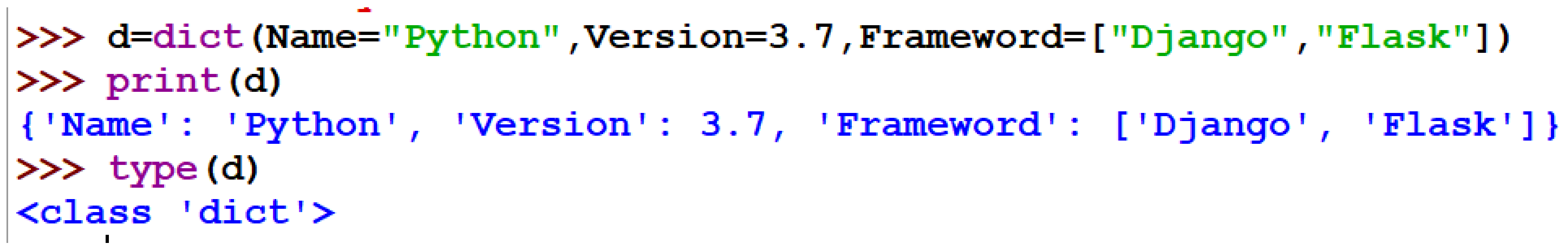
Dictionary Functions

Dict constructor

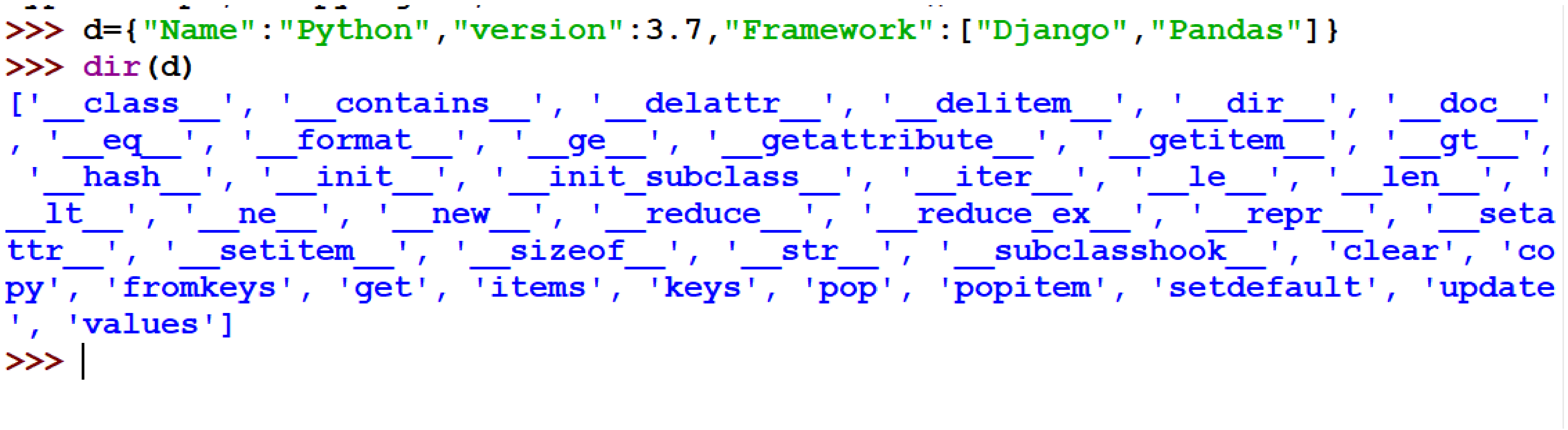
- keywords are not string literals
- the use of equals rather than colon for the assignment



```
>>> d=dict(Name="Python",Version=3.7,Frameword=["Django","Flask"])
>>> print(d)
{'Name': 'Python', 'Version': 3.7, 'Frameword': ['Django', 'Flask']}
>>> type(d)
<class 'dict'>
```

