Untitled1

April 3, 2023

[1]:
```
d={}
```

[4]:
```
type(d)
```

[4]:
```
dict
```

[5]:
```
## dictionary stores a data in terms of keys and values
```

[8]:
```
d1={"name": "punith","email":"lokeshaarmy@gmail","number":8978589}
```

[9]:
```
d1
```

[9]:
```
{'name': 'punith', 'email': 'lokeshaarmy@gmail', 'number': 8978589}
```

[11]:
```
type(d1)
```

[11]:
```
dict
```

[12]:
```
d2={"name":"punith","name":"Punith"}
```

[13]:
```
d2
```

[13]:
```
{'name': 'Punith'}
```

[14]:
```
## y second is choosen rather than first one
## the reason is the dictionary follows uniqueness and always try give op for␣
 ↪the latest one
```

[15]:
```
d3={245645:"abc"}
```

[16]:
```
d3
```

[16]:
```
{245645: 'abc'}
```

[17]:
```
## key can be an integer
```

[18]:
```
d4={2345.26:"abc"}
```

[19]:
```
d4
```

[19]:
```
{2345.26: 'abc'}
```

[20]:
```
## key can be float
```

[22]:
```
d5={True:"abc"}
```

[23]:
```
d5
```

[23]:
```
{True: 'abc'}
```

[24]:
```
## keys can be bool
```

[25]:
```
d6={@:"abc"}
```

```
  Cell In[25], line 1
    d6={@:"abc"}
        ^
SyntaxError: invalid syntax
```

[26]:
```
## keys do not support special characters like @ # $
```

[27]:
```
d8={[1,2,3,4,5,6]:"abc"}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 1
----> 1 d8={[1,2,3,4,5,6]:"abc"}

TypeError: unhashable type: 'list'
```

[28]:
```
## keys cannot be list
```

[29]:
```
d9={(1,2,3,45):"abc"}
```

[30]:
```
d9
```

[30]:
```
{(1, 2, 3, 45): 'abc'}
```

[31]:
```
## keys can be tuples
```

[32]:
```
d10={{1,2,3,4,5}:"abc"}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[32], line 1
----> 1 d10={{1,2,3,4,5}:"abc"}

TypeError: unhashable type: 'set'
```

[33]:
```
## sets cannot be considered as keys
```

[35]:
```
d11={{"key":1234}:"abc"}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 d11={{"key":1234}:"abc"}

TypeError: unhashable type: 'dict'
```

[36]:
```
## conclusion
## only TUPLES can be considered as keys
```

[38]:
```
d12={"couse_name":{"data science masters","web dev","java with dsa"}}
```

[39]:
```
d12
```

[39]:
```
{'couse_name': {'data science masters', 'java with dsa', 'web dev'}}
```

[40]:
```
## one key with multiple values
```

[41]:
```
d13={"key":(1,2,3,4,5,6)}
```

[42]:
```
d13
```

[42]:
```
{'key': (1, 2, 3, 4, 5, 6)}
```

[43]:
```
### values can be tuples
```

[44]:
```
d14={"key":{1,2,3,4,5}}
```

[45]:
```
## values can be sets
```

[46]:
```
d14={"key":[1,2,3,45,6]}
```

[47]:
```
d14
```

[47]:
```
{'key': [1, 2, 3, 45, 6]}
```

[48]:
```
## values can be list
```

[49]:
```
d15={"key":{"name":"punith","class":"DSM"}}
```

[50]:
```
d15
```

[50]:
```
{'key': {'name': 'punith', 'class': 'DSM'}}
```

[52]:
```
## values can be dictionary
## dictionary inside the dictionary is called nested dictionary
```

[89]:
```
d16={"batch_name":["data science masters","web␣
 ↪developement","JDS"],"start_date":(28,14,21),"mentor_name":
 ↪("krish","sudanshu")}
```

[90]:
```
d16
```

[90]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu')}
```

[91]:
```
d16["timings"]=(8,9,7)
d16["place"]=("HASSAN","BENGALURU","MYSURU")
d16["name"]=("punith")
```

[92]:
```
d16
```

[92]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7),
 'place': ('HASSAN', 'BENGALURU', 'MYSURU'),
 'name': 'punith'}
```

