**STRING FUNCTIONS**

**Code:**

from \_cffi\_backend import string

s=*"You are learing Python"*

print(s)

st=*"""*

*This is the string*

*Function of the Python*

*Programming"""*

print(st)

# INDEXING

print(s[0])

#REPUTATION

print(s\*3)

#LENGTH

print(len(s))

print(len(st))

**Output**

You are learing Python

This is the string

Function of the Python

Programming

Y

You are learing PythonYou are learing PythonYou are learing Python

22

54

**SLICING**

**Code:**

#SLICING

print(s[0:5])

print(s[5:])

print(s[:5])

print(s[-3:-1])

#PASSING A STEP VALUE - 3rd VALUE

print(s[0:10:1])

print(s[0:10:2])

print(s[15::-2])

#SIMPLE REVERSE A STRING

print(s[::-1])

**Output**

You a

re learing Python

You a

ho

You are le

Yuael

nre r o

nohtyP gnirael era uoY

**STRIPE**

#STRIP - TO TRUNCATE THE SPACES AT BEGINNING AND END

s1=*" DevOps is the Future along with Python "*

print(s1)

print(s1.strip())

print(s1.lstrip())

print(s1.rstrip())

**Output**

DevOps is the Future along with Python

DevOps is the Future along with Python

DevOps is the Future along with Python

DevOps is the Future along with Python

**SUBSTRING**

#SUBSTRING

s1=*" DevOps will work along with Programming Language, Cloud, CI/CD Pipeline. Python is programming language. AWS is Cloud"*

print(s1)

print(s1.find(*"Python"*,0,len(s1)))

print(s1.find(*"Python"*,0,15))

print(s1.count(*"Py"*))

print(s1.replace(*"along"*, *" "*))

print(s1.upper())

print(s1.lower())

print(s1.title())

**Output**

DevOps will work along with Programming Language, Cloud, CI/CD Pipeline. Python is programming language. AWS is Cloud

74

-1

1

DevOps will work with Programming Language, Cloud, CI/CD Pipeline. Python is programming language. AWS is Cloud

DEVOPS WILL WORK ALONG WITH PROGRAMMING LANGUAGE, CLOUD, CI/CD PIPELINE. PYTHON IS PROGRAMMING LANGUAGE. AWS IS CLOUD

devops will work along with programming language, cloud, ci/cd pipeline. python is programming language. aws is cloud

Devops Will Work Along With Programming Language, Cloud, Ci/Cd Pipeline. Python Is Programming Language. Aws Is Cloud

**COLLECTIONS IN PYTHON**

1. List – Any number of objects dynamically
2. Set – Does not allow duplicates
3. Dictionary – Map – Store Key and Value Pairs

**LIST**

**Code:**

emptyLst=[]

print(emptyLst)

dataList=[101,*"Kumar"*,*"Ranjan"*,*"05-09-1976"*,48,*"Dublin"*]

print(dataList)

print(dataList[3])

print(dataList[1:3])

print(dataList\*2)

print(len(dataList))

#ADD / REMOVE / DELETE

dataList.append(425.75)

dataList.append(True)

print(dataList)

dataList.remove(48)

print(dataList)

dataList.append(48)

print(dataList)

del(dataList[4])

print(dataList)

dataList.insert(5, *"K78VT41"*)

dataList.insert(3, *"087-7684507"*)

# MAX/ MIN / SORT

intList=[10,20,40,75,99,-10,-20,-50]

print(max(intList))

print(min(intList))

intList.insert(3,50)

print(intList)

intList.sort()

print(intList)

intList.sort(reverse=True)

print(intList)

#CLEAR

dataList.clear()

print(dataList)

**Output**

[]

[101, 'Kumar', 'Ranjan', '05-09-1976', 48, 'Dublin']

05-09-1976

['Kumar', 'Ranjan']

[101, 'Kumar', 'Ranjan', '05-09-1976', 48, 'Dublin', 101, 'Kumar', 'Ranjan', '05-09-1976', 48, 'Dublin']

