day-5

October 4, 2024

1 Dictionary

#Rules #Dictionary has no indexing. #Dictionary is mutable type #key->immutable, values->mutable #keys should be unique

#Mutable -> List,Sets,Dictionary #Immutable -> tuple, string,int,float,compex,boolean

[1]:
```
D1={"name":'safdar',"Gender":'Male'}
D1
```

[1]:
```
{'name': 'safdar', 'Gender': 'Male'}
```

[2]:
```
type(D1)
```

[2]:
```
dict
```

[3]:
```
D={}
D
```

[3]:
```
{}
```

[4]:
```
D1={"name":'safdar',"Gender":'Male',"name":'khan'}
D1
```

[4]:
```
{'name': 'khan', 'Gender': 'Male'}
```

[5]:
```
s=10
s=19
s
```

[5]:
```
19
```

[6]:
```
D2={[1,2,3]:'safdar'}
D2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\PROGRA~1\KMSpico\temp/ipykernel_9564/3604071048.py in <module>
----> 1 D2={[1,2,3]:'safdar'}
```

```
      2 D2

TypeError: unhashable type: 'list'
```

[7]:
```
D2={(1,2,3):'safdar'}
D2
```

[7]:
```
{(1, 2, 3): 'safdar'}
```

[8]:
```
D3={"name":'safdar',"Gender":'Male','Age':38,'Marks':{'m1':66,'ds':70,'ai':77}}
D3
```

[8]:
```
{'name': 'safdar',
 'Gender': 'Male',
 'Age': 38,
 'Marks': {'m1': 66, 'ds': 70, 'ai': 77}}
```

[9]:
```
D1
```

[9]:
```
{'name': 'khan', 'Gender': 'Male'}
```

[10]:
```
#access
D1[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\PROGRA~1\KMSpico\temp/ipykernel_9564/3823187793.py in <module>
      1 #access
----> 2 D1[0]

KeyError: 0
```

[11]:
```
D1['name']
```

[11]:
```
'khan'
```

[12]:
```
D3['Marks']
```

[12]:
```
{'m1': 66, 'ds': 70, 'ai': 77}
```

[14]:
```
D3['Marks']['ai']
```

[14]:
```
77
```

[15]:
```
D3['ai']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\PROGRA~1\KMSpico\temp/ipykernel_9564/4113960414.py in <module>
----> 1 D3['ai']

KeyError: 'ai'
```

[16]:
```
D3.get('Gender')
```

[16]:
```
'Male'
```

[18]:
```
D3.get('ai')
```

[19]:
```
#Edit
D3
```

[19]:
```
{'name': 'safdar',
 'Gender': 'Male',
 'Age': 38,
 'Marks': {'m1': 66, 'ds': 70, 'ai': 77}}
```

[20]:
```
D3['name']='Dr. Sardar'
D3
```

[20]:
```
{'name': 'Dr. Sardar',
 'Gender': 'Male',
 'Age': 38,
 'Marks': {'m1': 66, 'ds': 70, 'ai': 77}}
```

[21]:
```
D3['Marks']['ai']= 85
D3
```

[21]:
```
{'name': 'Dr. Sardar',
 'Gender': 'Male',
 'Age': 38,
 'Marks': {'m1': 66, 'ds': 70, 'ai': 85}}
```

[22]:
```
#Add
D1
```

[22]:
```
{'name': 'khan', 'Gender': 'Male'}
```

[23]:
```
D1['Age']=23
D1
```

[23]:
```
{'name': 'khan', 'Gender': 'Male', 'Age': 23}
```

[24]:
```
D1[101]=20
D1
```

[24]:
```
{'name': 'khan', 'Gender': 'Male', 'Age': 23, 101: 20}
```

[25]:
```
#Delete
D
```

[25]:
```
{}
```

[26]:
```
del D
```

[27]:
```
D
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\PROGRA~1\KMSpico\temp/ipykernel_9564/3632740253.py in <module>
----> 1 D

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

[28]:
```
del D1["Gender"]
```

[29]:
```
D1
```

[29]:
```
{'name': 'khan', 'Age': 23, 101: 20}
```

[30]:
```
D1.clear()
D1
```

[30]:
```
{}
```

[31]:
```
#operators
D2
```

[31]:
```
{(1, 2, 3): 'safdar'}
```

[32]:
```
D3
```

[32]:
```
{'name': 'Dr. Sardar',
 'Gender': 'Male',
 'Age': 38,
 'Marks': {'m1': 66, 'ds': 70, 'ai': 85}}
```

[33]:
```
D2 * 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\PROGRA~1\KMSpico\temp/ipykernel_9564/1167296547.py in <module>
----> 1 D2 * 3

