python-refreshment-part-l

March 4, 2024

1 Index for Python Tutorial

1.1 Python Type Casting

1. int()
2. float()
3. bool()
4. complex()
5. str()

1.2 Mutability and Immutability

- Explanation of mutability and immutability
- Functions:
  - split() and join() functions in str()

1.3 List

- Pre-Defined Functions in List:
  1. append()
  2. insert()
  3. remove()
  4. pop(index)
  5. pop()
  6. clear()
  7. index()
  8. copy()
  9. count()
  10. reverse()
  11. sort()
  12. extend()

1.4 Tuple

- Pre-defined Functions in Tuple:
  - index()
  - count()

1.5 Set

- Pre-defined Functions in Set:
    1. add()
    2. clear()
    3. remove()
    4. discard()
    5. pop()
    6. copy()
    7. update()
    - isdisjoint()
    - issuperset()
    - issubset()
    - union()
    - intersection()
    - difference()
    - symmetric_difference()

1.6 Questions

1. Find all the players who are playing all the games. (union())
2. Find all the players who are playing both Cricket and Tennis. (intersection())
3. Find all the players who are playing Only Cricket but not Tennis. (difference())
4. Find all the players who are playing Only Tennis but not Cricket. (difference())
5. Find all the players who are EXCLUSIVELY playing Tennis and Cricket. (symmetric_difference)

1.7 Dictionary

- Pre-defined Functions in Dictionary:
    - clear()
    - copy()
    - pop()
    - popitem()
    - keys()
    - values()
    - get()
    - items()
    - update()
- Special Points:
    - Nested Dict
    - List, Tuple, and Set in dict

1.8 Displaying Results

- Printing values with:
    - print(val1, val2, ..., val-n)
    - print(msg1, msg2, ..., msg-n)
    - print(message cum values)

  - print(message cum values with format())
  - print(message cum values with format specifiers)
  - print(value, end=)

1.9 Reading Data from Keyboard

- input()
- input(message)

1.10 Operators and Expressions

- Arithmatic Operators
- Assignment Operator
- Relational Operator (comparison)
- Logical Operators (comparison)
- Bitwise Operators
- Membership Operator
- Ternary Operator in Python

1.11 Flow Control Statements in Python

1.11.1 A. Conditional or Selection or Branching Statements

- Simple if statement
- if else statement
- if elif else statement
- match case statement ### B. Looping or Iterative or Repetitive Statements
- while loop or while..else loop
- for loop or for else loop ### C. Transfer Flow Control Statements
- break
- continue
- pass
- return ### D. Inner or Nested Loops
- for loop in for loop
- while loop in while loop
- for loop in while loop
- while loop in for loop

1.12 Pattern Programming

- Explanation and examples of pattern programming

1.13 Functions

- Positional Arguments
- Default parameters
- Keyword parameters (or) Arguments
- Variable length parameters
- Keyword Variable Length Parameters (or) arguments
- Recursion

1.14 Local variables and Global Variables

Practiced by

Prashant Sundge

---

2 PYTHON TYPE CASTING

- The process of converting one type of possible value into another possible type is called type casting
- in python we have 5 fundamental type casting techniques
  - 1.int()
  - 2.float()
  - 3.bool()
  - 4.complex()
  - 5.str()

3 1. int()

- syntax varname = int(float/bool/complex/str)

4 float type value –To –>> int type —> possible

[ ]:
```
a=12.45
print(type(a),a)
b=int(a)
print(type(b),b)
a=0.99
print(type(a),a)
b=int(a)
print(type(b),b)
```

```
<class 'float'> 12.45
<class 'int'> 12
<class 'float'> 0.99
<class 'int'> 0
```

5 bool type value –To–> int type —> possible

```
[ ]: a=True
     print(type(a),a)
     b=int(a)
     print(type(b),b)
     a=False
     print(type(a),a)
     b=int(a)
     print(type(b),b)
```

```
<class 'bool'> True
<class 'int'> 1
<class 'bool'> False
<class 'int'> 0
```

6 Complex type value –To–> int type —> Not Possible

- TypeError: int() argument must be a string, a bytes-like object or a real number, not 'complex'

```
[ ]: a=2+3j
     print(type(a),a)
     b=int(a)
     print(type(b),b)
```

```
<class 'complex'> (2+3j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-4ae889d634e3> in <cell line: 3>()
      1 a=2+3j
      2 print(type(a),a)
----> 3 b=int(a)
      4 print(type(b),b)
      5 # a=

TypeError: int() argument must be a string, a bytes-like object or a real␣
 ↪number, not 'complex'
```

Int str –to – int

```
[ ]: a='25'
     print(type(a),a)
     b=int(a)
     print(type(b),b)
```

```
<class 'str'> 25
<class 'int'> 25
```

[ ]:
```
# Float str to Int not possible
# ValueError: invalid literal for int() with base 10: '23.13'
a='23.13'
print(type(a),a)
b=int(a)
print(type(b),b)
```

```
<class 'str'> 23.13
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-fe099450aa62> in <cell line: 4>()
      2 a='23.13'
      3 print(type(a),a)
----> 4 b=int(a)
      5 print(type(b),b)

ValueError: invalid literal for int() with base 10: '23.13'
```

[ ]:
```
# Bool str to int not possible

a='True'
print(type(a),a)
b=int(a)
print(type(b),b)
```

```
<class 'str'> True
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-69a33cfbd3d2> in <cell line: 5>()
      3 a='True'
      4 print(type(a),a)
----> 5 b=int(a)
      6 print(type(b),b)

ValueError: invalid literal for int() with base 10: 'True'
```

[ ]:
```
#complex str to int not possible
a='2+3j'
print(type(a),a)
b=int(a)
print(type(b),b)
```

```
<class 'str'> 2+3j
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-dd8055f2970c> in <cell line: 4>()
      2 a='2+3j'
      3 print(type(a),a)
----> 4 b=int(a)
      5 print(type(b),b)

ValueError: invalid literal for int() with base 10: '2+3j'
```

[ ]:
```
#Pure str to int not possible
a='HYD'
print(type(a),a)
b=int(a)
print(type(b),b)
```

```
<class 'str'> HYD
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-f81cdfc45b09> in <cell line: 4>()
      2 a='HYD'
      3 print(type(a),a)
----> 4 b=int(a)
      5 print(type(b),b)

ValueError: invalid literal for int() with base 10: 'HYD'
```

7 2. Float()

- float() is used for converting one possible type of value in to float type
- syntax varname=float(int/bool/complex/str)

