PYTHON

S02 – E02

A l'abordage, moussaillons!

BIB 18.04.17

Winter - Licence MIT

BOUCLES

WHILE

Pseudo Code

```
TANT QUE expr ALORS action
```

Python

```
>>> while expr:
...     action
```

Pseudo Code

```
TANT QUE expr ALORS action1
SINON action2
```

Python

```
>>> while expr:
...     action1
... else:
...     action2
```

FOR

Pseudo Code

```
POUR TOUT x DANS y ALORS action
```

Python

```
>>> for x in y:
...     action1
```

Pseudo Code

```
POUR TOUT x DANS y ALORS action1
SINON action2
```

Python

```
>>> for x in y:
...     action1
... else:
...     action2
```

BREAK

```
>>> a = 0
>>> while True:
...     if a == 10:
...         break
...     print(a)
...     a += 1
```

TABLEAUX

LISTE

INDEX

```
>>> liste = [43, 233.45, 'Hello', True]
```

```
>>> liste[0]
43
```

```
>>> liste[-1]
True
```

```
>>> liste[-1] = False
[43, 344.45, 'Hello', False]
```

OPÉRATIONS

```
>>> liste = [0, 1, 2]
```

```
>>> len(liste)
3
```

```
>>> liste + [4, 5]
[0, 1, 2, 3, 4, 5]
```

```
>>> liste * 3
[0, 1, 2, 0, 1, 2, 0, 1, 2]
```

```
>>> del liste[0]
[1, 2]
```

SLICING

```
>>> liste = [1, 2, 3, 4, 5, 6]
```

```
>>> liste[A:B:P]
```

A: Index de départ (inclus)

B: Index d'arrivé (exclus)

P: Pas

```
>>> liste[3:]
[4, 5, 6]
```

```
>>> liste[:-1]
[1, 2, 3, 4, 5]
>>> liste[:3]
[1, 2, 3]
```

```
>>> liste[1:3]
[2,3]
```

```
>>> liste[0:len(liste):2]
[1, 3, 5]
```

```
>>> liste[::-1]
[6, 5, 4, 3, 2, 1]
```

MÉTHODES

```
>>> animaux = []
```

```
>>> animaux.append('Chat')
['Chat']
```

```
>>> animaux.extend(['Poisson', 'Mouette'])
['Chat', 'Poisson', 'Mouette']
```

```
>>> animaux.insert(0, 'Python')
['Python', 'Chat', 'Poisson', 'Mouette']
```

```
>>> animaux.remove('Chat')
['Python', 'Poisson', 'Mouette']
```

```
>>> animaux.index('Python')
0
```

LISTE EN COMPRÉHENSION

```
>>> liste = [x**2 for x in range(0, 5)]
[0, 1, 4, 9, 16]
```

```
>>> liste = [x for x in range(0, 5) if x % 2 == 0]
[0, 2, 4]
```

CHAÎNES DE CARACTÈRES

```
>>> hello = "Hello, Wolrd!"
>>> hello[0] = 'T'
TypeError: 'str' object does not support item assignment
```

TUPLE

```
>>> t = (10, 20, 30)
>>> t[1] = 40
TypeError: 'tuple' object does not support item assignment
```

UNPACKING

```
>>> (a, b, c) = (1, 2, 3)
>>> (a, b, c) = 1, 2, 3
>>> a, b, c = (1, 2, 3)
>>> a, b, c = 1, 2, 3
```

COLLECTION

```
>>> s = {0, 1, 2, 2, 3, 4, 5, 6, 6, 7}
{0, 1, 2, 3, 4, 5, 6, 7}
```

```
>>> s2 = {4, 3, 2, 1, 0}
{0, 1, 2, 3, 4}
```

```
>>> s[0]
TypeError: 'set' object does not support indexing
```

```
>>> s[0] = 1
TypeError: 'set' object does not support item assignment
```

```
>>> s.add(8)
{0, 1, 2, 3, 4, 5, 6, 7, 8}
```

```
>>> s.remove(0)
{1, 2, 3, 4, 5, 6, 7, 8}
```

DICTIONNAIRES

```
>>> d = {'name': 'John', 'age': 42, 'job': 'Hacker'}
{'age': 42, 'job': 'Hacker', 'name': 'John'}
```

```
>>> d['name']
'John'
```

```
>>> d['age'] = 66
{'age': 66, 'job': 'Hacker', 'name': 'John'}
```

```
>>> d['city'] = 'New York City'
{'age': 42, 'city': 'New York City', 'job': 'Hacker', 'name': 'Jo
```

ITERATIONS

```
>>> for key, value in d.items():
...     print(f'{key}: {value}')
```

RÉCAPITULATIF

	List	**String**	**Tuple**	**Set**	**Dict**
Séquence	**Oui**	**Oui**	**Oui**	Non	Non
Modifiable	**Oui**	Non	Non	Non	**Oui**
Ajout Supression	**Oui**	Non	Non	**Oui**	**Oui**

FONCTIONS

PRÉSENTATION

- Programme à l'intérieur d'un programme
- Permet de découper son code proprement
- Suit les conditions de nommages des variables

SQUELETTE

```
>>> def somme(a, b):
...     """Retourne la somme de a
...     return a + b
```

```
>>> somme(1, 2)
3
```

```
>>> c, d = 2, 2
>>> somme(c, d)
4
```

- En tête
- Docstring
- Corps

PARAMÈTRES

```
def somme(a, b):
```

```
def somme(a, b=0):
```

```
>>> somme(10)
10
```

```
>>> def somme(a=0, b):
...     return a + b
SyntaxError: non-default argument follows default argument
```

```
def somme(a=0, b=0):
```

```
>>> somme()
0
```

DOCUMENTATION

```
>>> help(somme)
```

```
Help on function somme in module __main__:

somme(a, b)
    Retourne la somme de a et b
```

TD

QCM

CONSIGNES

- Une **série** de questions
- Une question a **plusieurs** réponses numérotées
- Seule une réponse **correcte**
- Bonne réponse : **1 point**
- Mauvaise réponse : **0 point**
- Affiche le **score** à la fin

CONSEILS

- Découpez votre code en **plusieurs fonctions**
- Stockez vos questions dans une **liste**

EXEMPLE

```
question = [
    "Couleur du cheval blanc d'Henry 4?",
    [('Blanc', True),
     ('Brun', False),
     ('Rouge', False),
     ('Noir', False)]
]

def afficher_question(question):
    print(question[0])
    for numero, reponse in enumerate(question[1]):
        print(f'{numero} - {reponse[0]}')
```