TypeError: unsupported operand type(s) for *: 'dict' and 'int'
```

[34]:
```
for i in D3:
    print(i)
```

```
name
Gender
Age
Marks
```

[35]:
```
for i in D3:
    print(i,D3[i])
```

```
name Dr. Sardar
Gender Male
Age 38
Marks {'m1': 66, 'ds': 70, 'ai': 85}
```

[36]:
```
'Age' in D3
```

[36]: True

[38]:
```
"Male" in D3
```

[38]: False

[39]:
```
#Fuction
len(D3)
```

[39]: 4

[40]:
```
D3
```

[40]:
```
{'name': 'Dr. Sardar',
 'Gender': 'Male',
 'Age': 38,
 'Marks': {'m1': 66, 'ds': 70, 'ai': 85}}
```

[41]:
```
min(D3)
```

[41]: 'Age'

[42]:
```
max(D3)
```

[42]:
```
'name'
```

[43]:
```
sorted(D3)
```

[43]:
```
['Age', 'Gender', 'Marks', 'name']
```

[44]:
```
sorted(D3,reverse=True)
```

[44]:
```
['name', 'Marks', 'Gender', 'Age']
```

[45]:
```
D3.keys()
```

[45]:
```
dict_keys(['name', 'Gender', 'Age', 'Marks'])
```

[46]:
```
D3.values()
```

[46]:
```
dict_values(['Dr. Sardar', 'Male', 38, {'m1': 66, 'ds': 70, 'ai': 85}])
```

2 lambda function

In Python, a lambda function is a small, anonymous function defined without a name, using the lambda keyword. It's often used when a simple function is needed for a short period and can contain only a single expression. Unlike regular functions defined with def, lambda functions are typically used for quick, throwaway functionality.

lambda arguments: expression

[49]:
```
# A lambda function that adds 10 to a number
add_10 = lambda x: x + 10
print(add_10(100))
# Output: 15
# Here, lambda x: x + 10 creates a function that takes one argument x and␣
 ↪returns x + 10.
```

```
110
```

[52]:
```
# Lambda function to add two numbers
add = lambda x, y: x + y
print(add(5, 5))  # Output: 8
```

```
10
```

[53]:
```
z=lambda x,y: x*y
z(4,8)
```

[53]: 32

[54]:
```
type(z)
```

[54]: function

[55]:
```
b=lambda x:x[0]=='a'
b('apple')
```

[55]: True

[58]:
```
b=lambda x:x[0]=='a'
b('bapple')
```

[58]: False

[60]:
```
b=lambda x:'even' if x%2 == 0 else "odd"
b(5)
```

[60]: 'odd'

[61]:
```
b(4)
```

[61]: 'even'

3 Use Cases for Lambda Functions

Lambda functions are most commonly used in situations where you need a simple, short function for a single-use purpose. They are frequently used with higher-order functions like # map(), filter(), and reduce().

Lambda Function with map() map() applies a function to all items in a list (or any iterable) and returns a map object. Example:

[62]:
```
numbers = [1, 2, 3, 4]
squared = list(map(lambda x: x ** 2, numbers))
print(squared)
# Output: [1, 4, 9, 16]
```

```
[1, 4, 9, 16]
```

[63]:
```
# Lambda Function with filter()
# filter() filters elements based on a condition:
numbers = [1, 2, 3, 4, 5, 6]
evens = list(filter(lambda x: x % 2 == 0, numbers))
print(evens)  # Output: [2, 4, 6]
```

```
[2, 4, 6]
```

#Lambda Function with reduce() #reduce() is used to apply a function cumulatively to all elements in a list:

[64]:
```
from functools import reduce
numbers = [1, 2, 3, 4]
product = reduce(lambda x, y: x * y, numbers)
print(product)
```

```
24
```

4 Key Differences Between Lambda and def Functions

Anonymous: Lambda functions are not named unless you explicitly assign them to a variable (but they remain unnamed in the underlying code).

Single Expression: Lambda functions can only contain a single expression, unlike functions defined using def, which can have multiple statements.

Inline Use: Lambdas are often used in situations where defining a full function using def would be overkill.

Limitations of Lambda Functions:

Limited Functionality: Lambda functions can only contain a single expression. Complex functions with multiple lines and more logic are better suited for regular def functions.

Readability: While concise, lambda functions can make code harder to read when overused or in complex cases.

No Docstrings: Lambda functions do not support docstrings (documentation inside the function), which can make them harder to document.

Advantages of Lambda Functions:

Conciseness: Lambda functions provide a compact syntax for short functions. Use in Functional Programming: Useful with functions like map(), filter(), reduce(), and list comprehensions.

One-liners: Convenient for simple operations that can be written in one line.

[ ]:
```
#fuction capitalize/Title/Upper/Lower?swapcase
```

[65]:
```
c="kolkata"
c
```

[65]:
```
'kolkata'
```

[66]:
```
c.capitalize()
```

[66]:
```
'Kolkata'
```

[67]:
```
"it is raining".title()
```

[67]:
```
'It Is Raining'
```

[68]:
```
"KOlkaTA".swapcase()
```

[68]:
```
'koLKAta'
```

[71]:
```
"It is raninnng".count("i")
```

[71]:
```
2
```

[72]:
```
"It is raninnng".count("n")
```

[72]:
```
4
```

[79]:
```
#find / index
'It is raining'.find("i")
```

[79]:
```
3
```

[74]:
```
'It is raining'.index("s")
```

[74]:
```
4
```

[77]:
```
'It is raining'.index("n")
```

[77]:
```
9
```

[88]:
```
'It is raining'.endswith("ing")
```

[88]:
```
True
```

[91]:
```
'It is raining'.startswith("It")
```

[91]:
```
True
```

[92]:
```
"hello my name is {} and i am {}".format('safdar','khan')
```

[92]:
```
'hello my name is safdar and i am khan'
```

[97]:
```
"hello my name is {} and i am {}".format(777, "khan", "happy")
```

[97]:
```
'hello my name is 777 and i am khan'
```

[94]:
```
"hello my name is {1} {0} and i am {0} {2}".format('safdar','khan',101)
```

[94]:
```
'hello my name is khan safdar and i am safdar 101'
```

[95]:
```
"hello my name is {name} and i am {age}".format(name='safdar',age=39)
```

[95]:
```
'hello my name is safdar and i am 39'
```

[98]:
```
# Validation in fuction
"Flat20".isalnum()
```

[98]:
```
True
```

[104]:
```
"Flat".isalnum()
```

[104]:
```
True
```

[102]:
```
"Flat$".isalpha()
```

[102]:
```
False
```

[126]:
```
"Flat".isalpha()
```

[126]:
```
True
```

[105]:
```
"20".isdigit()
```

[105]:
```
True
```

[106]:
```
"20ww".isdigit()
```

[106]:
```
False
```

[107]:
```
"hello world".isidentifier()
```

[107]:
```
False
```

[108]:
```
"hello_world".isidentifier()
```

[108]:
```
True
```

[110]:
```
"for".isidentifier()
```

[110]:
```
True
```

[111]:
```
"who is the pm of india".split()
```

[111]:
```
['who', 'is', 'the', 'pm', 'of', 'india']
```

[112]:
```
"who is the pm of india".split('pm')
```

[112]:
```
['who is the ', ' of india']
```

[113]:
```
"who is the pm of india".split('i')
```

[113]:
```
['who ', 's the pm of ', 'nd', 'a']
```

[114]:
```
" ".join(['who', 'is', 'the', 'pm', 'of', 'india'])
```

[114]:
```
'who is the pm of india'
```

[115]:
```
"/".join(['who', 'is', 'the', 'pm', 'of', 'india'])
```

[115]:
```
'who/is/the/pm/of/india'
```

[116]:
```
"-".join(['who', 'is', 'the', 'pm', 'of', 'india'])
```

[116]:
```
'who-is-the-pm-of-india'
```

[117]:
```
'Hi my name is safdar'.replace("safdar","khan")
```

[117]:
```
'Hi my name is khan'
```

[119]:
```
name = "        safdar           "
```

[120]:
```
name
```

[120]:
```
'        safdar           '
```

[121]:
```
"hi" + name
```

[121]:
```
'hi        safdar           '
```

[122]:
```
name.strip()
```

[122]:
```
'safdar'
```

[ ]:
```
"hello".
```