Dictionary

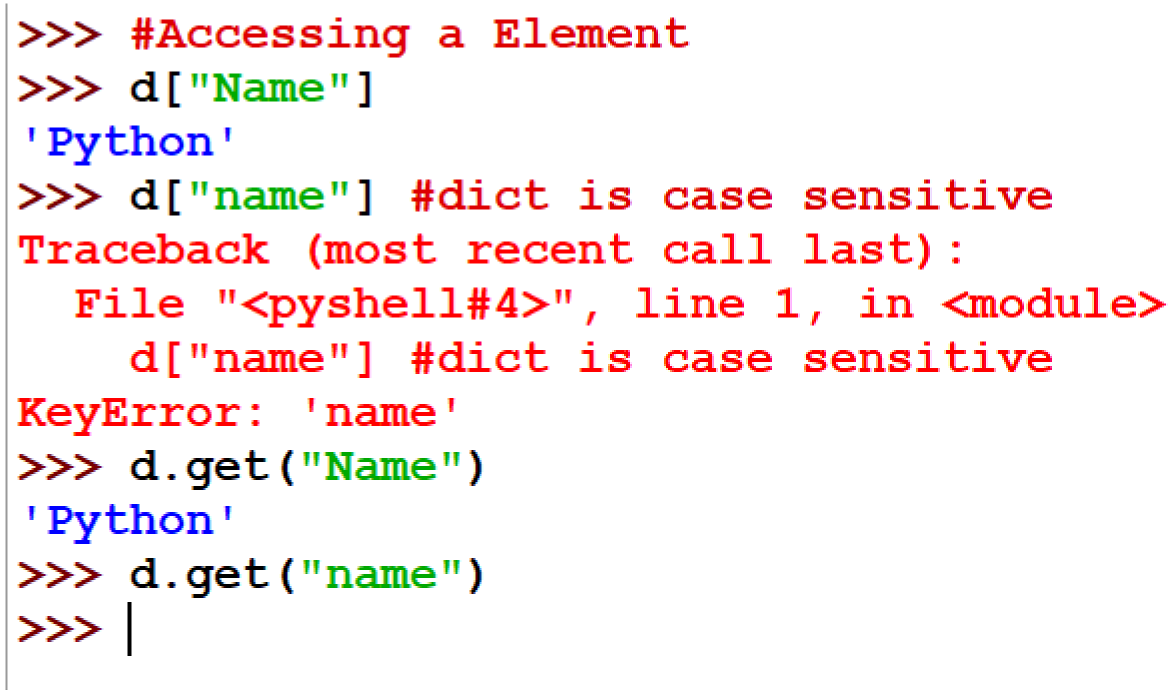
A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.



```
>>> d={"Name":"Python","version":3.7,"Framework":["Django","Pandas"]}
>>> dir(d)
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__'
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__',
 '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '
__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__seta
ttr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'co
py', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update
', 'values']
>>> |
```

Accessing Items

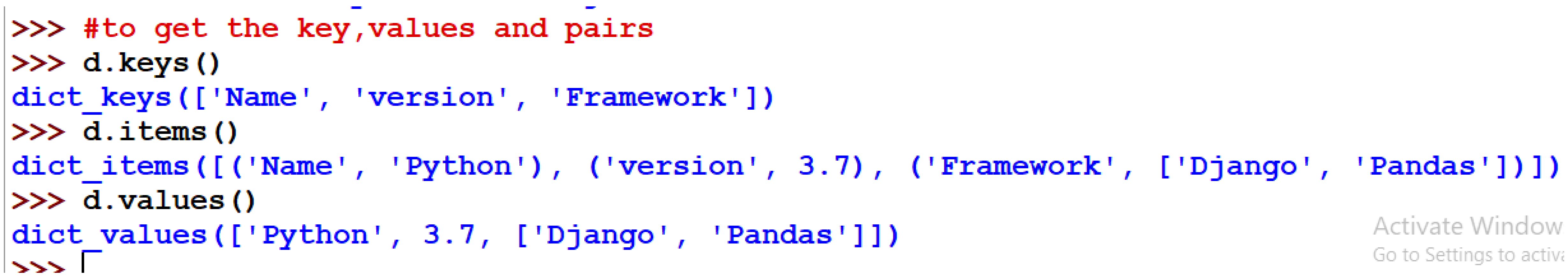
- We can access the dictionary by normal indexing method and get method
- Normal indexing method will give the error if there is no key name, get method will return the null value instead of NULL.



```
>>> #Accessing a Element
>>> d["Name"]
'Python'
>>> d["name"] #dict is case sensitive
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    d["name"] #dict is case sensitive
KeyError: 'name'
>>> d.get("Name")
'Python'
>>> d.get("name")
>>> 
```

