

□ Exercice 1

Une réponse brève d'une ligne est attendue dans le cadre qui suit immédiatement la question, on ne *demande pas* de justification.

1. En Python, quelle est la valeur de la variable `a` après exécution de l'instruction suivante `a = (9//2)**3`

```
64
```

2. Si la variable `x` vaut 2025, quelle est la valeur de l'expression `x % 2 == 0 or x % 10==5` ?

```
True
```

3. Si la variable `lst` est la liste `[2, 3, 5, 7, 11]`, alors que vaut l'expression `len(lst) + lst[1]` ?

```
8
```

4. Pour quelle(s) valeur(s) de la variable `i` sera effectuée la boucle `for i in range(2, 17,3)`

```
2, 5, 8, 11, 14
```

5. Si `s` est la chaîne de caractères "Bug !" quel est l'affichage produit par `print(s*2 + s[4] + s[4])`

```
"Bug !Bug !!!"
```

6. Si `point` est un tuple de longueur 3, écrire l'instruction permettant de décompacter ce tuple en récupérant les 3 valeurs dans 3 variables `x`, `y` et `z`

```
x, y, z = point
```

7. Expliquer l'origine de l'erreur `IndexError: list index out of range` lorsqu'on manipule une liste en Python.

```
Cela signifie qu'on accède à un indice non valide de la liste.
```

8. Ecrire l'instruction conditionnelle permettant de tester si une variable `n` est non nulle ou supérieure ou égale à 42.

```
if n!=0 or n>=42:
```

9. Ecrire une instruction permettant de créer *par compréhension* la liste `l=[0, 5, 10, 15, 20, 25, 30]`

```
l = [5*n for n in range(7)]
```

10. Quelle sera le contenu de la liste `lst2` après exécution des instructions suivantes :

```
1 lst1 = [1, 3, 9, 27]
2 lst2 = lst1
3 lst1.append(81)
```

```
[1, 3, 9, 27, 81]
```

□ Exercice 2 : QCM

Dans cette exercice, une question peut avoir *zéro une ou plusieurs bonnes réponses*. Pour chaque question, cocher les cases correspondantes aux bonnes réponses.

- Que peut-on dire de la variable définie par l'instruction `a = 21/4`?
 `a` est de type `int` `a` est de type `float` `a` vaut 5 `a` vaut 5.25
- Quelles sont les propositions exactes concernant les fonctions en Python?
 Leur définition commence par `def`
 Elles contiennent toujours au moins une instruction `return`
 Elles peuvent prendre zéro argument
 Elles doivent contenir un test ou une boucle
- Quel(s) test(s) sont vraies si et seulement si l'entier `n` est paire?
 `2%n==0` `n//2==0` `n%2==0` `n%10 == 2` `n%2!=1`
- Si `s` est une chaîne de caractère (type `str`), cocher les instructions valides (celles qui ne déclenchent pas d'erreur)
 `s + s` `s + 2` `s*3` `s + "2"` `s*"3"`
- Parmi les types de Python suivants lesquels sont itérables?
 `int` `float` `bool` `tuple` `str`
- On suppose que `c` est un entier valant 5, quelles expressions seront évaluées à `True`?
 `3!=c and 7>2*c` `not (4==c)` `True or (c==12)` `5>=c>=5`
- Si `lst` est du type `list` parmi les programmes suivants, quels sont ceux qui vont afficher les éléments de `lst`?


```
for elt in range(lst):  
    print(elt)
```

```
for i in range(len(lst)):  
    print(lst[i])
```



```
for elt in lst:  
    print(elt)
```

```
for i in range(len(lst)):  
    print(i)
```
- Quels sont les affirmations vraies concernant le type `tuple` de Python?
 On peut modifier un élément d'un tuple après sa création
 Tous les éléments d'un tuple doivent être du même type
 On peut accéder à l'élément d'indice `i` du tuple `t` avec `t[i]`
 On peut utiliser `append` sur un tuple
 La variable `var = ("PCSI", 2025, "Python")` permet de définir un tuple
- Si `s` est la chaîne de caractère `"cet exercice"`, quelles tranches contiennent `"ce"`?
 `s[:2]` `s[0:2]` `s[10:]` `s[len(ex)-2:]`
- Après exécution de l'instruction `l = [7*i for i in range(10)]`, quelles affirmations concernant `l` sont vraies?
 `len(l)` vaut 10
 `l` est la liste `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`
 `l` est un itérable
 0 est l'un des éléments de `l`

□ Exercice 3 : Définir une fonction

Ecrire en Python une fonction `nb_occ` qui prend en argument un caractère `car` et une chaîne de caractères `chaîne` et renvoie le nombre d'apparitions de `car` dans `chaîne`. Par exemple, `nb_occ("o", "toto")` doit renvoyer 2, et `nb_occ("o", "PCSI")` doit renvoyer 0.

```
1 def nb_occ(car, chaine):
2     # On initialise à 0 le nombre d'occurrence
3     cpt = 0
4     # On parcourt par élément la chaine
5     for elt in chaine:
6         # Si un élément est de la chaine est le caractère cherché
7         if elt == car:
8             # On incrémente le nombre d'apparitions
9             cpt += 1
10    return cpt
```

□ **Exercice 4** : *Exercice bonus*

Ecrire une fonction `deuxmin` qui prend en argument une liste d'entiers contenant au moins deux éléments et qui renvoie les deux plus petits éléments de cette liste. Par exemple `deuxmin([-1, 6, 0, 2, -3, 8])` renvoie `-3, -1`.

```
1 def deuxmin(lst):
2     assert len(lst)>=2, "La liste doit avoir au moins deux éléments"
3     if lst[0]<lst[1]:
4         min1 = lst[0]
5         min2 = lst[1]
6     else:
7         min1 = lst[1]
8         min2 = lst[0]
9     '''On parcourt le reste de la liste :
10    - Si un élément est plus petit que le plus petit (min1), il devient le nouveau
→ min1 et l'ancien min1 devient min2.
11    - Sinon, si l'élément est plus petit que le second minimum (min2), il devient
→ le nouveau min2.'''
12    for i in range(2, len(lst)):
13        if lst[i]<min1:
14            min2 = min1
15            min1 = lst[i]
16        elif lst[i]<min2:
17            min2 = lst[i]
18    return min1, min2
```