

NSI | 1,2,3

Nom :

Prénom :



CORRIGÉ DEVOIR #1

1) Donnez les valeurs que prennent chacune des onze variables définies dans le programme suivant :

```
1 x,y = 8,5
2 a = x+y*2
3 b = x/y
4 c = x//y
5 d = x*y
6 e = (x-y)**2
7 f = str(x)+str(y)
8 g = str(x*y)
9 h = x<y or x==8
10 i = int(a/b)
11 j = type(h)      # int, float, bool ou str
12 k = type(b)      # int, float, bool ou str
```

a: 18 b: 1.6 c: 1 d: 3
e: 9 f: "85" g: "40" h: True
i: 11 j: bool k: float

2) Même question avec les six variables définies dans le programme suivant :

```
1 x,y = "chat", "on"
2 a = x+y
3 b = y+"a"+x
4 c = "h" in a
5 d = "t" not in b
6 e = x[3]+a[1]+b[0]+y[1]
7 f = len(b)
```

a: "chaton" b: "onachat" c: True
d: False e: "thon" f: 7

3) Donnez la valeur obtenue dans la console à l'exécution du programme suivant :

```
1 x=int(input("Entrez un nombre: "))
2 if x>=15:
3     print("certainement")
4 elif x<10:
5     print("peut-être")
6 elif x==10:
7     print("probablement")
8 else:
9     print("sans doute")
```

a: L'utilisateur entre la valeur 3:

☐ "certainement" ☒ "peut-être" ☐ "probablement" ☐ "sans doute" ☐ ValueError

b: L'utilisateur entre la valeur 15:

☒ "certainement" ☐ "peut-être" ☐ "probablement" ☐ "sans doute" ☐ ValueError

c: L'utilisateur entre la valeur 5.0:

☐ "certainement" ☐ "peut-être" ☐ "probablement" ☐ "sans doute" ☒ ValueError

d: L'utilisateur entre la valeur 12:

☐ "certainement" ☐ "peut-être" ☐ "probablement" ☒ "sans doute" ☐ ValueError

4) Donnez la valeur obtenue dans la console à l'exécution du programme suivant :

```
1 for i in range(5,21,4):
2     print(i,end=" ")
```

Affichage: 5 9 13 17

`range(5, 21, 4)` produit les entiers de 5 à 21 **exclu** avec un pas de 4 : 5, 9, 13, 17. Le suivant serait 21, mais la borne de fin n'est jamais atteinte.

Le paramètre `end=" "` remplace le retour à la ligne final de `print` par un espace : les quatre valeurs s'affichent donc sur une seule ligne, séparées par un espace.

5) Donnez la valeur obtenue dans la console à l'exécution du programme suivant :

```
1 a="bateau"
2 for lettre in a:
3     if lettre=="b" or lettre=="a":
4         continue
5     print(lettre,end="")
```

Affichage: **teu**

La boucle parcourt la chaîne "bateau" caractère par caractère : b, a, t, e, a, u.

Quand la lettre vaut "b" ou "a", l'instruction `continue` interrompt l'itération en cours et passe **directement à la lettre suivante** : le `print` n'est pas exécuté. Les lettres b, a et a sont donc sautées.

Il reste t, e et u. Avec `end=""`, rien n'est ajouté entre deux affichages : les lettres sont collées.

6) Donnez les valeurs que prennent chacune des neuf variables définies dans le programme suivant (on rappelle que **and** et **not** sont prioritaires sur **or**) :

```
1 x,y,z = 18,6,6.0
2 a = x>z
3 b = y==z
4 c = x<y or z==y
5 d = not (x==y or z>x)
6 e = x>=z and y!=z
7 f = x==3*y or z<y
8 g = z is y
9 h = z==y
10 i = not a or b and h
```

a: ☒ **True** ☐ False b: ☒ **True** ☐ False c: ☒ **True** ☐ False

d: ☒ **True** ☐ False e: ☐ True ☒ **False** f: ☒ **True** ☐ False

g: ☐ True ☒ **False** h: ☒ **True** ☐ False i: ☒ **True** ☐ False

7) Observez attentivement les lignes de code des programmes suivant :

Que va afficher la console ? Expliquer en détaillant votre réponse :

Affichage: **visa**

`len(a)` vaut 8, donc `i` prend les valeurs 0, 1, 2, ..., 7 (indices des caractères de "vaisseau").

`i % 2 == 0` est vrai quand le reste de la division de `i` par 2 est nul, c'est-à-dire pour les **indices pairs** 0, 2, 4 et 6.

On concatène donc `a[0] = "v"`, `a[2] = "i"`, `a[4] = "s"` et `a[6] = "a"` : `b` vaut successivement "v", "vi", "vis" puis "visa".

Le `print(b)` n'est pas indenté : il est en dehors de la boucle et n'affiche qu'une seule fois le résultat final.

```
1 s="stressed"
2 b=""
3 for i in s:
4     b=i+b
5     print(b)
```

Que va afficher la console ? Expliquer en détaillant votre réponse :

Affichage: **desserts**

La boucle `for i in s` parcourt directement les caractères de "stressed" (ici `i` est une lettre, pas un indice).

L'instruction `b = i + b` place chaque nouvelle lettre **devant** la chaîne déjà construite (et non derrière comme `b = b + i`).
`b` vaut successivement "s", "ts", "rts", "erts", "serts", "sserts", "esserts", "desserts" : le programme **inverse la chaîne**.

`print(b)` est hors de la boucle : seul le résultat final "desserts" est affiché (« stressed » lu à l'envers).