To get dict properties

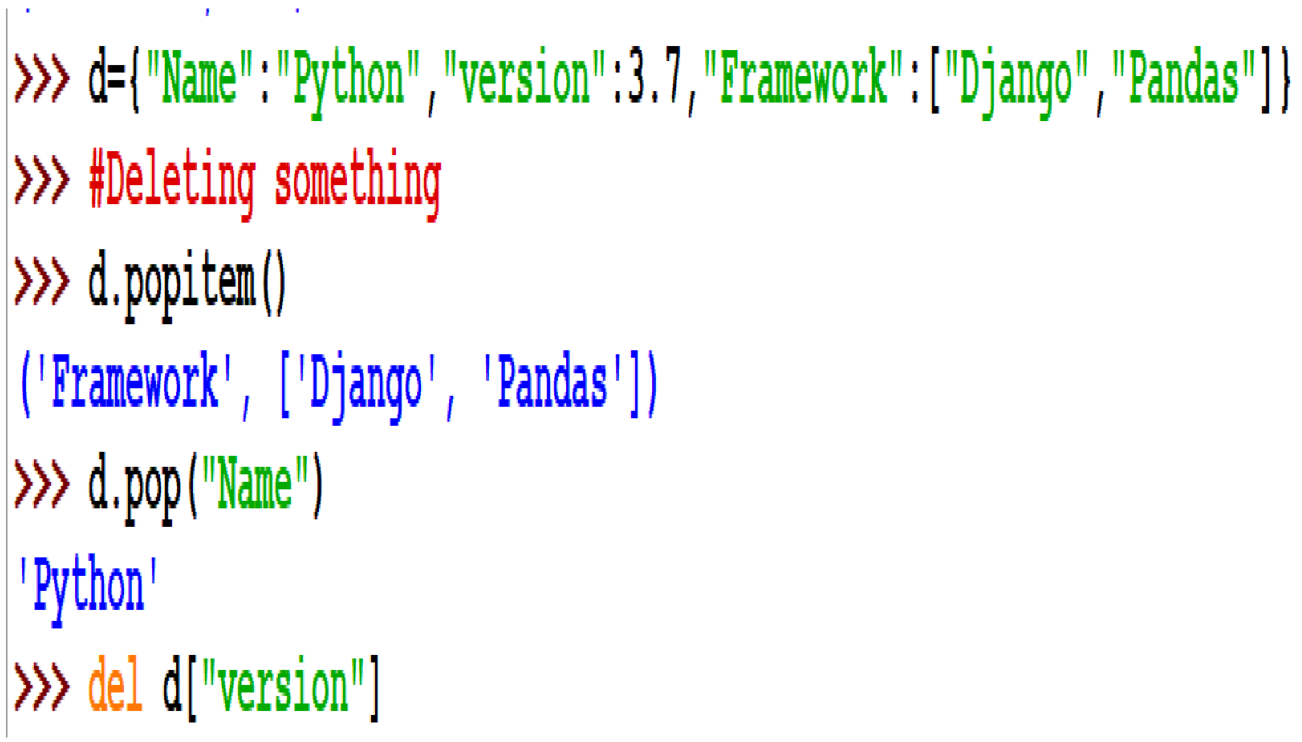
- We can able to get all the properties like only key name, or only key values or iteration of key name and key value



