Dictionnaire

Declaration :

```
> user = { "name" : "Salhi",
           "age" : 19,
           "sexe" : M
           "skills" : ["Html", "Css"]
         }

> user = dict()
> user = {}
```

→ dic key need to be immutable (list not allowed)
→ dic value : all types
→ key must be unique.

Access :

```
> user ["name"]
       Salhi

> user["name"][0]
       Html.
```

Fonctions :

```
> len (d)      # nbre elements
> min (d)      # plus petit clé
> max (d)      # plus grand clé
> del d[i]     # supprimer element associé à clé I
> list (d)     # retourner list des clés
```

Dictionnaire Methodes

get():

```
> d.get (key , value)
```

key → required, the key of item you want to return

value → option, Value returned si la clé ∄
default value = none

```
> car = { "Brand" : "Ford",
          "model" : "Mustang",
          "year" : 1964
        }

> x = car.get("model")
> print (x)
        Mustang
```

items():

- No parameters
- return view object, list of key-value pairs as Tuple.

```
> d.items()

> car = {"brand" : "ford", "model" : "Mustang", "year" : 1964}
> x = car.items()
      dict_items([('brand', 'Ford'), ('model', 'Mustang'), (
> list(x)[0]      # return Tuple.
  ('brand', 'Ford')
```

Keys():

→ no parameter.
→ return view objet, Keys as a list

```
> d.keys()
    dict_keys (['brand', 'model', 'color'])

> list(d.keys())[0]
    brand
```

values():

→ no parameters
→ return view objet. Values as a list

```
> d.values()
    dict_values (['Ford', 'mustang', 1964])

> list(d.values())[0]
    Ford
```

clear()

→ no parameter.
→ remove all items form à dict

```
> d.clear()
    none
```

Comparaison deux dictionnaires:

→ utuliser is et not is si, n'est =/=

update():

>d.update(iterable)

→ insert specified items to the dictionary
→ parameter : iterable : a dictionnary or iterable object with key-value pair

key value

> car.update({"color":"white"})

copy():

> d.copy()
→ return a shallow copy of a specified dictionary.
→ no parameter.

> car.copy()

setdefault():

> d.setdefault(keyname, value)

→ return the value of the item with the specified key.
if key ∄, insert the key with the value

keyname : required, the key of item
value : optional. if key exist this parameter no effect
default value : none.

ajouter clé si ∄

> car.setdefault("color","white")
white

fromkeys():

> d.fromkeys(keys, value)

- return a dictionary with the specified keys and specified value.
- keys: iterable for new dict, required.
  value: optional, the value for all keys
  default value: none.

```
> x = ('key1', 'key2', 'key3')
> d.fromkeys(x)
    {'key1': none, 'key2': none}
```

pop():

> d.pop(keyname, default value)

- remove specified item from dict.
  return de value to remove.
- keyname: required, the key to value removed
- defaultvalue: optional, A value to return if key ∄
  if key ∄ et this parameter no specified
  return error.

Popitem():

> d.popitem()

- remove the last inserted item
  return key-value as Tuple.
- no parameters