Session 9 - En comprehension -2022 - A.U. 23-24

November 29, 2023

1 Un itérable entant que dictionnaire

[1]:
```
s = "ali"  # est une chaine de caractère
dict_s = { 0: "a", 1: "l", 2: "i", }  # est un dictionnaire qui peut␣
 ↪représenter la chaine de caractère s

l = [1, 2, 3, 4, 5]  # est une liste
dict_l = { 0: 1, 1: 2, 2: 3, 3: 4, 4: 5, }  # est un dictionnaire qui peut␣
 ↪représenter la liste l

t = (1, 2, 3, 4, 5)  # est un tuple
dict_t = { 0: 1, 1: 2, 2: 3, 3: 4, 4: 5,
}  # est un dictionnaire qui peut représenter le tuple t

r = range(5)  # est un range
dict_r = { 0: 0, 1: 1, 2: 2, 3: 3, 4: 4,
}  # est un dictionnaire qui peut représenter le range r

s = {1, 2, 3, 4, 5}  # est un set
dict_s = { 0: 1, 1: 2, 2: 3, 3: 4, 4: 5,
}  # est un dictionnaire qui peut représenter le set s
```

Exercice: lire un entier?

[2]:
```
# Sans exigence
x=int(input('donner un entier'))

#Plus exigeant
x_str = input("donner un entier")
if x_str.isdigit():
    x = int(x_str)
```

2 Itérable en compréhension

- Exercice: donner l ensemble des racine carre arrondi(2 chiffres apres la virgule) des entiers qui sont:

- entre 0 et 10000
- sont divisible par 3, 5, 7, et 11

[3]:
```
# (Avec comprehension)
from math import sqrt

e = {
    round(sqrt(x), 2)
    for x in range(10001)
    if x % 3 == 0
    if x % 5 == 0
    if x % 7 == 0
    if x % 11 == 0
}
# Ou bien
e = {
    round(sqrt(x), 2)
    for x in range(10001)
    if (x % 3 == 0 and x % 5 == 0 and x % 7 == 0 and x % 11 == 0)
}
# Ou bien
e = {round(sqrt(x), 2) for x in range(10001) if (x % (3 * 5 * 7 * 11) == 0)}
e
```

[3]:
```
{0.0, 33.99, 48.06, 58.86, 67.97, 75.99, 83.25, 89.92, 96.12}
```

[4]:
```
# Ou bien (sans comprehension)
ee = set()
for x in range(10001):
    if x % 3 == 0 and x % 5 == 0 and x % 7 == 0 and x % 11 == 0:
        ee.add(round(sqrt(x), 2))
ee
```

[4]:
```
{0.0, 33.99, 48.06, 58.86, 67.97, 75.99, 83.25, 89.92, 96.12}
```

[5]:
```
t = tuple(l)
t
```

[5]:
```
(1, 2, 3, 4, 5)
```

Exercice: creer ce dictionnaire: d={'etud_0':0**2,....,'etud_i':i**2,'etud_10':10**2 }

[6]:
```
d = {"etud_{}".format(i): i**2 for i in range(11)}
# Ou bien
d = {"etud_" + str(i): i**2 for i in range(11)}
# Ou bien
d = {f"etud_{i}": i**2 for i in range(11)}
d
```

[6]:
```
{'etud_0': 0,
 'etud_1': 1,
 'etud_2': 4,
 'etud_3': 9,
 'etud_4': 16,
 'etud_5': 25,
 'etud_6': 36,
 'etud_7': 49,
 'etud_8': 64,
 'etud_9': 81,
 'etud_10': 100}
```

[ ]:
```
# Ou bien (sans comprehension)
dd = {}
for x in range(11):
    cle = f"etud_{x}"
    # cle='etud_'+str(x)
    dd[cle] = x**2
```

Exercice: A partir de liste de mots L, donner la liste mot en majuscule

[8]:
```
l = ["sa7ar", "achraf", "ala", "lakhal"]
L = [esm.upper() for esm in l]
L
```

[8]:
```
['SA7AR', 'ACHRAF', 'ALA', 'LAKHAL']
```

Exercice: donner le dictionnaire ds: s=‘bonjoue tout le monde’ ===> ds={0:‘b’,1:‘o’,……}

[9]:
```
s = "bonjoue tout le monde"
# 1ere methode (avec comprehension)
ds = {i: s[i] for i in range(len(s))}
# 2eme methode (sans comprehension)
ds = {}
for i in range(len(s)):
    d[i] = s[i]
# x=int(input('donner un entier'))
```

- Exercice: dictionnaire qui répresente la fréquence de chaque lettres voyelles
  s=‘bonjoue tout le monde’ ===> freq_voy={‘o’:4,e:3,……}

[10]:
```
s = "bonjoue tout le monde"

freq_voy = {car: s.count(car) for car in set(s) if car in "uioeya"}
# Ou bien

freq_voy = {car: s.count(car) for car in "uioeya" if car in s}
```

```
freq_voy
```

[10]: {'u': 2, 'o': 4, 'e': 3}

2.0.1 Effacer un itérable

[11]:
```
l = [1, 2]
l.clear()
l
s = {1, 2}
s.clear()
s
d = {"a": 1, "b": 2}
d.clear()
d
```

[11]: {}

[12]:
```
l = list(range(10))
l
# l=[]
l.clear()
```

2.0.2 L'execution de la commande se fait du gauche vers la droite

[13]:
```
x = 1
y = 2
z = x + y
z  # z=3
d0 = {
    "a": 1,
    "b": 2,
    "c": 3,
    "a": 10,
    "a": -5,
}  # les cles doivent etre uniques. # la derniere valeur de la cle est prise en␣
 ↪compte
d0
```

[13]: {'a': -5, 'b': 2, 'c': 3}

[14]:
```
éd0 = {"a": 1, "b": 2, "c": 3}
d1 = {"c": 30, "e": 40, "f": 50}
```

```
d0.update(d1)
d0
```

[14]: {'a': -5, 'b': 2, 'c': 30, 'e': 40, 'f': 50}

[15]:
```
# sum(I) prend en parametre un iterable I
sum((1, 2)) # --> 3
sum([1, 2]) # --> 3
# sum({1,2}) # --> 3
sum({1: "a", 2.5: "s"}) # --> 3.5
```

[15]: 3.5