[ ]:
```
#Int to float
a=25
print(type(a),a)
b=float(a)
print(type(b),b)

#bool to float
a=True
print(type(a),a)
b=float(a)
```

```
print(type(b),b)
# str to float
a='23.6778'
print(type(a),a)
b=float(a)
print(type(b),b)
# str to float
a='49'
print(type(a),a)
b=float(a)
print(type(b),b)
```

```
<class 'int'> 25
<class 'float'> 25.0
<class 'bool'> True
<class 'float'> 1.0
<class 'str'> 23.6778
<class 'float'> 23.6778
<class 'str'> 49
<class 'float'> 49.0
<class 'str'> PAMS
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-22-cf3803118d84> in <cell line: 25>()
     23 a='PAMS'
     24 print(type(a),a)
---> 25 b=float(a)
     26 print(type(b),b)
     27

ValueError: could not convert string to float: 'PAMS'
```

[ ]:
```
#complex to float Not possible
#TypeError: float() argument must be a string or a real number, not 'complex'
a=3+4j
print(type(a),a)
b=float(a)
print(type(b),b)
```

```
<class 'complex'> (3+4j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-12715897f6df> in <cell line: 5>()
      3 a=3+4j
      4 print(type(a),a)
```

```
----> 5 b=float(a)
      6 print(type(b),b)

TypeError: float() argument must be a string or a real number, not 'complex'
```

[ ]:
```
# pure  str to float
a='PAMS'
print(type(a),a)
b=float(a)
print(type(b),b)
```

8 3. bool()

- bool() is used for converting one possible type of value into bool type value
- All non-zero values are considered as True
- All zero values are considered as False
- syntax varname = bool(int/float/str/comnplex)

[ ]:
```
# int to bool possible
a=1
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)
# full int to bool possible
a=19797
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)
# float to bool possible
a=19.764
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)
# Complex to bool possible
a=34+2j
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)
# int str to bool posible
a='3456'
print(type(a),a)
```

```
b=bool(a)
print(type(b),b)
print('-'*50)
# float str to bool posible
a='345.6'
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)
# bool str to bool posible
a='False'
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)
# Complex str to bool posible
a='3+4j'
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)
# pure str to bool posible
a='Python'
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)

a=''
print(type(a),a)
b=bool(a)
print(type(b),b)
print('-'*50)
```

```
<class 'int'> 1
<class 'bool'> True
--------------------------------------------------
<class 'int'> 19797
<class 'bool'> True
--------------------------------------------------
<class 'float'> 19.764
<class 'bool'> True
--------------------------------------------------
<class 'complex'> (34+2j)
<class 'bool'> True
--------------------------------------------------
<class 'str'> 3456
```

```
<class 'bool'> True
--------------------------------------------------
<class 'str'> 345.6
<class 'bool'> True
--------------------------------------------------
<class 'str'> False
<class 'bool'> True
--------------------------------------------------
<class 'str'> 3+4j
<class 'bool'> True
--------------------------------------------------
<class 'str'> Python
<class 'bool'> True
--------------------------------------------------
<class 'str'>
<class 'bool'> False
--------------------------------------------------
```

9 4. complex()

- complex() is used for converting one possible type of values into complex type values
- syntax varname=complex(float/int/bool/str)

[ ]:
```
# int to complex possible
a=1
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)
# full int to complex possible
a=19797
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)
# float to complex possible
a=19.764
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)
# Complex to complex possible
a=34+2j
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)
```

```
# int str to complex posible
a='3456'
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)
# float str to complex posible
a='345.6'
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)

# Complex str to complex posible
a='3+4j'
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)
```

```
<class 'int'> 1
<class 'complex'> (1+0j)
--------------------------------------------------
<class 'int'> 19797
<class 'complex'> (19797+0j)
--------------------------------------------------
<class 'float'> 19.764
<class 'complex'> (19.764+0j)
--------------------------------------------------
<class 'complex'> (34+2j)
<class 'complex'> (34+2j)
--------------------------------------------------
<class 'str'> 3456
<class 'complex'> (3456+0j)
--------------------------------------------------
<class 'str'> 345.6
<class 'complex'> (345.6+0j)
--------------------------------------------------
<class 'str'> 3+4j
<class 'complex'> (3+4j)
--------------------------------------------------
```

[ ]:
```
# bool str to complex posible
a='False'
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)
```

```
<class 'str'> False
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-40-4c19c515d139> in <cell line: 4>()
      2 a='False'
      3 print(type(a),a)
----> 4 b=complex(a)
      5 print(type(b),b)
      6 print('-'*50)

ValueError: complex() arg is a malformed string
```

[ ]:
```
# pure str to complex posible
a='Python'
print(type(a),a)
b=complex(a)
print(type(b),b)
print('-'*50)
```

```
<class 'str'> Python
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-41-c33e9c061b1e> in <cell line: 4>()
      2 a='Python'
      3 print(type(a),a)
----> 4 b=complex(a)
      5 print(type(b),b)
      6 print('-'*50)

ValueError: complex() arg is a malformed string
```

10 5. str()

- str() is used for converting all types of values into str type values
- syntax varname=str(float/int/bool/complex)

[ ]:
```
# int to str possible
a=1
print(type(a),a)
b=str(a)
print(type(b),b)
print('-'*50)
# full int to str possible
a=19797
print(type(a),a)
b=str(a)
print(type(b),b)
print('-'*50)
# float to str possible
a=19.764
print(type(a),a)
b=str(a)
print(type(b),b)
print('-'*50)
# complex to str possible
a=34+2j
print(type(a))
b=str(a)
print(type(b),b)
print('-'*50)
# int to str posible
a='3456'
print(type(a))
b=str(a)
print(type(b),b)
print('-'*50)
# fl str to str posible
a='345.6'
print(type(a))
b=str(a)
print(type(b),b)
print('-')

# str str to str posible
a='3'
print(type(a),a)
b=str(a)
print(type(b),b)
print('-'*50)
# bool to str posible
a=True
print(type(a),a)
b=str(a)
print(type(b),b)
print('-'*50)
a=False
print(type(a),a)
b=str(a)
```

```
print(type(b),b)
print('-'*50)
```

```
<class 'int'> 1
<class 'str'> 1
--------------------------------------------------
<class 'int'> 19797
<class 'str'> 19797
--------------------------------------------------
<class 'float'> 19.764
<class 'str'> 19.764
--------------------------------------------------
<class 'complex'>
<class 'str'> (34+2j)
--------------------------------------------------
<class 'str'>
<class 'str'> 3456
--------------------------------------------------
<class 'str'>
<class 'str'> 345.6
-
<class 'str'> 3
<class 'str'> 3
--------------------------------------------------
<class 'bool'> True
<class 'str'> True
--------------------------------------------------
<class 'bool'> False
<class 'str'> False
--------------------------------------------------
```