[93]:
```
## addind new key to the existing dictionary
```

[94]:
```
d16["place"]
```

[94]:
```
('HASSAN', 'BENGALURU', 'MYSURU')
```

[95]:
```
d16["batch_name"]
```

[95]:
```
['data science masters', 'web developement', 'JDS']
```

[96]:
```
d16["name"]
```

[96]:
```
'punith'
```

[97]:
```
## extracting the values by using the keys
```

[98]:
```
d16
```

[98]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7),
 'place': ('HASSAN', 'BENGALURU', 'MYSURU'),
 'name': 'punith'}
```

[99]:
```
d16["name"].upper()
```

[99]:
```
'PUNITH'
```

[101]:
```
d15
```

[101]:
```
{'key': {'name': 'punith', 'class': 'DSM'}}
```

[102]:
```
d15["key"]["class"]
```

[102]:
```
'DSM'
```

[104]:
```
## to obtain any particular value in dictionary
## go one by one
```

[105]:
```
d15["key1"]="abc"
```

[106]:
```
d15
```

[106]:
```
{'key': {'name': 'punith', 'class': 'DSM'}, 'key1': 'abc'}
```

[107]:
```
del d15["key1"]
```

[108]:
```
d15
```

[108]:
```
{'key': {'name': 'punith', 'class': 'DSM'}}
```

[109]:
```
## the key can be added and also be deleted
```

[110]:
```
d15.clear()
```

[111]:
```
d15
```

[111]:
```
{}
```

[112]:
```
## clear command clears everything inside the dictionary
```

[114]:
```
d16
```

[114]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7),
 'place': ('HASSAN', 'BENGALURU', 'MYSURU'),
 'name': 'punith'}
```

[116]:
```
len(d16)
```

[116]:
```
6
```

[117]:
```
d16.keys()
```

[117]:
```
dict_keys(['batch_name', 'start_date', 'mentor_name', 'timings', 'place',
'name'])
```

[118]:
```
## extract  all the keys only
```

[119]:
```
d16.values()
```

[119]:
```
dict_values([['data science masters', 'web developement', 'JDS'], (28, 14, 21),
('krish', 'sudanshu'), (8, 9, 7), ('HASSAN', 'BENGALURU', 'MYSURU'), 'punith'])
```

[120]:
```
# extracts all the values
```

[130]:
```
list(d16.keys())
```

[130]:
```
['batch_name', 'start_date', 'mentor_name', 'timings', 'place', 'name']
```

[133]:
```
## it converts keys into the list
```

[134]:
```
list(d16.values())
```

[134]:
```
[['data science masters', 'web developement', 'JDS'],
 (28, 14, 21),
 ('krish', 'sudanshu'),
 (8, 9, 7),
 ('HASSAN', 'BENGALURU', 'MYSURU'),
 'punith']
```

[135]:
```
## converted only values into the list
```

[140]:
```
list(d16.items())
```

[140]:
```
[('batch_name', ['data science masters', 'web developement', 'JDS']),
 ('start_date', (28, 14, 21)),
 ('mentor_name', ('krish', 'sudanshu')),
 ('timings', (8, 9, 7)),
 ('place', ('HASSAN', 'BENGALURU', 'MYSURU')),
 ('name', 'punith')]
```

[141]:
```
## converts both values and keys into the list by using the items
```

[142]:
```
d16
```

[142]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7),
 'place': ('HASSAN', 'BENGALURU', 'MYSURU'),
 'name': 'punith'}
```

[143]:
```
d17=d16.copy()
```

[144]:
```
d17
```

[144]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7),
 'place': ('HASSAN', 'BENGALURU', 'MYSURU'),
 'name': 'punith'}
```

[155]:
```
del d16["name"]
```

[156]:
```
d16
```

[156]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7),
 'place': ('HASSAN', 'BENGALURU', 'MYSURU')}
```

[157]:
```
d17
```

[157]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7),
 'place': ('HASSAN', 'BENGALURU', 'MYSURU'),
 'name': 'punith'}
```