6

[101, 'Kumar', 'Ranjan', '05-09-1976', 48, 'Dublin', 425.75, True]

[101, 'Kumar', 'Ranjan', '05-09-1976', 'Dublin', 425.75, True]

[101, 'Kumar', 'Ranjan', '05-09-1976', 'Dublin', 425.75, True, 48]

[101, 'Kumar', 'Ranjan', '05-09-1976', 425.75, True, 48]

99

-50

[10, 20, 40, 50, 75, 99, -10, -20, -50]

[-50, -20, -10, 10, 20, 40, 50, 75, 99]

[99, 75, 50, 40, 20, 10, -10, -20, -50]

[]

**TUPLE**

Its like a list, which is immutable / read-only list. It maintains the insertion order, which allows duplicates. Different Types can be added. () is optional. If single value, have to be created with (comma - ,).

**Code:**

tupList = (20,30,40,50,60,70)

print(type(tupList))

print(tupList)

tupList1 = (*"Kumar"*, 48, *"Jesu"*, True, *"k78vt41"*)

print(type(tupList1))

print(tupList1)

tupList2=(23,)

print(type(tupList2))

print(tupList2)

print(tupList\*3)

print(tupList.count(20))

print(tupList1.index(*"Kumar"*))

**Output**

<class 'tuple'>

(20, 30, 40, 50, 60, 70)

<class 'tuple'>

('Kumar', 48, 'Jesu', True, 'k78vt41')

<class 'tuple'>

(23,)

(20, 30, 40, 50, 60, 70, 20, 30, 40, 50, 60, 70, 20, 30, 40, 50, 60, 70)

1

0

**Convert from List to Tuple**

**Code**

lst=[25,75.00,-100,True,*"Kumar"*]

print(type(lst))

tup=tuple(lst)

print(type(tup))

**Output**

<class 'list'>

<class 'tuple'>

**LIST**

1. Has to be inside the [], can use comma to separate, but [] are mandatory
2. List is Mutable – which we can modify the records such as add / delete / insert
3. Cannot be Key to Dictionary – because it has to be Hashable and Immutable.

**TUPLE**

1. Has to be inside (), can use , to separate, but () are optional
2. Tuple is Immutable – Which we cannot modify the records
3. Since its Immutable, we can use Tuple for Key to Dictionary

**SET**

Set Object does not support Indexing / Scaling / Reputation.

**Code**

datSet={10,20,30,40,50, *"Jerrick"*}

print(datSet)

print(type(datSet))

datSet1={10,20,30,40,50,20,30,40}

print(datSet1)

print(type(datSet1))

datSet.update([25,100,*"Jazlyn"*])

print(datSet)

print(type(datSet))

datSet.remove(10)

print(datSet)

print(type(datSet))

**Output**

{'Jerrick', 50, 20, 40, 10, 30}

<class 'set'>

{50, 20, 40, 10, 30}

<class 'set'>

{'Jerrick', 10, 20, 25, 30, 100, 40, 'Jazlyn', 50}

<class 'set'>

{'Jerrick', 20, 25, 30, 100, 40, 'Jazlyn', 50}

<class 'set'>

**FROZEN SET**

Frozen set does not support delete / update. Its used to navigate and print the data using for loop. Its Immutable.

**RANGE**

**Code**

rangeValue = range(10)

rangeValue1 = range(1,10)

rangeValue2 = range(1,10,3)

for i in rangeValue:

print(i)

for j in rangeValue1:

print(j)

for k in rangeValue2:

print(k)

**Output**

0

1

2

3

4

5

6

7

8

9

1

2

3

4

5

6

7

8

9

1

4

7

**BYTES & BYTEARRAY**

**Code**

#BYTES AND BYTEARRAY

lst=[10,20,30,40,50]

print(type(lst))

byt=bytes(lst)

print(type(byt))