11 Mutability and Immutability

- mutability or mutable is one , which allows us to perform updations/modifications on the object at same address
- Example list(), set(),dict()
- immutability or immutable as object is one which will satisfy the following Properties
  - Value can be modified and modified placed in New address
  - immutable objects never allow to performs item assignment (does not support item assignments)
- Example int(), float(), complex(), bool(), str(), tuple(), bytes(), range(), set(), frozenset()

```
[ ]:
```

[ ]:
```
# mutable

list1=[10,20,30]
print(f" type {type(list1)} , id {id(list1)} , {list1}")
list1.append(25)
print(f" type {type(list1)} , id {id(list1)} , {list1}")
list1.pop()
print(f" type {type(list1)} , id {id(list1)} , {list1}")
```

```
type <class 'list'> , id 137387713497856 , [10, 20, 30]
type <class 'list'> , id 137387713497856 , [10, 20, 30, 25]
type <class 'list'> , id 137387713497856 , [10, 20, 30]
```

[ ]:
```
# immutable
# Once an integer object is created, its value cannot be changed.
a=10
print(f" type {type(a)} , id {id(a)} , {a}")
a=25
print(f" type {type(a)} , id {id(a)} , {a}")
tup = (1, 2, 3)
# Attempting to modify elements of tup will result in an error.
print(f" type {type(tup)} , id {id(tup)} , {tup}")
```

```
type <class 'int'> , id 137388995068432 , 10
type <class 'int'> , id 137388995068912 , 25
type <class 'tuple'> , id 137387714580352 , (1, 2, 3)
```

12 str() predefined function

[ ]:
```
string = 'python is awsome..!'
print(string)
print('-'*50)
#capitalize()
print(f" Capitalize {string.capitalize()}")
print('-'*50)
#title()
print(f" Title {string.title()}")
print('-'*50)
#index()
print(f" index {string.index('s')}")
print('-'*50)
# find index values using for loop and enumerate
s='python'
for i , v in enumerate(s):
  print(f" index : -- {i}  values: -- { v} ")
print('-'*50)
# find
```

```python
s='python'
print(f" index using slice {string.index(s[-1])}")
print(f" index using slice {string.index(s[2])}")
print(f" index using slice {string.index(s[-5])}")
print('-'*50)
#title()
print(f" Title {string.title()}")
print('-'*50)
#upper()
print(f" convert the string to upper: {string.upper()}")
print('-'*50)
#lower()
print(f" convert the string to lower : {string.lower()}")
print('-'*50)
#isupper()
print(f" True if string is upper else False: {string.isupper()}")
print('-'*50)
#islower()
print(f" check if string lower : {string.islower()}")
print('-'*50)
#isalpha()
print(f" string check if alpha: {string.isalpha()}")
print('-'*50)
#isdigit()
print(f" will check if string is digit: {string.isdigit()}")
print('-'*50)
#isalnum()
print(f" will check if all string is number : {string.isalnum()}")
print('-'*50)
#isspace()
space=" "
print(f" check purely space like : {space.isspace()}")
print(f" check purely space like : {string.isspace()}")
print('-'*50)
```

```
python is awsome..!
--------------------------------------------------
 Capitalize Python is awsome..!
--------------------------------------------------
 Title Python Is Awsome..!
--------------------------------------------------
 index 8
--------------------------------------------------
 index : -- 0  values: -- p
 index : -- 1  values: -- y
 index : -- 2  values: -- t
 index : -- 3  values: -- h
 index : -- 4  values: -- o
 index : -- 5  values: -- n
--------------------------------------------------
 index using slice 5
 index using slice 2
 index using slice 1
--------------------------------------------------
 Title Python Is Awsome..!
--------------------------------------------------
 convert the string to upper: PYTHON IS AWSOME..!
--------------------------------------------------
 convert the string to lower : python is awsome..!
--------------------------------------------------
 True if string is upper else False: False
--------------------------------------------------
 check if string lower : True
--------------------------------------------------
 string check if alpha: False
--------------------------------------------------
 will check if string is digit: False
--------------------------------------------------
 will check if all string is number : False
--------------------------------------------------
 check purely space like : True
 check purely space like : False
--------------------------------------------------
```

13 split() and join() functions in str()

split() - split() function is used for splitting the given str obejct data into differnet words base specifieed delimeter (- _ # % ^ & * etc) - the default deliemter is space - the function returns spiting data in the form of list objects - syntax strobj.split('delimter') or strobj.split()

join()

- This Function is used for combining or joining list of values from any iterable object
- systax strobj.join(iterableobject)

[ ]:
```python
s='python is awesome language '
print(len(s))
print(s.split())
print(len(s.split(' ')), type(s.split()))

s='29-02-2024'
split_s=s.split('-')
print(type(s))
print(type(split_s), split_s)
```

```
s='29-02-2024'
dob=s.split('-')
print('Day', dob[0])
print('Month', dob[1])
print('year', dob[2])
# maxsplit
date = "2024-02-29"
parts = date.split('-', maxsplit=1)
print("Parts:", parts)
```

```
27
['python', 'is', 'awesome', 'language']
5 <class 'list'>
<class 'str'>
<class 'list'> ['29', '02', '2024']
Day 29
Month 02
year 2024
Parts: ['2024', '02-29']
```

[ ]:
```
#join
#list to str
# join will convert the list data to str data
lst=['MH', "TN", 'KA', 'AP']
print(lst, type(lst))
s=' '
print(s.join(lst), '\n' ,type(s.join(lst)))

words = ['Hello', 'world', 'from', 'Python']
sentense=' '.join(words)
print(sentense)

numbers = [1, 2, 3, 4, 5]
print(type(numbers), numbers)
result=' '.join(map(str, numbers))
print(type(result), result)
```

```
['MH', 'TN', 'KA', 'AP'] <class 'list'>
MH TN KA AP
 <class 'str'>
Hello world from Python
<class 'list'> [1, 2, 3, 4, 5]
<class 'str'> 1 2 3 4 5
```

14 list

- list is one of the pre-defined class and treated as list category data type.

