Je vais vous aider à résoudre ces exercices étape par étape. Commençons par la première partie du quiz.

a) type

1. Créez une variable string contenant "toto" et une variable number contenant le nombre 10.

```python

string = "toto"

number = 10

```

2. Convertissez string en unicode et number en float.

```python

unicode\_string = string.encode('utf-8')

float\_number = float(number)

```

3. Inverser le contenu des variables string et number.

```python

string = string[::-1]

number = number[::-1]

```

b) List

1. Créez une liste first\_list contenant les entiers suivants : 1, 3, 2, 7, 4, 10, 46.

```python

first\_list = [1, 3, 2, 7, 4, 10, 46]

```

2. Affichez les trois premiers éléments de first\_list - en une seule ligne de code.

```python

print(first\_list[:3])

```

3. Créez une deuxième liste qui ne contient que les 3ème, 4ème et 5ème éléments de first\_list. Votre indexation de liste doit utiliser 3 caractères au maximum (non inclus les espaces).

```python

second\_list = first\_list[2:5]

```

4. Concaténez first\_list et second\_list pour former third\_list.

```python

third\_list = first\_list + second\_list

```

5. Utilisez la fonction zip pour associer les valeurs de first\_list et second\_list dans la variable tuple\_of\_lists.

```python

tuple\_of\_lists = zip(first\_list, second\_list)

```

6. Ajoutez le nombre 5 à first\_list.

```python

first\_list.append(5)

```

7. Ajoutez un élément null à third\_list.

```python

third\_list.append(None)

```

**### You want to make parameter `n` optional and set its value to 2 by default. How do you do it?**

def concatenate\_list(my\_list, n=2):

return my\_list \* n

8. Écrivez une fonction qui prend comme arguments une liste my\_list et un entier n et renvoie le résultat de n concaténations de my\_list avec lui-même. Par exemple, si my\_list=[1,2,3], n=2 --> [1,2,3,1,2,3]

```python

def concatenate\_list(my\_list, n=):

return my\_list \* n

```

9. Avec une boucle while, imprimez chaque élément de third\_list jusqu'à l'élément null.

```python

i = 0

while i < len(third\_list):

print(third\_list[i])

if third\_list[i] is None:

break

i += 1

```

10. Calculez, avec une boucle for, le nombre d'entiers pairs présents dans la liste first\_list.

```python

even\_count = 0

for num in first\_list:

if num % 2 == 0:

even\_count += 1

```

11. Réécrivez l'expression suivante avec une boucle for multiligne.

```python

even\_numbers = []

for element in first\_list:

if element % 2 == 0:

even\_numbers.append(element)

```

12. Créez une fonction qui prend une chaîne de caractères en entrée et renvoie la première lettre simple de la chaîne.

```python

def first\_letter(input\_string):

for char in input\_string:

if char.isalpha():

return char

```

c) Dictionnaires

1. Créez un dictionnaire voiture avec les clés et valeurs suivantes :

```python

car = {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

```

2. Accédez à la valeur année.

```python

year = car['year']

```

3. Utilisez une boucle pour afficher toutes les clés du dictionnaire.

```python

for key in car:

print(key)

```

4. Utilisez une boucle pour afficher toutes les valeurs du dictionnaire.

```python

for value in car.values():

print(value)

```

5. Utilisez une boucle pour afficher tous les éléments du dictionnaire. Affichez également l'index de chaque itération de boucle (fonction enumerate).

```python

for i, item in enumerate(car.items()):

print(f"Item {i}: {item}")

```

6. Avec l'aide d'une compréhension de dictionnaire, créez un dictionnaire avec les clés et valeurs suivantes :

```python

new\_dict = {'cle\_1': car, 'cle\_2': car, 'cle\_3': car}

```

d) Fonctions

1. Créez une fonction Fibonacci appelée fibonacci (la séquence Fibonacci : 0 1 1 2 3 5 8 13 ...)

```python

def fibonacci(n):

seq = [0, 1]

while len(seq) <= n:

seq.append(seq[-1] + seq[-2])

return seq[:n+1]

```

2. Exécutez votre fonction. fibonacci(100) doit être exécuté en moins d'une seconde.

```python

print(fibonacci(100))

```

3. Créez la même fonction en utilisant un générateur

```python

def generate\_fibonacci(n):

a, b = 0, 1

for \_ in range(n):

yield a

a, b = b, a + b

```