QUIZ 23 - MODIFICATION DE LA PORTÉE AVEC GLOBAL ET NONLOCAL

(4 points possibles)

Variables globales (1)

Peut-on modifier une variable globale depuis une fonction ?

- [ ] Jamais
- [x] Toujours du moment qu'il y a une affectation dans la fonction
- [ ] Uniquement si la variable est déclarée globale avec la directive `global` dans la fonction ✔

✖

EXPLANATION

Une variable globale ne peut être modifiée depuis une fonction que si elle est déclarée comme `global` dans la fonction.

Variables globales (2)

```
x = 1
def f():
    global x
    x = 10
f()
print(x)
```

Que va afficher l'exécution de ce code ?

- [ ] 10 ✔

- [x] 1
- [ ] x

✖

EXPLANATION

La directive `global` permet de modifier la variable `x` dans l'espace de nommage du module. Si la variable déclarée comme global n'existe par encore dans le module, elle est créée à la première affectation.

Variables et obj

Dans quel cas la fonction modifie en place **l'objet** `[1, 2]`

initialement référencé par la variable `var` ?

Proposition 1

```
var = [1, 2]
def f():
    var = 20
f()
```

Proposition 2

```
var = [1, 2]
def f():
    var = [1, 3]
f()
```

Proposition 3

```
var = [1, 2]
def f():
    var.append(3)
f()
```

Proposition 4

```
    var = 1
f()
```

Proposition 5

```
var = [1, 2]
def f():
    global var
    var.append(10)
f()
```

Choisissez une ou plusieurs propositions.

- [ ] Proposition 1
- [x] Proposition 2
- [x] Proposition 3 ✔
- [ ] Proposition 4
- [x] Proposition 5 ✔

✖

EXPLANATION

Ce quiz joue avec les notions de modification d'objets et de références. Une variable référence un objet et l'opération d'affectation permet de changer l'objet référencé par une variable. Par contre, sans changer l'objet référencé par une variable, on peut modifier un objet mutable par effet de bord. Regardons maintenant les différentes propositions.

La fonction dans la proposition 1 ne modifie pas l'objet puisque la fonction crée une variable locale `var` distincte de la variable globale `var`.

La fonction dans la proposition 2 ne modifie pas non plus l'objet puisqu'ici encore la fonction crée une variable locale `var` distincte de la variable globale `var`.

Ensuite, la fonction fait un append sur `var`, elle modifie donc en place l'objet référencé par la variable globale `var`.

La fonction dans la proposition 4 ne modifie pas l'objet référencé. C'est le cas le plus subtil. La fonction déclare la variable `var` comme globale, donc, dans la fonction, on modifie bien la variable globale `var` en lui affectant l'objet entier 1, mais on ne modifie pas en place l'objet initialement référencé par `var`.

La fonction dans la proposition 5 modifie bien en place l'objet référencé. Dans ce cas, l'utilisation de la directive `global` est inutile, mais légale.

global et nonlocal (4)

Que va afficher le code suivant ?

```
a = 10
def f():
    global a
    a = a + 10
    b = 20
    def g():
        nonlocal b
        b = b + 20
    g()
    return b

print(f() + a)
```

- Une exception
- `None`
- 10
- 30
- 40 ✖

- 95.5

EXPLANATION

La fonction `f` modifie la variable globale `a` qui vaut donc 20 après l'appel de `f`. La fonction `g` modifie la variable `b` dans `f`, donc après l'appel de `g`, la fonction `f` retourne 40. Au final, on affiche la somme de la valeur de retour de `f` (soit 40) avec la valeur de `a` (soit 20). On voit donc apparaître 60.

Vous avez utilisé 3 essais sur 3