**Liste avancée, dictionnaires et boucles**

**I. Boucles For**

**A. Boucles For et dictionnaires**

Nous pouvons parcourir à la fois les clés et les valeurs d’un dictionnaire.

my\_books = {

"title": "Harry Potter",

"author": "JK Rowling",

}

for x, y in my\_books.items():

print("the" + x + "is" + y)

>> the title is Harry Potter

the author is JK Rowling

**B. Opérateur de boucles**

**range(start, stop[, step]) : itérateur dans les boucles.**

print(list(range(1, 10, 2)))

>> [1, 3, 5, 7, 9]

**enumerate(iterable) : énumérer chaque élément de l' *itérable***

for item in enumerate('abcd'):

print(item)

>>

(0, 'a') # Syntax : (index , value)

(1, 'b')

(2, 'c')

(3, 'd')

for (index\_count, letter) in enumerate('abcd'):

print('At index {} the letter is {}'.format(index\_count, letter))

>>

At index 0 the letter is a

At index 1 the letter is b

At index 2 the letter is c

At index 3 the letter is d

**zip(itérable,..) : concat [itérables, …] dans un tuple.**

list1 = [1,2,3]

list2 = ['a','b','c']

list3 = [1.1, 2.2, 3.3, 4.4, 5.5]

for item in zip(list1, list2, list3): # only go as far it is possible

print(item)

>>

(1, 'a', 1.1)

(2, 'b', 2.2)

(3, 'c', 3.3)

**C. Pour le reste**

Cette elsepartie est facultative. Lorsqu'elle est incluse, elle est toujours exécutée une fois la forboucle terminée, sauf si une breakinstruction est rencontrée.

for i in range(1, 3):

print(i)

else:

print('The for loop is over')

>>

1

2

The for loop is over

**D. Tandis que d'autre part**

while some\_condition:

# do something

else:

# do another thing

x = 0

while x < 2:

print(f'x is {x}')

x += 1

else:

print('x is bigger than 2')

>>

x is 0

x is 1

x is bigger than 2

**E. Pause, Continuer, Passer**

* break: briser la boucle (arrêter l'exécution d'une instruction en boucle)

si vous *sortez* d'une boucle forou , aucun bloc whilede boucle correspondant **n'est** exécuté.else

for letter in 'Leonardo':

if letter == 'a':

break

print(letter, end='') # end='' renders each letter next to the other

>> Leon

while True:

s = input('Enter something : ')

if s == 'quit':

break

print('Length of the string is', len(s))

print('Done')

* continue: revenir au début de la boucle ( *continuer* à l'itération suivante de la boucle)

for letter in 'Leonardo':

if letter == 'o':

continue

print(letter, end='') # dont execute for 'o' letter

>> Lenard

while True:

s = input('Enter something : ')

if s == 'quit':

break

if len(s) < 3:

print('Too small')

continue

print('Input is of sufficient length')

* pass: ne rien faire

for item in [1,2,3]:

# comment

pass # to avoid the error

print('Finish the script')

>> Finish the script

**II. Compréhension de liste : un moyen rapide de créer une liste**

De nombreux objets en Python sont « itérables », ce qui signifie que nous pouvons parcourir chaque élément de l’objet.

Les compréhensions de listes sont un moyen unique de créer rapidement une liste avec Python.

**A. La manière de base d'ajouter un élément à une liste**

my\_number = '1234'

my\_list = []

for num in my\_number:

my\_list.append(num)

print(my\_list)

>> ['1', '2', '3', '4']

**B. La méthode de compréhension de liste**

my\_number = '1234'

my\_list = []

my\_list = [num for num in my\_number]

print(my\_list)

>> ['1', '2', '3', '4']

**Exemples avec la méthode de portée**

my\_list = [x for x in range(0,6)]

print(my\_list)

>> [0, 1, 2, 3, 4, 5]

my\_list = [x\*\*2 for x in range(0,6)] # square

print(my\_list)

>> [0, 1, 4, 9, 16, 25]

my\_list = [x for x in range(0,11) if x%2 == 0] # only even

print(my\_list)

>> [0, 2, 4, 6, 8, 10]

**C. La méthode de base pour ajouter un élément à une liste avec une boucle imbriquée**

my\_list = []

for i in [2, 3, 4]:

for j in [100, 200, 300]:

my\_list.append(i\*j)

print(my\_list)

>> [200, 400, 600, 300, 600, 900, 400, 800, 1200]

**D. La méthode de compréhension de liste**

my\_list = []

my\_list = [(i\*j) for i in [2, 3, 4] for j in [100, 200, 300]]

print(my\_list)

>> [200, 400, 600, 300, 600, 900, 400, 800, 1200]

**E. Compréhension du dictionnaire**

dictionary = {key: value for var in iterable}

family\_age = {'Lea': 12, 'Mark': 25, 'George': 50}

new\_year = 1

new\_family\_age = {name: age+new\_year for (name, age) in family\_age.items()}

print(new\_family\_age)

>> {'Lea': 13, 'Mark': 26, 'George': 51}