- the list is that to store multiple values either of same data types or differnet data type
- on List object we can perform both
  - indexing
  - slicing
- list object belongs to mutable becuase it allow to update/ modify the values of list

15 pre-Defined Functions in List

- we can perform many other operations using pre-defined function which are present in the list object
  - 1.append()
  - 2.insert()
  - 3.remove() - Removed based on value
  - 4.pop(index) - removed based on specified index
  - 5.pop() - Removed based on last index
  - 6.clear()
  - 7.index()
  - 8.copy() - Shallow copy
  - 9.count()
  - 10.reverse()
  - 11.sort()
  - 12.extend()

[ ]:
```
#examples of list
example_list = [1, 2, 3, 4, 5,'raj', 3+3j, 34.67, 'python', ['MH', "AP"]]
print(type(example_list), len(example_list), example_list)
print('-'*50)
# indexing
print("o index at",example_list[0])
print('-'*50)
print("-2 index at",example_list[-2])
print('-'*50)
print("4 index at",example_list[4])
print('-'*50)
# slicing
print("o slicing starts ",example_list[0:])
print('-'*50)
print("[-6:-3] slicing at",example_list[-6:-3])
print('-'*50)
print("[-1] reverse slicing at",example_list[::-1])
print('-'*50)
```

```
<class 'list'> 10 [1, 2, 3, 4, 5, 'raj', (3+3j), 34.67, 'python', ['MH', 'AP']]
--------------------------------------------------
o index at 1
--------------------------------------------------
-2 index at python
--------------------------------------------------
```

```
4 index at 5
--------------------------------------------------
o slicing starts  [1, 2, 3, 4, 5, 'raj', (3+3j), 34.67, 'python', ['MH', 'AP']]
--------------------------------------------------
[-6:-3] slicing at [5, 'raj', (3+3j)]
--------------------------------------------------
[-1] reverse slicing at [['MH', 'AP'], 'python', 34.67, (3+3j), 'raj', 5, 4, 3,
2, 1]
--------------------------------------------------
```

[ ]:
```
# append()
lst=[10,2,30,40]
print(lst)
lst.append(50)
print(lst)
lst.append('prashant')
print(lst)

# for loop
for i in range(5):
  lst.append(i)
print(lst)
```

```
[10, 2, 30, 40]
[10, 2, 30, 40, 50]
[10, 2, 30, 40, 50, 'prashant']
[10, 2, 30, 40, 50, 'prashant', 0, 1, 2, 3, 4]
```

[ ]:
```
# insert
# Insert expected 2 arguments 1 is index value and the value which you want to␣
 ↪insert

lst=[10,2,30,40]
print(lst)
lst.insert(3,'Pams')
print(lst)
lst.insert(-1,'prashant')
print(lst)
```

```
[10, 2, 30, 40]
[10, 2, 30, 'Pams', 40]
[10, 2, 30, 'Pams', 'prashant', 40]
```

[ ]:
```
# remove
print(lst)
lst.remove('Pams')
print(lst)
lst.remove('prashant')
```

```
print(lst)
```

```
[10, 2, 30, 'Pams', 'prashant', 40]
[10, 2, 30, 'prashant', 40]
[10, 2, 30, 40]
```

[ ]:
```
# pop(index) - removed based on specified index

lst=[10,2,30,40]
for i, v in enumerate(lst):
    print(i, '-- ' , v)

lst.pop(3)
print(lst)
lst.pop(0)
print(lst)
```

```
0 --  10
1 --  2
2 --  30
3 --  40
[10, 2, 30]
[2, 30]
```

[ ]:
```
# pop() - removed based last index

lst=[10,2,30,40]
print(lst)
lst.pop()
print(lst)

lst=[10,2,30,40]
for i in lst:
  lst.pop()
  print(lst)
```

```
[10, 2, 30, 40]
[10, 2, 30]
[10, 2, 30]
[10, 2]
```

[ ]:
```
# clear()

lst=[10,2,30,40]
print(len(lst), lst)
lst.clear()
print(len(lst), lst)
```

```
4 [10, 2, 30, 40]
0 []
```

[ ]:
```
# index by puting the values it will show you the index values
lst=[10,2,30,40]
print(lst)

print(lst.index(10))
print(lst.index(2))
print(lst.index(30))
```

```
[10, 2, 30, 40]
0
1
2
```

[ ]:
```
# count it will check the occuracnce of any values in list
example_list = [1, 2, 3, 4, 5,'raj', 3+3j, 34.67, 'python', ['MH',␣
↪"AP"],3,4,5,3,3,3,3]
print(example_list.count(3))

lst1='Prashant sundge'
lst=list(lst1)
print(lst)
print(lst.count('a'))
print(lst.count('s'))
```

```
6
['P', 'r', 'a', 's', 'h', 'a', 'n', 't', ' ', 's', 'u', 'n', 'd', 'g', 'e']
2
2
```

[ ]:
```
# reverse

lst12=[1, 2, 3, 4, 5]
print(lst12)
lst12.reverse()
print(lst12)
l1=[10,20,15,12,-56,78,4]
l1.reverse()
print(l1)
```

```
[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]
[4, 78, -56, 12, 15, 20, 10]
```

[ ]:
```
#sort this will sort the values
l1=[10,20,15,12,-56,78,4]
```

```
print("Normal ", l1)
l1.sort()
print("After sort :", l1)
```

```
Normal  [10, 20, 15, 12, -56, 78, 4]
After sort : [-56, 4, 10, 12, 15, 20, 78]
```

[ ]:
```
#extend
l1=[10,20,15,12,-56,78,4]
print( l1)
print(len(l1))
l1.extend([1, 2, 3, 4, 5])
print("Extend : ", l1)
print(len(l1))
l1.append([1, 2, 3, 4, 5])
print("Append : ", l1)
print(len(l1))
```

```
[10, 20, 15, 12, -56, 78, 4]
7
Extend :  [10, 20, 15, 12, -56, 78, 4, 1, 2, 3, 4, 5]
12
Append :  [10, 20, 15, 12, -56, 78, 4, 1, 2, 3, 4, 5, [1, 2, 3, 4, 5]]
13
```

16 Tuple()

- tuple is one of the pre defined class and treated as list data type
- The Purpose of tuple data type is that to store multiple values either of same type or different type or both the types in single object with unique and duplicate values
- an obejct in tuple maintains insertion order
- an object in tuple perform index and slicing
- tuple is immutable
- syntax varname=(val1,val2,val3)

[ ]:
```
t1=(10,20,30,40,50)
print(type(t1), t1)
print(t1[0])
print(t1[0:3])
print(t1[3:1: -1])
```

```
<class 'tuple'> (10, 20, 30, 40, 50)
10
(10, 20, 30)
(40, 30)
```

