Langage Python

Safwan Chendeb
safwan.chendeb@gmail.com
Janvier 2020

tuple

Comme les listes (slicing, len, in, …) avec une différence importante:

Les tuples sont immuables ! ! !

```
In [4]: t=(1,2)

In [5]: type(t)
Out[5]: tuple

In [6]: t=(1,2,"text", True)

In [7]: type(t)
Out[7]: tuple

In [8]: t=(1)

In [9]: type(t)
Out[9]: int

In [10]: t=(1,) # attention il faut ajouter , en cas de singleton
```

```
In [18]: t=(1,2,"text", True)

In [19]: t[1]=10
Traceback (most recent call last):

  File "<ipython-input-19-8baebb1fb83f>", line 1, in <module>
    t[1]=10

TypeError: 'tuple' object does not support item assignment


In [20]:

In [20]: l=list(t)

In [21]: l[1]=10

In [22]: t=tuple(l)

In [23]: t
Out[23]: (1, 10, 'text', True)
```

Tuple unpacking

```
In [29]: (mini, maxi) =[1,100]

In [30]: mini
Out[30]: 1

In [31]: maxi
Out[31]: 100

In [32]: mini, maxi =1,100

In [33]: l=list(range(10))

In [34]: x,*y=l #extended tuple unpacking

In [35]: x
Out[35]: 0

In [36]: y
Out[36]: [1, 2, 3, 4, 5, 6, 7, 8, 9]

In [37]:
```

Dictionnaire

Un dictionnaire est une liste modifiable d'éléments hétérogènes indicés par des clés permettant de stocker des couples clé : valeur, avec un accès très rapide à la valeur à partir de la clé. La clé ne peut être présente qu'une seule fois dans le tableau.

```
In [51]: mondicovide=dict()

In [52]: age={}

In [53]: age={'lea':20,'marc':22,'luc':21}

In [54]: age['lea']
Out[54]: 20
```

```
In [44]: age={'lea':20,'marc':22,'luc':21}

In [45]: age['lea']
Out[45]: 20

In [46]: cal=[('jan',31), ('fev',28), ('mars',31)]

In [47]: dicCal=dict(cal)

In [48]: dicCal['fev']
Out[48]: 28

In [49]: del dicCal['fev']

In [50]: dicCal
Out[50]: {'jan': 31, 'mars': 31}
```

Dictionnaire

```
In [58]: dcal={'jan':31, 'fev':28, 'mars':31}

In [59]: 'jan' in dcal
Out[59]: True

In [60]: 'jun' not in dcal
Out[60]: True

In [61]: dcal.keys()
Out[61]: dict_keys(['jan', 'fev', 'mars'])

In [62]: dcal.values()
Out[62]: dict_values([31, 28, 31])

In [63]: dcal.items()
Out[63]: dict_items([('jan', 31), ('fev', 28), ('ma
```

```
In [70]: dcal={'jan':31, 'fev':28, 'mars':31}

In [71]: dcal.items()
Out[71]: dict_items([('jan', 31), ('fev', 28), ('mars', 31)])

In [72]: l=dcal.items()

In [73]: type(l)
Out[73]: dict_items

In [74]: dcal['avr']=30

In [75]: l
Out[75]: dict_items([('jan', 31), ('fev', 28), ('mars', 31), ('avr', 30)])

In [76]: for mois,nbjour in dcal.items():
    ...:     print(f"{mois} {nbjour}", end=' / ')
    ...:
jan 31 / fev 28 / mars 31 / avr 30 /
```

Dictionnaire et table de hash (clé immuable)

```
In [90]: l1 =range(10000)

In [91]: l2=range(10000)

In [92]: l=zip(l1,l2)

In [93]: dico=dict(l)

In [94]: %timeit 'x' in l1
509 µs ± 5.61 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

In [95]: %timeit 'x' in dico
65.1 ns ± 1.29 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops each)
```

Les classes

```
 9 class Point:
10     """
11     une classe point
12     """
13     def __init__(self):
14         print("point instancié")
15
```

```
 9 class Point:
10     """
11     une classe point
12     """
13     def __init__(self,x,y):
14         print("point instancié")
15         self.x=x
16         self.y=y
17     def module(self):
18         return (self.x**2+self.y**2)**0.5
19     def __str__ (self):
20         return str(self.x) + ' ' + str(self.y)
```

Héritage (simple ou multiple)

```
23 class PointTempo(Point):
24     def __init__(self, x,y,t):
25         Point.__init__(self, x,y)
26         self.t=t
```

```
In [123]: p2=PointTempo(2,6,1)
point instancié

In [124]: p2.module()
Out[124]: 6.324555320336759

In [125]: print(p2)
2 6

In [126]: p2.x
Out[126]: 2

In [127]: p2.t
Out[127]: 1
```

```
In [131]: class A:
     ...:     pass
     ...: class B(A):
     ...:     pass
     ...: class C(A):
     ...:     pass
     ...: class D(B,C): #heritage multiple (possible en python)
     ...:     pass
     ...:

In [132]: D.mro() #methode resolution order
Out[132]: [__main__.D, __main__.B, __main__.C, __main__.A, object]
```