#Cannot add or remove in the byte

# No slicing or reputation is allowed on bytes or bytesarray

# bytes – Immutuable

# bytearray - Mutable

bytArr=bytearray(lst)

print(type(bytArr))

bytArr[2]=25

print(bytArr)

**Output**

<class 'list'>

<class 'bytes'>

<class 'bytearray'>

bytearray(b'\n\x14\x19(2')

**DICTIONARY**

**Code**

#DICTIONARY

dctVal={1:*"Jazy"*,2:*"Jerrick"*,3:*"Augusta"*,4:*"Kumar"*}

print(dctVal)

print(dctVal.values())

keyVal=dctVal.keys()

for i in keyVal:

print(i)

val=dctVal.values()

for j in val:

print(j)

del dctVal[4]

print(dctVal)

**Output**

{1: 'Jazy', 2: 'Jerrick', 3: 'Augusta', 4: 'Kumar'}

dict\_values(['Jazy', 'Jerrick', 'Augusta', 'Kumar'])

1

2

3

4

Jazy

Jerrick

Augusta

Kumar

{1: 'Jazy', 2: 'Jerrick', 3: 'Augusta'}

**IMMUTABLE**

A diagram of a mathematical equation

Description automatically generated with medium confidence

**Code**

# IMMUTABLE EXAMPLE

a=100

b=100

print(id(a))

print(id(b))

print(a is b)

a=105

b=200

print(id(a))

print(id(b))

print(a is b)

c=*"Jazy"*

d=*"Jazy"*

print(id(c))

print(id(d))

print(c is d)

d=*"Jerrick"*

print(id(c))

print(id(d))

print(c is d)

**Output**

140728583738904

140728583738904

True

140728583739064

140728583742104

False

1801794306096

1801794306096

True

1801794306096

1801794115968

False

**DATA STRUCTURES**

**Code**

students = {

*'101'*:[*'Kumar'*,48,*"C# Developer"*],

*'102'*:[*'Purusoth'*,48,*"Network Engineer"*],

*'103'*:[*'Ragav'*,38,*"Systems Admin"*],

*'104'*:[*'Kingsly'*,45,*"Java Developer"*],

*'105'*:[*'Anwar'*,40,*"Technical Consultant"*],

*'106'*:[*'Senthil'*,38,*"Team Leader"*]

}

keys = students.keys()

for eachkey in keys:

print(*"Student Details of Id : "*, eachkey)

print(*"============================="*)

for eachval in students[eachkey]:

print(eachval)

**Output**

Student Details of Id : 101

=============================

Kumar

48

C# Developer

Student Details of Id : 102

=============================

Purusoth

48

Network Engineer

Student Details of Id : 103

=============================

Ragav

38

Systems Admin

Student Details of Id : 104

=============================

Kingsly

45

Java Developer

Student Details of Id : 105

=============================

Anwar

40

Technical Consultant

Student Details of Id : 106

=============================

Senthil

38

Team Leader

**QUIZ**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a login page

Description automatically generated

A screenshot of a computer

Description automatically generated

**NONE TYPE –** When the function m1 is not returning any values, this will return none as output.

**Code**

def **m1**():

a=10

print(m1())

**Output**

None

**ESC CHARACTERS**

**Code**

print(*"\tPython \n is a commonly used programming language in the \n \"DEVOPS\" Environment"*)

**Output**

Python

is a commonly used programming language in the

"DEVOPS" Environment

**CONSTANTS**

# CONSTANT VALUES - Convention for Defining Constants

PI = 3.14

**DEL – KEYWORK TO DELETE AN OBJECT**

**Code**

#DELETE - DEL OBJECTS

a=10

del a

print(a)

**Output**

The below will throw an error;

Traceback (most recent call last):

File "C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\pythonBasics\escChar.py", line 9, in <module>

print(a)

^

NameError: name 'a' is not defined

**Code**

#DELETE – STRING DEL OBJECTS

s=*"kumar"*

del s

s=None

print(s)

s=*"augusta"*

print(s)

**Output**

None

Augusta

**QUIZ**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated