IOTA ACADEMY

Become Job Ready...

Dictionary

- Python dictionary is an unordered collection of items. This means we *can't index* it.
- While other compound data types have only value as an element, a dictionary has a **key: value pair**.
- Efficient data structure to store data. by means of key:value pair
- Dictionaries are like hashtables. Don't worry if you don't know what hashtable is.

Dict Creation

In [21]:
```
# empty dictionary
my_dict = {}
print(type(my_dict))

# dictionary with integer keys
my_dict = {1: 'abc', 2: 'xyz'} #key:value
print(my_dict)

# dictionary with mixed keys
my_dict = {'name': 'iota', 1: ['abc', 'xyz']}
print(my_dict)


# create empty dictionary using dict()
my_dict = dict()

my_dict = dict([(1, 'abc'), (2, 'xyz')])     # create a dict with list of tuples
print(my_dict)
```

```
<class 'dict'>
{1: 'abc', 2: 'xyz'}
{'name': 'iota', 1: ['abc', 'xyz']}
{1: 'abc', 2: 'xyz'}
```

Dict Access

In [22]:
```
my_dict = {'name': 'iota', 'age': 26, 'address': 'indore','courses':{'data analytics': 'DA'}}

# get name
print(my_dict['name'])

# get age
print(my_dict['age'])

print(my_dict['courses']['data analytics'])
```

```
iota
26
DA
```

In [23]:
```
# nested dictionary
print(my_dict['courses']['data analytics'])
```

```
DA
```

In [24]:
```
# if key is not present it gives KeyError
print(my_dict['degree'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[24], line 2
      1 # if key is not present it gives KeyError
----> 2 print(my_dict['degree'])

KeyError: 'degree'
```

get()

To avoid KeyError if key is not present.

Return the value for key if key is in the dictionary, else default.

Syntax: `dict.get(key, default=None)`

In [25]:
```
print(my_dict.get('name1')) # it will return default value if key is not present here default is none
```

```
None
```

In [26]:
```
print(my_dict.get('name1', 'i am not present'))    # here default is 'i am not present'
```

```
i am not present
```

In [27]:
```
# example
a = {1:{
    1:{'name':'hemant'}
       },
     2:'iota'}
a
```

Out[27]:
```
{1: {1: {'name': 'hemant'}}, 2: 'iota'}
```

In [28]:
```
print(a[1][1]['name'])
```

```
hemant
```

In [29]:
```
# by get method
print(a.get(1).get(1).get('name',"I am not in the dictionary"))
```

```
hemant
```

Dict Add or Modify Elements

In [30]:
```
# We can add elements to a dictionary using the name of the dictionary with [].

my_dict = {'name': 'hemant', 'age': 26, 'address': 'indore'}
my_dict["state"] = "MP"    # add new key
print(my_dict)
```

```
{'name': 'hemant', 'age': 26, 'address': 'indore', 'state': 'MP'}
```

In [31]:
```
# We can also use [] to change the value associated with a particular key.

my_dict = {'name': 'hemant', 'age': 26, 'address': 'indore'}

# update name
my_dict['name'] = 'iota'

print(my_dict)
```

```
{'name': 'iota', 'age': 26, 'address': 'indore'}
```

In [32]:
```
# example

dict1 = {'name': 'Hemant', 'subject': 'Python', 'city':'Indore' }

print(dict1['subject'])

dict1['subject'] = 'SQL'
print(dict1['subject'])
print(dict1)

dict1['Education'] = {'School':'JNV', 'College':'IITB'}
print(dict1)

print(dict1['Education']['College'])
```

```
Python
SQL
{'name': 'Hemant', 'subject': 'SQL', 'city': 'Indore'}
{'name': 'Hemant', 'subject': 'SQL', 'city': 'Indore', 'Education': {'School': 'JNV', 'College': 'IITB'}}
IITB
```

In [34]:
```
# taking inputs and saving them in a dictionary

variable_list = ['Name','Age','Gender','Mobile']

dict1 = {}
for i in variable_list:
    dict1[i]=input(f"Enter your {i}")

print(dict1)
```

```
{'Name': 'ankit', 'Age': '25', 'Gender': 'male', 'Mobile': '2326356255632'}
```

Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

In [35]:
```
# print all dictionary methods using dir function

print(dir(dict))
```

```
['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq_
_', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__', '__init_subcla
ss__', '__ior__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__reduce__', '__re
duce_ex__', '__repr__', '__reversed__', '__ror__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__su
bclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update',
'values']
```

clear()

removes all elements from a dictionary

In [36]:
```
# create a dictionary
my_dict = {'name': 'Hemant', 'age': 27, 'address': 'Indore'}

# clear
my_dict.clear()
print(my_dict)
```

```
{}
```

copy()

Returns a copy of the dictionary

In [37]:
```
# create a dictionary
my_dict = {'name': 'Hemant', 'age': 27, 'address': 'Indore'}

# copy
new_dict = my_dict.copy()
print(new_dict)
```

```
{'name': 'Hemant', 'age': 27, 'address': 'Indore'}
```

fromkeys()

Returns a dictionary with the specified keys and value

In [38]:
```
# fromkeys[seq[, v]] -> Return a new dictionary with keys from seq and value equal to v (defaults to None).

subjects = {}.fromkeys(['Math', 'English', 'Hindi'], 0) # (iterable, value)
print(subjects)
```

```
{'Math': 0, 'English': 0, 'Hindi': 0}
```

get()

