1. *Que va afficher le programme suivant ?*

```
x = 5.6
y = 7
print(type(y))
z = x + y
print(z, type(z))
a = "OK"
b = " boomer"
c = a + b
print(c, type(c))
l = a + y
```

```
<class 'int'>
12.6 <class 'float'>
OK boomer <class 'str'>

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

Quelques commentaires :
— le type de la variable `x` est « entier », le type de la variable `y` est « réel », et le type de la somme est forcément « réel ».
— Le signe `+` entre deux chaînes de caractère réalise la concaténation, ce qu'on voit ici, et cette concaténation est de type « chaîne ».
— La dernière instruction déclenche une erreur car même si le signe `+` a du sens à la fois pour des entiers (c'est alors l'addition) et pour des chaînes de caractères (c'est la concaténation), on ne voit pas quel sens pourrait avoir le signe `+` entre un entier et une chaîne de caractères.

2. *Écrire un programme qui saisit une valeur entière au clavier et affiche le reste de sa division par 3 (division entière).*

```
x = input('Donnez un nombre entier : ')
n = int(x)
print(n % 3)
```

3. *Le programme suivant contient une erreur de syntaxe. Quelle est cette erreur ?*

```
x = int(input("Donnez un nombre"))
if x = 1000:
   print("Le nombre est trop grand !")
```

L'erreur se situe dans l'expression figurant derrière `if` : si on veut faire une égalité, il faut utiliser le signe `==`, et pas le signe `=` qui, lui, correspond à une affectation.

4. *Écrire un programme qui affiche les 30 premiers caractères d'une chaîne de caractères (qu'on suppose assez longue).*

```
# On suppose que s est une variable 'str'
print(s[:30])
# Autre possibilité : (affiche une lettre par ligne)
for i in range(30):
   print(s[i])
```

5. *Que fait le programme suivant ? Que va-t-il afficher ?*

```
l = "Longtemps je me suis couché de bonne heure"
for c in l:
   if c in ['a', 'e', 'i']:
      print(c, end='')
```

Ce programme affiche dans l'ordre où les rencontre toutes les occurrences des lettres *a*, *e* et *i* qui apparaissent dans la chaîne `l`. L'affichage se fait sans retour à la ligne, on obtient donc : `eeeieeee`.

6. *Écrire un programme qui ouvre un fichier "`monfichier.txt`" au format texte, en lecture, et affiche toutes les lignes de longueur supérieure à 200.*

```
with open("monfichier.txt","r") as f:
  for line in f:
     if len(line)>200:
        print(line)
```

7. *Écrire une fonction `prefixe_un()` qui renvoie la première lettre d'une chaîne de caractères passée en paramètre.*

```
def prefixe_un(s):
   return s[0]
```

8. *Écrire une fonction `entier_max()` qui renvoie le plus grand entier d'une liste (d'entiers) passée en paramètre.*

```
def entier_max(l):
   max = l[0]
   for n in l:
      if n > max:
          max = n
   return max
```

9. *[On se place dans l'environnement `turtle`]. Que va afficher ce programme ? Proposer une nouvelle fonction `g()` à deux paramètres qui permette de paramétriser le nombre de côtés de la figure.*
   Ce programme affiche un triangle dont les côtés sont de longueur `l`. Si on paramétrise le nombre de côtés, on peut dessiner avec la même fonction des carrés (n=4), pentagones (n=5) etc.

```
def f(l):
   for i in range(3):
      forward(l)
      left(360/3)

f(50)
```

```
def g(l,n):
   for i in range(n):
      forward(l)
      left(360/n)
```

10. *Écrire un programme qui découpe une chaîne de caractères en une liste de mots (tokens), fait en sorte que tous les mots de longueur supérieure à 3 qui ne sont pas en majuscule commencent par une majuscule, et affiche la chaîne ainsi transformée.*

```
# on suppose la chaîne dans la variable s
liste_m = s.split()
for i in range(len(liste_m)):
   mot = liste_m[i]
   if len(mot) > 3:
      if not mot.isupper():
         liste_m[i] = mot.capitalize()

# Affichage de la chaîne :
for m in liste_m:
   print(m, end=' ')
```