17 Pre-Defined Function in tuple

- index()
- count()

```
[ ]: t1=(10,20,30,40,50,20,20,20)
     print(t1.index(20))
     print(t1.count(20))
```

```
1
4
```

```
[ ]: tp=('Prashant', 'Ranjana', 'Rajesh')
     name, name2 , name3 = tp
     print(name)
     #print(name2)
     print(name3)
```

```
Prashant
Rajesh
```

18 set()

- set is one of the pre-defined class and treated as set data type.
- set stores only unique values no duplicate values are stored in set
- an object set never maintains insertion order
- no indexing and slicing in set
- an object of set belongs to both Mutable and Immutable
  - mutable in case of add()
  - immutable in case of item assignments

Pre_defined Functions in Set
- 1 add() - 2 clear() - 3 remove() - 4 discard() - 5 pop() - 7 copy() - 8 update()

- isdisjoint()
- issuperset()
- issubset()
- union()
- intersection()
- difference()
- symmetric_difference()

```
[ ]: #set
     # st[1]  op : TypeError: 'set' object is not subscriptable
     st = {10,20,30,40,30,20,40,50,40,30,20,10}
     print(type(st), len(st), st)
     # st[0] = 200  # TypeError: 'set' object does not support item assignment
```

```
<class 'set'> 5 {50, 20, 40, 10, 30}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-193-deb130683d59> in <cell line: 5>()
      3 st = {10,20,30,40,30,20,40,50,40,30,20,10}
      4 print(type(st), len(st), st)
----> 5 st[0] = 200

TypeError: 'set' object does not support item assignment
```

```
[ ]: # add
     st = {10,20,30,40}
     print(type(st), len(st), st)
     st.add(100)
     print(type(st), len(st), st)
     st.add('pams')
     print(type(st), len(st), st)
```

```
<class 'set'> 4 {40, 10, 20, 30}
<class 'set'> 5 {100, 40, 10, 20, 30}
<class 'set'> 6 {100, 40, 10, 'pams', 20, 30}
```

```
[ ]: #clear()
     st = {10,20,30,40}
     print(type(st), len(st), st)
     st.clear()
     print(type(st), len(st), st)
```

```
<class 'set'> 4 {40, 10, 20, 30}
<class 'set'> 0 set()
```

```
[ ]: # remove()
     st = {10,20,30,40}
     print(type(st), len(st), st)
     st.remove(40)
     print(type(st), len(st), st)
     #st.remove(100) #KeyError: 100
     print(type(st), len(st), st)
```

```
<class 'set'> 4 {40, 10, 20, 30}
```

```
<class 'set'> 3 {10, 20, 30}
<class 'set'> 3 {10, 20, 30}
```

[ ]:
```
#discard()
st = {10,20,30,40}
print(type(st), len(st), st)
st.discard(40)
print(type(st), len(st), st)
st.discard(1000)  # no key error
#st.remove(100) # KeyError: 100
print(type(st), len(st), st)
```

```
<class 'set'> 4 {40, 10, 20, 30}
<class 'set'> 3 {10, 20, 30}
<class 'set'> 3 {10, 20, 30}
```

[ ]:
```
# pop()
st = {10,20,30,40}
print(type(st), len(st), st)
st.pop()
print(type(st), len(st), st)
st.pop()
print(type(st), len(st), st)
```

```
<class 'set'> 4 {40, 10, 20, 30}
<class 'set'> 3 {10, 20, 30}
<class 'set'> 2 {20, 30}
```

[ ]:
```
#update
st1={10,20,30}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
st1.update(st2)
print(type(st1), len(st1), st1)
```

```
<class 'set'> 3 {10, 20, 30}
<class 'set'> 2 {40, 50}
<class 'set'> 5 {50, 20, 40, 10, 30}
```

isdisjoint - the function returns True if no common elements in both setobj1 and setobj2 - syntax setobj1.isdisjoint(setobj2)

[ ]:
```
#isdisjoint
st1={10,20,30}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
```

```
st1.isdisjoint(st2)
```

```
<class 'set'> 3 {10, 20, 30}
<class 'set'> 2 {40, 50}
```

```
[ ]: True
```

issuperset - This Function returns True provided setobj1 contains all the elements of setobj2 - syntax setobje.issuperset(steobk2)

```
[ ]: #issuperset
st1={10,20,30}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
print(st1.issuperset(st2))
st1={10,20,30, 40,50}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
print(st1.issuperset(st2))
```

```
<class 'set'> 3 {10, 20, 30}
<class 'set'> 2 {40, 50}
False
<class 'set'> 5 {50, 20, 40, 10, 30}
<class 'set'> 2 {40, 50}
True
```

issubset - this function returns True provided setobj2 contains all the elements of setobj1 - syntax setobje.issubset(steobk2)

```
[ ]: # issubset
st1={10,20,30}
print(type(st1), len(st1), st1)
st2={40,50,10,20,30}
print(type(st2), len(st2), st2)
print(st1.issubset(st2))
st1={10,20,30, 40,50}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
print(st1.issubset(st2))
```

```
<class 'set'> 3 {10, 20, 30}
<class 'set'> 5 {50, 20, 40, 10, 30}
True
<class 'set'> 5 {50, 20, 40, 10, 30}
```

```
<class 'set'> 2 {40, 50}
False
```

union - this function takes all the unique elements of setobj1 and setobj2 and result placed in setobj3

- syntax setobj3=setobj1.union(setobj2)

[ ]:
```
# union
st1={10,20,30}
print(type(st1), len(st1), st1)
st2={40,50,30}
print(type(st2), len(st2), st2)
st3=st1.union(st2)
print(st3)
st1={10,20,30, 40,50}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
print(st1.union(st2))
```

```
<class 'set'> 3 {10, 20, 30}
<class 'set'> 3 {40, 50, 30}
{50, 20, 40, 10, 30}
<class 'set'> 5 {50, 20, 40, 10, 30}
<class 'set'> 2 {40, 50}
{50, 20, 40, 10, 30}
```

intersection - this function takes all the unique elements of setobj1 and setobj2 and result placed in setobj3

- syntax setobj3=setobj1.intersection(setobj2)