[167]:
```
## deeep copy;
## copying the whole data into different dictionary
## this is the replica of the d16 and repicates into new memory
## and it reserves the new space
### if any new changes is done!  it does not effect each other.
```

[ ]:
```
d18=d16
```

[160]:
```
d18
```

[160]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7),
 'place': ('HASSAN', 'BENGALURU', 'MYSURU')}
```

[161]:
```
del d16["place"]
```

[166]:
```
d18
```

[166]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7)}
```

[164]:
```
## this method is not the replica of the d16
## in this method new changes will directly effect each other
## this method is called swallow copy
```

[169]:
```
d16
```

[169]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu'),
 'timings': (8, 9, 7)}
```

[170]:
```
d16.pop("timings")
```

[170]:
```
(8, 9, 7)
```

[171]:
```
d16
```

[171]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21),
 'mentor_name': ('krish', 'sudanshu')}
```

[172]:
```
d16.pop("mentor_name")
```

[172]:
```
('krish', 'sudanshu')
```

[173]:
```
d16
```

[173]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21)}
```

[174]:
```
## POP removes the specific key and value
```

[177]:
```
d.fromkeys((1,2,3),("a","b","c"))
```

[177]:
```
{1: ('a', 'b', 'c'), 2: ('a', 'b', 'c'), 3: ('a', 'b', 'c')}
```

[178]:
```
## fromkeys used to itterate
```

[189]:
```
d19={"key1":"value1","key2":"value2"}
d20={"key3":"value3","key4":"value4"}
```

[190]:
```
(d19,d20)
```

[190]:
```
({'key1': 'value1', 'key2': 'value2'}, {'key3': 'value3', 'key4': 'value4'})
```

[181]:
```
## writing the dictionary into the single tuples
```

[191]:
```
d19.update(d20)
```

[192]:
```
d19
```

[192]:
```
{'key1': 'value1', 'key2': 'value2', 'key3': 'value3', 'key4': 'value4'}
```

[194]:
```
## another way of defining into the one single dictionary by using update
```

[195]:
```
d20.update(d19)
```

[196]:
```
d20
```

[196]:
```
{'key3': 'value3', 'key4': 'value4', 'key1': 'value1', 'key2': 'value2'}
```

[197]:
```
d20.get("punith")
```

[199]:
```
#### the name punith does not exists in the d20
```

[198]:
```
d20.get("key1")
```

[198]:
```
'value1'
```

[202]:
```
d20["key1"]
```

[202]:
```
'value1'
```

[203]:
```
## the value of the key 1 is obtained by the GET function
## the above function can also be used
```

0.1 Dictionary Comprehension

[216]:
```
{i : i**2 for i in range(1,11)}
```

[216]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

[217]:
```
list(range(1,11))
```

[217]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[218]:
```
{i:i+10 for i in range(1,11)}
```

[218]:
```
{1: 11, 2: 12, 3: 13, 4: 14, 5: 15, 6: 16, 7: 17, 8: 18, 9: 19, 10: 20}
```

[226]:
```
import math
d21={ i:math.log10(i) for i in range(1,11)}
```

[227]:
```
d16
```

[227]:
```
{'batch_name': ['data science masters', 'web developement', 'JDS'],
 'start_date': (28, 14, 21)}
```

[228]:
```
'batch_name' in d16
```

[228]:
```
True
```

[225]:
```
## to check whether the function does exists?
```

[229]:
```
d21
```

[229]:
```
{1: 0.0,
 2: 0.3010299956639812,
 3: 0.47712125471966244,
 4: 0.6020599913279624,
 5: 0.6989700043360189,
 6: 0.7781512503836436,
 7: 0.8450980400142568,
 8: 0.9030899869919435,
 9: 0.9542425094393249,
 10: 1.0}
```

[232]:
```
for i in d21.keys():
    if i% 2==0:
        print(d21[i])
```

```
0.3010299956639812
0.6020599913279624
0.7781512503836436
0.9030899869919435
1.0
```

[233]:
```
## to get the value of only the even numbers
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: