Assignment - 5

Q1. What does an empty dictionary's code look like?

Ans: An empty dictionary is represented by pair of empty curly brackets or can also be created through fuction call.

`d = {} (or)`

`d = dict()`

For example:

```
dictionary_1 = {}
print(f"dictionary 1: {dictionary_1} and it's type {type(dictionary_1)}")
print("*"*80)

dictionary_2 = dict()
print(f"dictionary 2: {dictionary_2} and it's type {type(dictionary_2)}")
```

```
dictionary 1: {} and it's type <class 'dict'>
********************************************************************************
dictionary 2: {} and it's type <class 'dict'>
```

Q2. what is the value of dictionary value with key 'foo' and the value 42 ?

Ans: The dictionary value will be `{'foo' : 42}`. let's check it with a code:

```
# dictionary
d = {'foo' : 42}
print(f"Dictionary value : {d}")
print('*'*80)
print(f"d.items()  : {d.items()},\nd.values() : {d.values()},\nd.keys()   : {d.keys()}")
```

```
Dictionary value : {'foo': 42}
********************************************************************************
d.items()  : dict_items([('foo', 42)]),
d.values() : dict_values([42]),
d.keys()   : dict_keys(['foo'])
```

Q3. What is the most significant distinction between a dictionary and a list?

Ans:

```
from prettytable import PrettyTable
```

```
#creating column list
cols = ["Serial No.", "List", "Dictionaries"]

#creating table and passing column lists
tbl = PrettyTable(cols)

#Adding rows

tbl.add_row(["1.", "Lists are created by placing all the elements between square brackets '[
tbl.add_row(["2.", "A comma separates the elements in the list.", "Elements are stored in the
tbl.add_row(["3.", "It is an ordered collection of data.", "It is an un-ordered collection of

print(tbl)
```

```
+------------+-----------------------------------------------------------------------------
| Serial No. |                                    List
+------------+-----------------------------------------------------------------------------
|     1.     | Lists are created by placing all the elements between square brackets '[
|     2.     |                 A comma separates the elements in the list.
|     3.     |                     It is an ordered collection of data.
+------------+-----------------------------------------------------------------------------
```

Q4. What happens if you try to access spam ['foo'] if spam is {'bar':100} ?

Ans: It will throws error that is `KeyError : 'foo'`

```
spam = {'bar':100}

# Spam is a dictionaries and every dictionary is in key:value format
# lets see the items that spam conatins
print(f"Items that spam contains: {spam.items()}")
print("*"*80)

# The keys that spam contains
print(f"Keys that spam contains: {spam.keys()}")
print("*"*80)

# The values these keys have of spam dictionaries
print(f"Values that keys have inside spam dictionaries: {spam.values()}")
print("*"*80)

# What happens if we called a wrong key or the key which is not present inside spam dictionar
print(f"Wrong key called : {spam['foo']}")
```

```
Items that spam contains: dict_items([('bar', 100)])
********************************************************************************
Keys that spam contains: dict_keys(['bar'])
********************************************************************************
Values that keys have inside spam dictionaries: dict_values([100])
********************************************************************************
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-4-6f6cdd34928a> in <module>()
     15
     16 # What happens if we called a wrong key or the key which is not present
```

Q5. if a dictionary is stored in spam,what is the difference between the expressions 'cat' in spam and 'cat' in spam.keys() ?

Ans: There is no difference. The operator checks whether a value exist as a key in dictionary or not

```
# we have a dictionary "spam"
# 'cat' as a key and 0 as a value
spam = {'cat': 0}

# keys in spam
print(f"Keys that spam contains: {spam.keys()}")
print("*"*80)

# A function that returns the key from the value
def getKey(val):
  for key, value in spam.items():
    if val == value:
      return key
  return "Key doesnot exist"

# print(getKey(0))
# print(list(spam.keys())[0])

if (list(spam.keys())[0]) == (getKey(0)):
  print("Expressions 'cat' in spam and 'cat' in spam.keys() are same.")
else:
  print("Expressions 'cat' in spam and 'cat' in spam.keys() are different.")
```

```
Keys that spam contains: dict_keys(['cat'])
********************************************************************************
Expressions 'cat' in spam and 'cat' in spam.keys() are same.
```

Q6. If a dictionary is stored in spam, what is the difference between the expressions 'cat' in spam and 'cat' in spam.values() ?

Ans: 'cat' in spam checks whether there is a key 'cat' is present in the dictionary or not,

while 'cat' in **spam.values()** checks whether there is a value 'cat' for one of the keys in spam.

Q7. What is a shortcut for the following code?

```
 if 'color' not in spam:
   spam['color'] = 'black';
```

Ans: spam.setdefault('color', 'black')

```
spam={}
spam.setdefault('color', 'black')
spam.items()
```

```
     dict_items([('color', 'black')])
```

Q8. How do you "pretty print" dictionary values using which module and function?

Ans: We can 'pretty print' dictionary values in two ways:

1. By using pprint() function of pprint module
   Note: pprint() function doesn't prettify nested dictionaries
2. By using dumps() function in Json and
   using dump() in yaml module

```
import pprint
import json
import yaml

employee = [
           {'Name': 'Sonu', 'Age':23, 'Residence': {'Country':'USA', 'City':'New York'}},
           {'Name': 'Monu', 'Age':44, 'Residence': {'Country':'Spain', 'City':'Madrid'}},
           {'Name': 'Anju ', 'Age':26, 'Residence': {'Country':'UK', 'City':'England'}},
           {'Name': 'Yun Lee', 'Age':30, 'Residence': {'Country':'Japan', 'City':'Osaka'}},
]


print('Printing using print() function\n', employee)
print('-'*331)
print('Printing using pprint() function\n')
pprint.pprint(employee)
print('-'*331)
jsondump = json.dumps(employee, indent=4)
print('Printing using dumps() method\n', jsondump)
print('-'*331)
yamldump = yaml.dump(employee)
print('Printing using dump() method\n', yamldump)
```

```
 {'Age': 26,
  'Name': 'Anju ',
  'Residence': {'City': 'England', 'Country': 'UK'}},
 {'Age': 30,
  'Name': 'Yun Lee',
  'Residence': {'City': 'Osaka', 'Country': 'Japan'}}]
-----------------------------------------------------------------------------------------
Printing using dumps() method
 [
    {
        "Name": "Sonu",
        "Age": 23,
        "Residence": {
            "Country": "USA",
            "City": "New York"
        }
    },
    {
        "Name": "Monu",
        "Age": 44,
        "Residence": {
            "Country": "Spain",
            "City": "Madrid"
        }
    },
    {
        "Name": "Anju ",
        "Age": 26,
        "Residence": {
            "Country": "UK",
            "City": "England"
        }
    },
    {
        "Name": "Yun Lee",
        "Age": 30,
        "Residence": {
            "Country": "Japan",
            "City": "Osaka"
        }
    }
]
-----------------------------------------------------------------------------------------
Printing using dump() method

 - Age: 23
  Name: Sonu
  Residence: {City: New York, Country: USA}
- Age: 44
  Name: Monu
  Residence: {City: Madrid, Country: Spain}
- Age: 26
  Name: 'Anju '
  Residence: {City: England, Country: UK}
- Age: 30
  Name: Yun Lee
  Residence: {City: Osaka, Country: Japan}
```

✓ 0s completed at 3:55 PM