[ ]:
```
# intersection
st1={10,20,30}
print(type(st1), len(st1), st1)
st2={40,50,30}
print(type(st2), len(st2), st2)
st3=st1.intersection(st2)
print(st3)
st1={10,20,30, 40,50}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
print(st1.intersection(st2))
```

```
<class 'set'> 3 {10, 20, 30}
<class 'set'> 3 {40, 50, 30}
{30}
<class 'set'> 5 {50, 20, 40, 10, 30}
```

```
<class 'set'> 2 {40, 50}
{40, 50}
```

difference()

- this function removes common elements from setobj1 and setobj2 and takes the remaining elements of setobj1 and result placed in setobj3
- syntax setobj3=setobj2.difference(setobj1)

[ ]:
```
# difference
st1={10,20,30,40}
print(type(st1), len(st1), st1)
st2={40,50,30}
print(type(st2), len(st2), st2)
st3=st1.difference(st2)
print(st3)
st1={10,20,30, 40,50}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
print(st1.difference(st2))
```

```
<class 'set'> 4 {40, 10, 20, 30}
<class 'set'> 3 {40, 50, 30}
{10, 20}
<class 'set'> 5 {50, 20, 40, 10, 30}
<class 'set'> 2 {40, 50}
{10, 20, 30}
```

symmetric_difference()

- this function removes common elements from setobj1 and setobj2 and takes the remaining elements of setobj1 and result placed in setobj3
- syntax setobj3=setobj2.symmetric_difference(setobj1)

[ ]:
```
# symmetric_difference
st1={10,20,30,40}
print(type(st1), len(st1), st1)
st2={40,50,30}
print(type(st2), len(st2), st2)
st3=st1.symmetric_difference(st2)
print(st3)
st1={10,20,30, 40,50}
print(type(st1), len(st1), st1)
st2={40,50}
print(type(st2), len(st2), st2)
print(st1.symmetric_difference(st2))
```

```
<class 'set'> 4 {40, 10, 20, 30}
<class 'set'> 3 {40, 50, 30}
```

```
{10, 50, 20}
<class 'set'> 5 {50, 20, 40, 10, 30}
<class 'set'> 2 {40, 50}
{10, 20, 30}
```

18.1 Questions

1) Find all the players who are playing all the games.–union()
2) Find all the players who are playing both Cricket and Tennis–intersection()
3) Find all the players who are playing Only Cricket but not tennis—difference()
4) Find all the players who are playing Only Tennis but not cricket—difference()
5) Find all the players who are EXCLUSIVELY playing Tennis and cricket—symmetric_difference()

[ ]:
```
tenis_player={'Sachin', 'Rahul', 'irfan', 'virat','surya'}
cricket_player={'Sachin', 'shivani', 'isha', 'virat','priya'}
```

[ ]:
```
all_playing_games =tenis_player.union(cricket_player)
print(f"Players are playing all the games {all_playing_games}")
```

```
Players are playing all the games {'Sachin', 'surya', 'virat', 'isha', 'irfan',
'priya', 'Rahul', 'shivani'}
```

[ ]:
```
both_playing= tenis_player.intersection(cricket_player)
print(f"Both Criket and tennis players {both_playing}")
```

```
Both Criket and tennis players {'Sachin', 'virat'}
```

[ ]:
```
only_cricket_playing= tenis_player.difference(cricket_player)
print(f"Only  Criket  players {only_cricket_playing}")
```

```
Only  Criket  players {'irfan', 'Rahul', 'surya'}
```

[ ]:
```
only_tenis_playing= cricket_player.difference(tenis_player)
print(f"Only  tennis  players {only_tenis_playing}")
```

```
Only  tennis  players {'isha', 'priya', 'shivani'}
```

[ ]:
```
exclosively_playing_cricket_tenis=tenis_player.
  ↪symmetric_difference(cricket_player)
print(f"Exclosively playing {exclosively_playing_cricket_tenis}")
```

```
Exclosively playing {'surya', 'irfan', 'isha', 'priya', 'Rahul', 'shivani'}
```

19 Dict

- The purpose of dict data type is to store *KEY, VALUE* in single variable

- in (*key* , *value*) the key is unique and values of value may or may not be unique
- An object of dict does not support indexing and slicking because values and key itself considered as indices
- in dict object values of *key* are treated as immutable and values of *value* are treated as mutable
- syntax dictobje = {} or dictobje=dict()

Pre_defined Functions in Dict{}

- 1. clear()
- 2. copy()
- 3. pop()
- 4. popitem()
- 5. keys()
- 6. values()
- 7. get()
- 8. items()
- 9. update()

[ ]:
```
dct={}
print(type(dct))
dct=dict()
print(type(dct))
d1={10:"Python",20:"Data Sci",30:"Django"}
print(type(d1), d1)
# print(d1[0])  #KeyError: 0
print(type(d1), d1[10])
print(type(d1), d1[30])
d1[40] ='mysql'
print(type(d1), d1)
```

```
<class 'dict'>
<class 'dict'>
<class 'dict'> {10: 'Python', 20: 'Data Sci', 30: 'Django'}
<class 'dict'> Python
<class 'dict'> Django
<class 'dict'> {10: 'Python', 20: 'Data Sci', 30: 'Django', 40: 'mysql'}
```