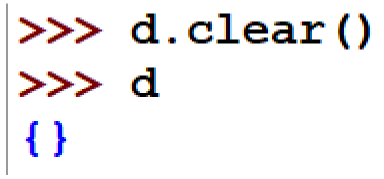
```
>>> #to get the key,values and pairs
>>> d.keys()
dict_keys(['Name', 'version', 'Framework'])
>>> d.items()
dict_items([('Name', 'Python'), ('version', 3.7), ('Framework', ['Django', 'Pandas'])])
>>> d.values()
dict_values(['Python', 3.7, ['Django', 'Pandas']])
>>>
```

Deleting the Elements

- popitem → Remove the last element, Previous it was removing the random element
- pop(keyname) → it remove only keyname after some sec the keyvalue auto destroy
- del d[keyname]→ Normal python keyword method
- Clear → gives the empty dictionary



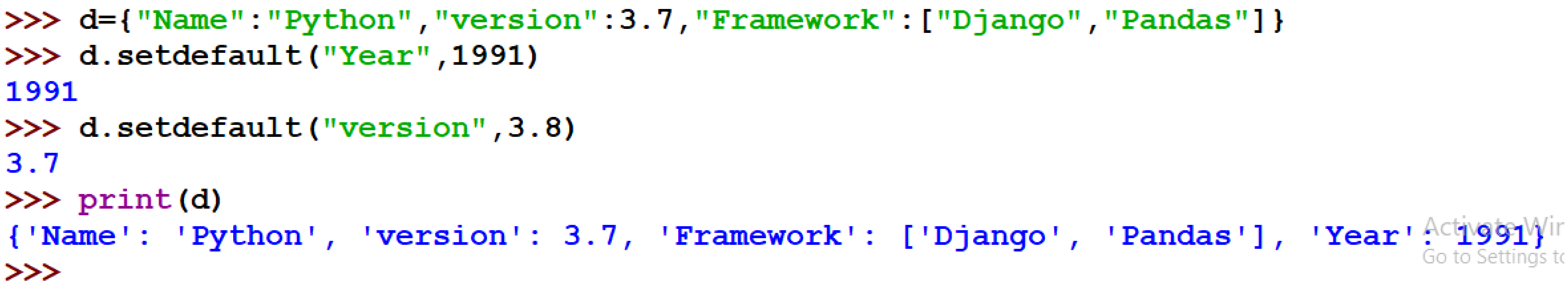
```
>>> d={"Name":"Python","version":3.7,"Framework":["Django","Pandas"]}
>>> #Deleting something
>>> d.popitem()
('Framework', ['Django', 'Pandas'])
>>> d.pop("Name")
'Python'
>>> del d["version"]
```



```
>>> d.clear()
>>> d
{}
```

setdefault()

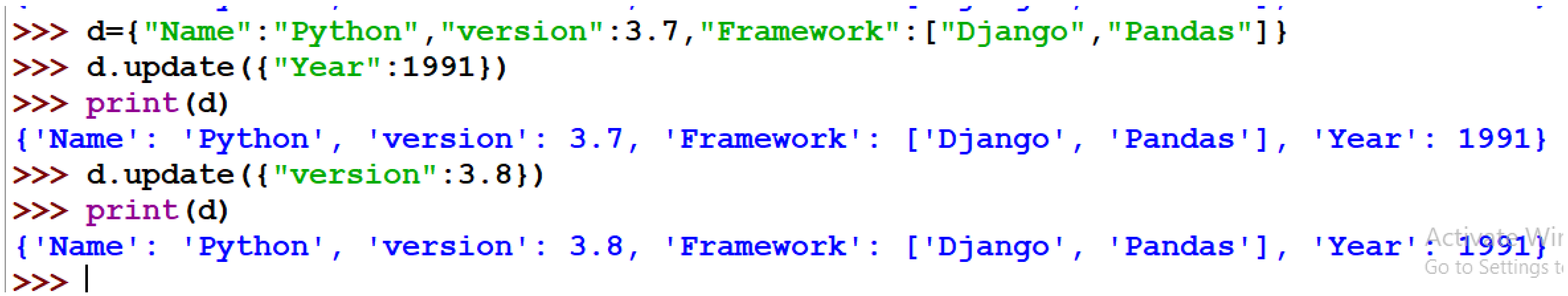
- It set some default values to the dict key, it wont update the values but if the key is not there it insert the key
- In first Example the year is not there it added it, second the version is there it didn’t update the version



```
>>> d={"Name":"Python","version":3.7,"Framework":["Django","Pandas"]}
>>> d.setdefault("Year",1991)
1991
>>> d.setdefault("version",3.8)
3.7
>>> print(d)
{'Name': 'Python', 'version': 3.7, 'Framework': ['Django', 'Pandas'], 'Year': 1991}
>>>
```

update()

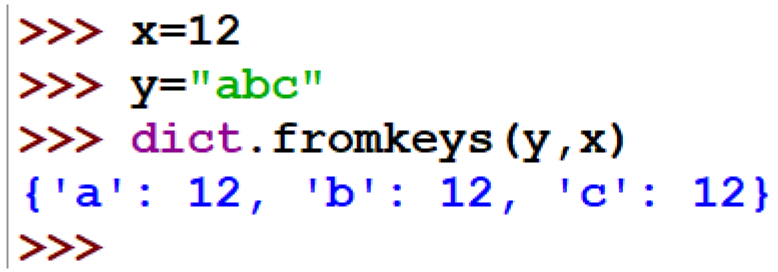
- This is used to add and update the key
- In this example year is not there it added the new keyname,version is there it update the value from 3.7 to 3.8



```
>>> d={"Name":"Python","version":3.7,"Framework":["Django","Pandas"]}
>>> d.update({"Year":1991})
>>> print(d)
{'Name': 'Python', 'version': 3.7, 'Framework': ['Django', 'Pandas'], 'Year': 1991}
>>> d.update({"version":3.8})
>>> print(d)
{'Name': 'Python', 'version': 3.8, 'Framework': ['Django', 'Pandas'], 'Year': 1991}
>>> |
```

From keys

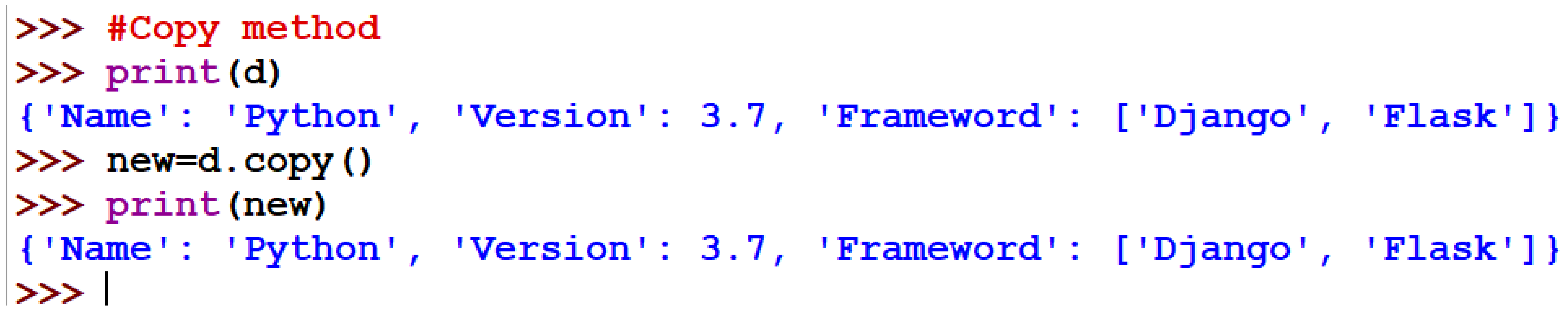
- The fromkeys() method returns a dictionary with the specified keys and values.
- First parameter is itratable we need to give in sequence for keyname



```
>>> x=12
>>> y="abc"
>>> dict.fromkeys(y,x)
{'a': 12, 'b': 12, 'c': 12}
>>>
```

copy()

- Copying a dictionary



```
>>> #Copy method
>>> print(d)
{'Name': 'Python', 'Version': 3.7, 'Frameword': ['Django', 'Flask']}
>>> new=d.copy()
>>> print(new)
{'Name': 'Python', 'Version': 3.7, 'Frameword': ['Django', 'Flask']}
>>> |
```