1.'hello'.capitalize()

2. def cls():

print('\n' \* 50)

3.

See email

Uppercase

Lists—linear representation of data-hetrogenous(int, string,float)

Fruits—apple,banana,cherry,date-----🡪homogenous

Fruits—apple,3,banana,2,cherry,5,date,1-----🡪heterogenous

Tuples=readonly data

Ex:days of week. We wont want 8 day week

Dictionary:unique identitifier

Ex:Aadhar, SSN.

Array:it depends on values something like index values

Dictionary : depends on keys

Functions: block of code within program used for reuse.

Modules:

P1 – f1, f2, f3

P2- f1,f2, p1, f1

Python

You can use excel also

Exceptions

EDxception Handling

Regular Expression\*

Classes – oops

APIE

Debugging in python

Logging

API->browser-🡪program(python)->database

CGI-common Gateway interface- Way in which python program is interacting with browser

Database Connectivity:

Pycharm

1)import random(number generator)

import sys

import os

2)print("Ravo")

3)#comment

4)‘’’

Comments

‘’’

5)name =”Derek”

print(name)

6) 5 main types in python

1.Numbers

2. Strings

3. Lists

4. Tuples

5. Dictionaries

7)Arithmetic operators

+,-,\*,/,%,\*\*(exponential calcaulations),//----7 of them

print(“5 +2=”,5+2)--- 5 +2= 7

print(“5 -2=”,5-2)---- 5 -2= 3

print(“5 \*2=”,5\*2)--- 5 \*2= 10

print(“5 /2=”,5/2)---- 5 /2= 2.5

print(“5 \*\*2=”,5\*\*2)--- 5 \*\*2= 25

print(“5 //2=”,5//2)---- 5 //2= 2

print(“5 %2=”,5%2)--- 5 %2= 1

print(“1+2-3\*2=”,1+2-3\*2)------ -3

print(“(1+2-3)\*2=”,(1+2-3)\*2)—0

8) quote="\"Always remember you are unique\""

multi\_line\_quote='''\"just

... like everyone else\"'''

print(“%s %s %s” %(‘I like the quote’,quote,multi\_line\_quote))

9)print('\n'\*5)

10)print("I don't like the ",end="")

print("newlines")

11)**lists**

a) grocery\_list=['Juice','Tomatoes','Potatoes','Bananas']

>>> print(grocery\_list)

['Juice', 'Tomatoes', 'Potatoes', 'Bananas']

b)print(“First\_item:” ,grocery\_list[0])

c)grocery\_list[0]="greenjuice"

>>> print("First\_item:" ,grocery\_list[0])

First\_item: greenjuice

d) >>> print(grocery\_list[1:3])

['Tomatoes', 'Potatoes']

e)

>>> other\_events=['Wash Car','Pickup Kids','Cash Check']

>>> to\_do\_list=[other\_events,grocery\_list]

>>> print(to\_do\_list)

[['Wash Car', 'Pickup Kids', 'Cash Check'], ['greenjuice', 'Tomatoes', 'Potatoes', 'Bananas']]

f) >>> print((to\_do\_list**[1][1]**))

Tomatoes

g) >>> grocery\_list.**append**('Onions')

>>> print(to\_do\_list)

[['Wash Car', 'Pickup Kids', 'Cash Check'], ['greenjuice', 'Tomatoes', 'Potatoes', 'Bananas', 'Onions']]

h) >>> grocery\_list.**insert**(1,'Pickle')

>>> print(to\_do\_list)

[['Wash Car', 'Pickup Kids', 'Cash Check'], ['greenjuice', 'Pickle', 'Tomatoes', 'Potatoes', 'Bananas', 'Onions']]

i) >>> grocery\_list.**remove**('Onions')

>>> print(to\_do\_list)

[['Wash Car', 'Pickup Kids', 'Cash Check'], ['greenjuice', 'Pickle', 'Tomatoes', 'Potatoes', 'Bananas']]

j) >>> grocery\_list.**sort**()

>>> print(to\_do\_list)

[['Wash Car', 'Pickup Kids', 'Cash Check'], ['Bananas', 'Pickle', 'Potatoes', 'Tomatoes', 'greenjuice']]

>>> grocery\_list.**reverse**()

>>> print(to\_do\_list)

[['Wash Car', 'Pickup Kids', 'Cash Check'], ['greenjuice', 'Tomatoes', 'Potatoes', 'Pickle', 'Bananas']]

k) >>> **del** grocery\_list[4]

>>> print(to\_do\_list)

[['Wash Car', 'Pickup Kids', 'Cash Check'], ['greenjuice', 'Tomatoes', 'Potatoes', 'Pickle']]

l) >>> to\_do\_list2=other\_events**+**grocery\_list

>>> print(to\_do\_list2)

['Wash Car', 'Pickup Kids', 'Cash Check', 'greenjuice', 'Tomatoes', 'Potatoes', 'Pickle']

m) >>> print(**len**(to\_do\_list2))

7

n) >>> print(**max**(to\_do\_list2))

greenjuice

o) >>> print(**min**(to\_do\_list2))

Cash Check

**Tuples**

Tuples are surrounded by () like a ball

pi\_tuple = (3,1,4,1,5,9)

**tuple to list**

new\_tuple=list(pi\_tuple)

**list to tuple**

new\_list=tuple(new\_tuple)

>>> **max**(new\_tuple)

9

>>> **min**(new\_tuple)

1

**Dictionaries**:

a)It consists of unique keys. Joins cannot happen. It uses curly braces{}.key Value Pair

b)>>> super\_villains={'Fiddler':'Isaac','Captain':'Leonard','Weather':'Mark','Mirror':'Sam','Pied':'Thomas'}

>>> print(super\_villains)

{'Weather': 'Mark', 'Captain': 'Leonard', 'Pied': 'Thomas', 'Mirror': 'Sam', 'Fiddler': 'Isaac'}

c)>>> **print(super\_villains['Captain'])**

Leonard

>>> print(super\_villains['Captain'])

Leonard

d) >>> **super\_villains['Pied']='hartley'**

>>> print(super\_villains)

{'Weather': 'Mark', 'Captain': 'Leonard', 'Pied': 'hartley', 'Mirror': 'Sam'}

e) >>> **print(super\_villains.get('Pied'))**

Hartley

f) >>> print(super\_villains.keys())

dict\_keys(['Weather', 'Captain', 'Pied', 'Mirror'])

g) >>> print(super\_villains.values())

dict\_values(['Mark', 'Leonard', 'hartley', 'Sam'])

**Conditions**

ifelse elif == != > >= <=

a)>>> age=21

>>> if age>16:

... print('you are old enough to drive')

... else:

... print('you are not old enough to drive')

...

you are old enough to drive

b) >>> if age>21:

... print('tractor')

... elif age>=16:

... print('car')

... else:

... print('no\_drive')

...

Car

**Logical Operators**

And,or,not

a)>>> if ((age>=1)and (age<=18)):

... print("bday")

... elif(age==21):

... print("major")

... elif not(age==30):

... print("middle")

... else:

... print("Heel")

...

Major

[‎02/‎13/‎17 9:49 AM] Talakokkula, Sai Kishore (CONT):

https://www.a2hosting.com/kb/developer-corner/postgresql/connecting-to-postgresql-using-python

[‎02/‎13/‎17 9:50 AM] Talakokkula, Sai Kishore (CONT):

for sql server, https://blogs.msdn.microsoft.com/cdndevs/2015/03/11/python-and-data-sql-server-as-a-data-source-for-python-applications/