[ ]:
```
# clear
d1={10:"Python",20:"Data Sci",30:"Django"}
print(type(d1), len(d1),d1)
d1.clear()
print(type(d1), len(d1),d1)
print('-'*50)
# copy
d1={10:"Python",20:"Data Sci",30:"Django"}
```

```
print(type(d1), len(d1),d1)
d2 = d1.copy()
print(type(d2), len(d2),d2)
print('-'*50)
# pop() it removes values by providing keys
d1={10:"Python",20:"Data Sci",30:"Django"}
print(type(d1), len(d1),d1)
d1.pop(10)
print(type(d1), len(d1),d1)
d1.pop(30)
print(type(d1), len(d1),d1)
print('-'*50)

# popitem() removes last key value pair
d1={10:"Python",20:"Data Sci",30:"Django"}
print(type(d1), len(d1),d1)
d1.popitem()
print(type(d1), len(d1),d1)
d1.popitem()
print(type(d1), len(d1),d1)
print('-'*50)
# keys() this is used to print all keys
d1={10:"Python",20:"Data Sci",30:"Django"}
print(type(d1), len(d1),d1)
print(type(d1), len(d1),d1.keys())
print(type(d1), len(d1),d1.keys())
print('-'*50)

# values() this is used to print all values
d1={10:"Python",20:"Data Sci",30:"Django"}
print(type(d1), len(d1),d1)
print(type(d1), len(d1),d1.values())
print(type(d1), len(d1),d1.values())
print('-'*50)


# get() this is used to print values by giving keys
d1={10:"Python",20:"Data Sci",30:"Django"}
print(type(d1), len(d1),d1)
print(type(d1), len(d1),d1.get(10))
print(type(d1), len(d1),d1.get(20))
print('-'*50)


#items() this is used to print key & values
d1={10:"Python",20:"Data Sci",30:"Django"}
print(d1.items())
```

```python
for k,v in d1.items():
  print(f"{k} -- {v}")
print('-'*50)

#update() this is used to print key & values
d1={10:"Python",20:"Data Sci",30:"Django"}
d2={100:'prashant', 200:'Ranju',300:'nandu'}
d1.update(d2)
print(d1)
```

```
<class 'dict'> 3 {10: 'Python', 20: 'Data Sci', 30: 'Django'}
<class 'dict'> 0 {}
--------------------------------------------------
<class 'dict'> 3 {10: 'Python', 20: 'Data Sci', 30: 'Django'}
<class 'dict'> 3 {10: 'Python', 20: 'Data Sci', 30: 'Django'}
--------------------------------------------------
<class 'dict'> 3 {10: 'Python', 20: 'Data Sci', 30: 'Django'}
<class 'dict'> 2 {20: 'Data Sci', 30: 'Django'}
<class 'dict'> 1 {20: 'Data Sci'}
--------------------------------------------------
<class 'dict'> 3 {10: 'Python', 20: 'Data Sci', 30: 'Django'}
<class 'dict'> 2 {10: 'Python', 20: 'Data Sci'}
<class 'dict'> 1 {10: 'Python'}
--------------------------------------------------
<class 'dict'> 3 {10: 'Python', 20: 'Data Sci', 30: 'Django'}
<class 'dict'> 3 dict_keys([10, 20, 30])
<class 'dict'> 3 dict_keys([10, 20, 30])
--------------------------------------------------
<class 'dict'> 3 {10: 'Python', 20: 'Data Sci', 30: 'Django'}
<class 'dict'> 3 dict_values(['Python', 'Data Sci', 'Django'])
<class 'dict'> 3 dict_values(['Python', 'Data Sci', 'Django'])
--------------------------------------------------
<class 'dict'> 3 {10: 'Python', 20: 'Data Sci', 30: 'Django'}
<class 'dict'> 3 Python
<class 'dict'> 3 Data Sci
--------------------------------------------------
dict_items([(10, 'Python'), (20, 'Data Sci'), (30, 'Django')])
10 -- Python
20 -- Data Sci
30 -- Django
--------------------------------------------------
{10: 'Python', 20: 'Data Sci', 30: 'Django', 100: 'prashant', 200: 'Ranju', 300:
'nandu'}
```

20 *Special Points: (Dict in Dict–Nested Dict)*

```
[ ]: d1={"CRS1":{1:"Python",2:"ML",3:"AI"},"CRS2":{10:"JAVA",20:"Servlets"},"CRS3":
      ↪{1:"Oracle",2:"MySQL",3:"MongoDB",4:"DB2"}}
     print(d1.get('CRS1'))
     for k,v in d1.items():
       print(f"{k} --{v}" ,type(v))
```

```
{1: 'Python', 2: 'ML', 3: 'AI'}
CRS1 --{1: 'Python', 2: 'ML', 3: 'AI'} <class 'dict'>
CRS2 --{10: 'JAVA', 20: 'Servlets'} <class 'dict'>
CRS3 --{1: 'Oracle', 2: 'MySQL', 3: 'MongoDB', 4: 'DB2'} <class 'dict'>
```

21 *List, Tuple and set in dict*

```
[ ]: d1={"CRS1":["PYTHON","JAVA","R"],"CODES":(10,20,30),"CC":{"Py12","Ja45","r45"}}
     for k,v in d1.items():
       print(f"{k} --{v}" ,type(v))
```

```
CRS1 --['PYTHON', 'JAVA', 'R'] <class 'list'>
CODES --(10, 20, 30) <class 'tuple'>
CC --{'r45', 'Py12', 'Ja45'} <class 'set'>
```

```
[ ]: # dict in list
     l1=[10,{"Python":"RS","Numpy":"TV"},20,30]
     for k,v in l1[1].items():
       print(f"{k} --{v}" ,type(v))

     # dict in tuple
     t1=(10,{"Python":"RS","Numpy":"TV"},20,30)
     print(type(t1))
     for k,v in t1[1].items():
       print(f"{k} --{v}" ,type(v))

     # set in dict
     # s1={10,{"Python":"RS","Numpy":"TV","Pandas":"MC"},20,30,40}
     # print(type(s1))  #TypeError: unhashable type: 'dict'
     # print(s1)  #TypeError: unhashable type: 'dict'
```

```
Python --RS <class 'str'>
Numpy --TV <class 'str'>
<class 'tuple'>
Python --RS <class 'str'>
Numpy --TV <class 'str'>
```

22 displaying the result of program on the consol

- To display the result of the python program on the console, we use pre-defined fucntion called print()
- print can be used in following syntax
  - print(val1,val2…val-n)
  - print(msg1, msg2,…msg-n)
  - print(message cum values)
  - print(message cum values with format())
  - print(message cum values with format specifiers)
  - print(value, end=)

