivant afin qu'il calcule la moyenne du tableau. Si l'un des champs n'est pas un nombre positif, une erreur **TypeError** ou **AssertionError** est levée.

```
def moyenne(tab: list) -> float:
    m = ...
    for i in range( len(tab) ):
        if type( tab[i] ) not in (int, float):
            raise ...
        assert ...
        m += tab[i]
    return ...
```

Exo 10 : Proposer un code qui demande à l'utilisateur trois notes. Si l'une des notes n'est pas un nombre, une `TypeError` sera levée, si l'une des notes n'est pas entre 0 et 20, une `ValueError` sera levée. Si toutes les notes sont correctes, la moyenne sera alors affichée.

Exo 11 : On considère le code ci-contre :

Ajouter un `try/except` qui permette de rattraper la `ValueError` renvoyée par la fonction `int()` et qui affiche qu'il faut entrer un nombre positif.

```
age = -1
while age < 0:
    print( "Quel est ton âge ?" )
    age = int( input() )
print( f"Ok, tu as {age} ans." )
```

Exo 12 : Proposer un code qui demande à l'utilisateur 5 entiers pour en afficher la somme. Si l'utilisateur n'entre pas un entier, le code rattrape l'erreur.

Exo 13 : Corriger /compléter ce code `python` afin qu'il renvoie le reste de la division euclidienne de n par q. ici on se restreindra à $n \in \mathbb{Z}$ et à $q \in \mathbb{N}^*$.
e.g. `reste(9, 4)` renverra 1 car $9 = 2 \times 4 + 1$
tandis que `reste(2.1, 4)` renverra **TypeError** : "n doit être un entier"

```
def reste(n: int, q: int) -> int:
    if type(n) == float:
        raise typerror("reste: n doit être du type float")
    if q < 0:
        raise ZeroDivisionError "reste: q doit être > 0"
    return q % n

STOP = False
while not STOP:
    try:
        n = input( Entrer la valeur de n : )
        q= int( input( Entrer la valeur de q : ) )
        reste(n, q)
    except e as ZeroDivisionError:
        print( str(e) )
    else:
        STOP = True
```

Exo 14 : Proposer un code qui demande le nom de l'utilisateur. S'il entre une chaîne de caractères représentant un nombre, une `ValueError` est levée.

Exo 15 : L'organisateur du téléthon a sauvegardé les distances parcourues en 10 minutes dans sa ville. Il doit faire remonter les distances totales par catégorie. On suppose que la variable `CATEGORIES = ['CADET', 'JUNIOR', ...]` contient toutes les catégories.

On considère alors le fichier de résultats `donnees.csv` ci-contre :

dossard,categorie,distance(m)
1,BENJAMIN,1234
2,JUNIOR,3020
⋮

a) Proposer un code qui lit un fichier `csv` et met les informations dans un tableau de tableaux.

b) Proposer un code qui transforme toute les distances en entier. Si une erreur se produit, la distance entrée est nulle.

c) Proposer un code qui fait la somme des distances pour chacune des catégories et renvoie le résultat sous forme d'un dictionnaire `{'CADET' : xx, 'JUNIOR' : xx, ...}`.

Exo 16 : On souhaite demander à l'utilisateur son âge et lui renvoyer son année de naissance. Proposer

une fonction `age2annee(age: int) -> int`, qui réponde au problème et lève une **ValueError** si l'âge est négatif ou supérieur à 130, une **TypeError** si le type est incorrect.

Exo 17 : Proposer une fonction `verifTab(tab: list) -> bool`, qui analyse le contenu du tableau d'entiers compris entre 0 et 20. Si le tableau contient des flottants, une `ValueError` est levée avec l'indice qui pose problème. Si le tableau contient un autre type, une `TypeError` est levée avec l'indice qui pose problème.

Exo 18 : Proposer une fonction `verifKeys(dico: dict) -> bool`, qui vérifie que toutes les clefs d'un dictionnaire soient une chaîne de caractères commençant par une lettre majuscule. À la première erreur, une `typeError` est levée s'il ne s'agit pas d'une chaîne de caractères, une `valueError` est levée si le mot ne commence pas par une lettre majuscule.

Exo 19 : Proposer un code qui teste un utilisateur en calcul (10 multiplications d'entiers) et lui renvoye son score. Si l'utilisateur n'écrit pas un nombre, une erreur **ValueError** sera levée par la fonction de convertion `int()`. Rattrapper cette erreur et redemander à l'utilisateur sa réponse.

Exo 20 : On considère un fichier

dossier,age,distance(m),temps(s)
1,12,400,92
2,11,200,39
⋮

a) Proposer une fonction `lectTabDico(nomFichier: str) -> list`, qui lit le fichier, renvoie un tableau de dictionnaire composé de chaînes de caractères.

b) Proposer une fonction `verifDico(dico: dict) -> bool`, qui vérifie si le dictionnaire donné n'est composé que d'entiers (auquel cas `True` est renvoyé, sinon `False`.

c) Proposer une fonction `verifTabDico(tabDico: list) -> bool`, qui vérifie si tous les dictionnaires de la liste ne sont composés que d'entiers.

d) Proposer une procédure `convertDico(dico: dict) -> None`, qui transforme tous les champs du dictionnaire en entier. Si l'un des champs n'est pas convertible en floattant une erreur **TypeError** est levée, si un des champs est convertible en floattant mais pas en entier, une erreur **ValueError** est levée.

e) Proposer une procédure `convertTabDico(tabDico: list) -> None`, qui transforme tous les champs de tous les dictionnaires de la liste en entier. Les erreurs levées sont transmises.

f) Proposer une fonction `vitesse(tabDico: list) -> list`, qui renvoie un tableau avec la vitesse de chaque dossier. Si l'un des temps est nul, une erreur **ZeroDivisionError** est levée.