

Découvrir et approfondir python



Structures algorithmiques, fonctions, compréhension de liste :



V W2-05 Introduction aux tests if.mp4



V W2-06 Introduction aux boucles for et aux fonctions.mp4



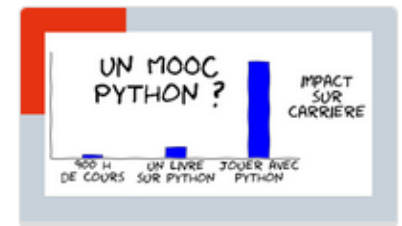
V W2-07 Introduction aux compréhensions de liste.mp4

Vidéos proposées par l'INRIA sur



Python 3 : des fondamentaux aux concepts avancés du langage

Thierry PARMENTELAT et Arnaud Legoud



Pour bien comprendre et approfondir :

Exemples de codes :

a) Que répond le code ci-contre

- ☐ le code est invalide
- ☐ le code affiche 'oui'
- ☐ le code affiche 'non'

```
if i < 2
    print ("Oui")
else:
    print ("Non")
```

b) Que dire du code ci-contre

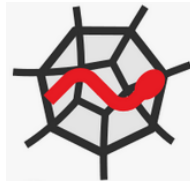
- ☐ le code est invalide
- ☐ le code est correct
- ☐ le code répond 'OUI'
- ☐ il n'est pas recommandé

```
# code pas du tout recommandé
if 'a' in 'pas du tout recommande':
    succes = True
    print('OUI')
else:
    print('NON')
```

c) Que dire du code ci-contre

- ☐ le code est invalide
- ☐ le code est correct
- ☐ le code répond 'e mais pas b'

```
if 'e' in entree:
    if 'b' in entree:
        cas11 = True
        print('e et b')
    else:
        cas12 = True
        print('e mais pas b')
```



d) Que contient resultat après l'exécution du code ci-contre ?

- ☐ une erreur
- ☐ 1,2,'3',4,'Fin'
- ☐ 1,2,3,4,FIN
- ☐ '1,2,3,4,FIN'

```
a = []
for i in [1, 2, '3', 4, 'FIN']:
    a.append(str(i))
resultat = ",".join(a)
```

e) Pour obtenir '1 2 3'

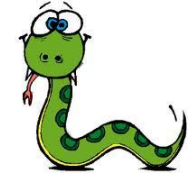
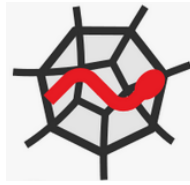
- ☐ to_str([1, 2, 3])
- ☐ to_str(['1', '2', '3'])
- ☐ to_str('123')
- ☐ to_str(['123'])

```
def to_str(a):
    tmp = []
    for i in a:
        tmp.append(str(i))
    return " ".join(tmp)
```

f) On a entree = [1, 2, 3, 4, 5, 6, 7, 8, 9]

Quels sont les codes valides :

- ☐ carres = [for x in entree: x**2]
- ☐ carres = [x**2 for x in entree]
- ☐ carres = [for x in entree: if x % 3 == 0: x**2]
- ☐ carres = [x**2 for x in entree: if x % 3 == 0]
- ☐ carres = [x**2 for x in entree if x % 3 == 0]



Corrigé :

Exemples de codes :

a) Que répond le code ci-contre

- ☒ le code est invalide
- ☐ le code affiche 'oui'
- ☐ le code affiche 'non'

```
if i < 2
    print ("Oui")
else:
    print ("Non")
```

b) Que dire du code ci-contre

- ☐ le code est invalide
- ☒ le code est correct
- ☒ le code répond 'OUI'
- ☒ il n'est pas recommandé

```
# code pas du tout recommandé
if 'a' in 'pas du tout recommande':
    succes = True
    print('OUI')
else:
    print('NON')
```

c) Que dire du code ci-contre

- ☒ le code est invalide
- ☐ le code est correct
- ☐ le code répond 'e mais pas b'

```
if 'e' in entree:
    if 'b' in entree:
        cas11 = True
        print('e et b')
    else:
        cas12 = True
        print('e mais pas b')
```

d) Que contient resultat après l'exécution du code ci-contre ?

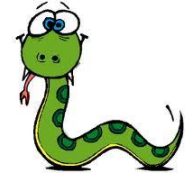
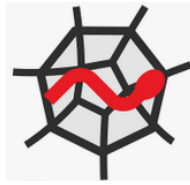
- ☐ une erreur
- ☐ 1,2,'3',4,'Fin'
- ☐ 1,2,3,4,FIN
- ☒ '1,2,3,4,FIN'

```
a = []
for i in [1, 2, '3', 4, 'FIN']:
    a.append(str(i))
resultat = ",".join(a)
```

e) Pour obtenir '1 2 3'

- ☒ to_str([1, 2, 3])
- ☒ to_str(['1', '2', '3'])
- ☒ to_str('123')
- ☐ to_str(['123'])

```
def to_str(a):
    tmp = []
    for i in a:
        tmp.append(str(i))
    return " ".join(tmp)
```



f) On a `entree = [1, 2, 3, 4, 5, 6, 7, 8, 9]`

Quels sont les codes valides :

☐ `carres = [for x in entree: x**2]`

☒ `carres = [x**2 for x in entree]`

☐ `carres = [for x in entree: if x % 3 == 0: x**2]`

☐ `carres = [x**2 for x in entree: if x % 3 == 0]`

☒ `carres = [x**2 for x in entree if x % 3 == 0]`
