La complexité des types de bases en Python un aspect pratique

Nom :	**Note : / 20**
	Classe :

Q1. Proposez un algorithme pour réaliser cette fonction en pseudo langage.

Q2. Expliquez le fonctionnement de la ligne de code.

```
nouvelle_valeur = random.randrange(0,valeur_max)
```

Q3. Rappeler brièvement ce que signifie la complexité d'un algorithme.

Q4. Quelle est son utilité ?

Q5. Que signifie une complexité en $O(1)$?

Q6. Que signifie une complexité en $O(n)$?

Analyse du fonctionnement, notez les durée d'exécution en µS

Nombre de valeurs	**Méthode basique**	**Méthode avancée**
1000		
2000		
4000		
5000		
10000		
20000		

Q7. Déterminez la complexité de la solution basique et la complexité de la solution avancée à partir des résultats obtenus.

Q8. En comparant le code des deux fonctions proposer une explication de la différence de comportement.

Algorithme basique

```
44  def initTableBasic(nombre_elements):
45      '''
46      On initialise une table à [nombre_elements] différents donc
47      sans aucun doublons
48      '''
49      global table
50
51      # La table des valeurs à construres
52      table = []
53
54      # On ajuste la valeur max pour assurer le tirage d'aucun doublons
55      # avec suffisamment de valeurs possibles
56      valeur_max = 10 * nombre_elements
57
58      # Tant que l'on a pas atteint le nombre de valeurs souhaité
59      while len(table) < nombre_elements:
60          # On tire au sort une nouvelle valeur
61          nouvelle_valeur = random.randrange(0,valeur_max)
62          # Si la nouvelle valeur n'est pas déjà dans table
63          # On l'ajoute
64          if nouvelle_valeur not in table:
65              table.append(nouvelle_valeur)
```

Algorithme avancé

```
15  def initTableAvance(nombre_elements):
16      '''
17      On initialise une table à [nombre_elements] différents donc
18      sans aucun doublons
19      '''
20      global table
21
22      element_deja_present = set()
23
24      # La table des valeurs à construires
25      table = []
26
27      # On ajuste la valeur max pour assurer le tirage d'aucun doublons
28      # avec suffisamment de valeurs possibles
29      valeur_max = 10 * nombre_elements
30
31      # Tant que l'on a pas atteint le nombre de valeurs souhaité
32      while len(table) < nombre_elements:
33          # On tire au sort une nouvelle valeur
34          nouvelle_valeur = random.randrange(0,valeur_max)
35          # Si la nouvelle valeur n'est pas déjà dans table
36          # On l'ajoute
37          if nouvelle_valeur not in element_deja_present:
38              element_deja_present.add(nouvelle_valeur)
39              table.append(nouvelle_valeur)
```