Returns the value of the specified key

In [39]:
```
# create a dictionary
my_dict = {'name': 'Hemant', 'age': 27, 'address': 'Indore'}

# get, if key not present then doesn't raise error
print(my_dict.get('age'))

print(my_dict.get("degree")) # no message then it returns None

print(my_dict.get("degree","Custom Message here"))
```

```
27
None
Custom Message here
```

items()

Returns a list containing a tuple for each key value pair

In [40]:
```
# items
subjects = {'maths':67, 'english':91, 'hindi':96, 'science':75}
print(subjects.items())
```

```
dict_items([('maths', 67), ('english', 91), ('hindi', 96), ('science', 75)])
```

keys()

Returns a list containing the dictionary's keys

In [41]:
```
# keys
subjects = {'maths':67, 'english':91, 'hindi':96, 'science':75}
print(subjects.keys())
```

```
dict_keys(['maths', 'english', 'hindi', 'science'])
```

values()

Returns a list of all the values in the dictionary

In [42]:
```
# values
subjects = {'maths':67, 'english':91, 'hindi':96, 'science':75}
print(subjects.values())
```

```
dict_values([67, 91, 96, 75])
```

pop()

Removes the element with the specified key

In [43]:
```
# create a dictionary
my_dict = {'name': 'Hemant', 'age': 27, 'address': 'Indore'}

# pop: remove a particular item(key-value pair) based on key
print(my_dict.pop('age'))

print(my_dict)
```

```
27
{'name': 'Hemant', 'address': 'Indore'}
```

popitem()

Removes the last inserted key-value pair

In [44]:
```
# create a dictionary
my_dict = {'name': 'Hemant', 'age': 27, 'address': 'Indore'}

# popitem
x = my_dict.popitem()

print(x)
```

```
('address', 'Indore')
```

setdefault()

Returns the value of the specified key. If the key does not exist: insert the key, with the specified value

In [45]:
```
# create a dictionary
coaching = {
  "name": "IOTA Academy",
  "year": 2022,
  "location": "India"
}

#setdefault
x = coaching.setdefault("course", "Python") # when key is not present
print(x)
print(coaching)

y = coaching.setdefault("name","Data Analytics")  # when key is present
print(y)
print(coaching)
```

```
Python
{'name': 'IOTA Academy', 'year': 2022, 'location': 'India', 'course': 'Python'}
IOTA Academy
{'name': 'IOTA Academy', 'year': 2022, 'location': 'India', 'course': 'Python'}
```

update()

Updates the dictionary with the specified key-value pairs

In [46]:
```
# create a dictionary
coaching = {
  "name": "IOTA Academy",
  "year": 2022,
  "location": "India"
}

# update
coaching.update({"course": "Python"}) # when key is not present
print(coaching)

coaching.update({"year": 2023,'degree':'Masters'}) # when key is present
print(coaching)
```

```
{'name': 'IOTA Academy', 'year': 2022, 'location': 'India', 'course': 'Python'}
{'name': 'IOTA Academy', 'year': 2023, 'location': 'India', 'course': 'Python', 'degree': 'Masters'}
```

Dict Delete or Remove Element

In [47]:
```
squares = {2: 4, 3: 9, 4: 16, 5: 25}

# delete particular key
del squares[2]

print(squares)
```

```
{3: 9, 4: 16, 5: 25}
```

In [48]:
```
squares = {2: 4, 3: 9, 4: 16, 5: 25}

# delete dictionary itself
del squares

print(squares) # NameError because dict is deleted
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[48], line 6
      3 # delete dictionary itself
      4 del squares
----> 6 print(squares)

NameError: name 'squares' is not defined
```

Using Dictionary Methods

In [49]:
```
# for loop (iterate over key-value pair (as a tuple))
subjects = {'maths':67, 'english':91, 'hindi':96, 'science':75}
print(subjects, end='\n\n')

for a,b in subjects.items():
    print("key is: ",a)
    print("value is: ",b)
```

```
{'maths': 67, 'english': 91, 'hindi': 96, 'science': 75}

key is:  maths
value is:  67
key is:  english
value is:  91
key is:  hindi
value is:  96
key is:  science
value is:  75
```

In [50]:
```
# iterate over only keys
for i in subjects.keys():
    print("marks of",i,"is")
```

```
marks of maths is
marks of english is
marks of hindi is
marks of science is
```

Dict Comprehension (will cover later)

In [51]:
```
# Dict comprehensions are just like list comprehensions but for dictionaries

# normal way
d = {'a': 1, 'b': 2, 'c': 3,'d': 4}
d1={}
for pair in d.items():
    if pair[1]>2:
        d1[pair[0]]=pair[1]
print(d1)
```

```
{'c': 3, 'd': 4}
```

In [52]:
```
# Creating a new dictionary with only pairs where the value is larger than 2

# dictionary comprehension
d = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
new_dict = {k:v for k, v in d.items() if v > 2}
print(new_dict)
```

```
{'c': 3, 'd': 4}
```

In [53]:
```
# We can also perform operations on the key value pairs
d = {'a':1,'b':2,'c':3,'d':4,'e':5}
d = {k + 'c':v * 2 for k, v in d.items() if v > 2}  # concise but confusing for some
print(d)
```

```
{'cc': 6, 'dc': 8, 'ec': 10}
```

That's Great!