[ ]:
```
#Syntax-1: print(val1)
a=10
print(a)
b='python'
print(b)
print('-'*50)
#2  Syntax-2: print(message1)
print("Hellp Python world")
print('Hellp Python world')
print('''Hellp Python world
          Welcome to the
      world of Programing ''')
print("""Hellp Python world
          Welcome to the
      world of Programing """)
print("Hello" + "Python" + "World")
print("Hello "+"Python "+"World ")
#print("Hello "+10) #TypeError: can only concatenate str (not "int") to str
print("Hello" + '10')
print('-'*50)
# 3  Syntax-3: print(Messages cum values)
a=10
b=20
print("value of a ", a)
print("value of b ", b)
print("sum of a , b ", a+b)
#4 Syntax-4 : print(Messages Cum Values with format() )
print('-'*50)
sno=20
sname="Rossum"
marks=33.33
print("Roll Number={} Name={} and Marks={}".format(sno,sname,marks))
print("Roll Number={} Name='{}' and Marks={}".format(sno,sname,marks))

print('-'*50)
#Syntax-5: print(Messages Cum Values with format specifiers)
```

```
b=20
c=a+b
print("sum=%d" %c)
#sum of 10 and 20=30
print("sum of %d and %d=%d" %(a,b,c))
sno=10
sname="Rossum"
marks=23.45
print("Roll Number:%d name:%s and Marks:%f" %(sno,sname,marks))
print("Roll Number:%d name:'%s' and Marks:%0.2f" %(sno,sname,marks))
print("Roll Number:%d name:'%s' and Marks:%0.3f" %(sno,sname,marks))
print('-'*50)


#Syntax-6: print(Value, end= )

r=range(1,6)
for i in r:
  print(i , end="\t")
print('\n')

for i in r:
  print(i, end ='')
print('\n')
print('-'*50)
for i in r:
  print(i, end ='...')
print('\n')
print('-'*50)
for i in r:
    print(i, end ='--')
print('\n')
print('-'*50)
```

```
10
python
--------------------------------------------------
Hellp Python world
Hellp Python world
Hellp Python world
          Welcome to the
      world of Programing
Hellp Python world
          Welcome to the
      world of Programing
HelloPythonWorld
```

```
Hello Python World
Hello10
--------------------------------------------------
value of a  10
value of b  20
sum of a , b  30
--------------------------------------------------
Roll Number=20 Name=Rossum and Marks=33.33
Roll Number=20 Name='Rossum' and Marks=33.33
--------------------------------------------------
sum=30
sum of 10 and 20=30
Roll Number:10 name:Rossum and Marks:23.450000
Roll Number:10 name:'Rossum' and Marks:23.45
Roll Number:10 name:'Rossum' and Marks:23.450
--------------------------------------------------
1       2       3       4       5

12345

--------------------------------------------------
1…2…3…4…5…

--------------------------------------------------
1--2--3--4--5--

--------------------------------------------------
```

23 Reading the Data from Keyboard

- input()
- This fucntion is used for reading any type of data from keyboard in the form or str and that values placed in left hand side variable name
- input(message)
  - This funciton is used for reading any type of data from keyboard in the form of str ant that values are placed in left hand side variable name
  - this funciton additionlly gives user prompting messages

[ ]:
```python
print("Enter any values ")
val=input()
print(val)
```

```
Enter any values
125
125
```

[ ]:
```
var=input("Enter any values ")
print(var)
```

```
Enter any values 1286
1286
```

24 PROGRAMS

[ ]:
```
# program substracting of two number
print("Enter a value ")
a=input()
print("Enter  b value")
b=input()
print(type(a)) # str type
print(type(a)) # str type
# type casting
print("type casting")
A=int(a)
B=int(b)
print(type(A)) # str type
print(type(B)) # str type
C=A-B
print("Substracted value is : ", C)
```

```
Enter a value
1258
Enter  b value
458
<class 'str'>
<class 'str'>
type casting
<class 'int'>
<class 'int'>
Substracted value is :  800
```

[ ]:
```
# program for cal area of rectangle
print("Enter  Height ")
a=input()
print("Enter  Width")
b=input()
print(type(a)) # str type
print(type(a)) # str type
# type casting
print("type casting")
A=int(a)
B=int(b)
print(type(A)) # str type
print(type(B)) # str type
C=A*B
print("Area of Rectangle is : ", C)
```

```
Enter  Height
125
Enter  Width
50
<class 'str'>
<class 'str'>
type casting
<class 'int'>
<class 'int'>
Substracted value is :  6250
```

[ ]:
```
# accept 2 values from input and multiply
print("Enter  Value for A ")
a=input()
print("Enter  Value for B")
b=input()
print(type(a)) # str type
print(type(a)) # str type
# type casting
print("type casting")
A=int(a)
B=int(b)
print(type(A)) # str type
print(type(B)) # str type
C=A*B
print("Multiplication : ", C)
```

```
Enter  Value for A
12
Enter  Value for B
5
<class 'str'>
<class 'str'>
type casting
<class 'int'>
<class 'int'>
Multiplication :  60
```

[ ]:
```
# accept 2 values from input and multiply
print("Enter  Value for A ")
a=float(input())
print("Enter  Value for B")
b=float(input())
```

```
print(type(a)) # str type
print(type(a)) # str type
C=a*b
print("Multiplication : ", C)
```

```
Enter  Value for A
10.5
Enter  Value for B
56.3
<class 'float'>
<class 'float'>
Multiplication :  591.15
```

[ ]:
```
# accept 2 values from input and multiply
print("Enter 2 values")
c= float(input()) * float(input())
print(c)
```

```
Enter 2 values
12
6
72.0
```

[ ]:
```
print(" Mul = {}".format(float(input('Enter a value'))*float(input("enter␣
 ↪second value"))))
```

```
Enter a value12
5
 Mul =60.0
```

25 Operators and Expressions

- an Operator is one of the symbol and it is used for performing certain operation on data
- an expression is a collection of objects / variables connected with operator
- in otherwards if collection of object / variables connected with an operator then it called as expression
- types of operator
  - Arithmatic Operators
  - Assignment Operator
  - Relational Operator(comparision)
  - Logical operators (comparision)
  - Bitwise Operators (MOST IMP)
  - Membership Operator
    * in
    * not in
  - Identity Operators
    * is

* is not

Notes

– shorthand opertors in python
– pre-increment / pre-decrement are not present in python
– ternary operator(? :) present in c, c++ , java not present in pyhton
– python programming contains its own ternary operators called *if … else* operator

26 Arithmatic operators

- The purpose of Arithmetic Operators is that "To Perform Arithemtic Operations such as Addition, Substraction, Multiplication..etc".
- If Collection of Variables / Objects connected with Arithmetic Operators then It is called Arithmetic Expression.
- In Python Programming, we have 7 Arithmetic Operators. They are given in the following table

SL NO	SYMBOL	MEANING	EXAMPLE
1.	'+'	Addition	print(10+20)—->30
2.	'-'	Subtraction	print(10-20)——> -10
3.	[*]	Multiplication	print(10*20)——> 200
4.	/	Division	print(10/3)—->3.333333333335 (Float Quotient)
5.	//	Floor Division	print(10//3)—>3 (Integer Quotient)
6.	%	Modulo Division	print(10%3)—>1 (remainder)
7.	[**]	Exponentiation	print(10**3)—>1000 (Power), print(10**-3)–>0.001

[ ]:
```
a=10
b=20
c=a+b
print(" Addition ", c)
c= a-b
print(" Substraction ", c)
c= a*b
print(" Multiplication ", c)
c= a/b
print(" division ", c)
c= a//b
print(" Floor division ", c)
c= a % b
print(" Modulo division ", c)
c= a ** b
print(" Exponention  ", c)
```

```
Addition  30
 Substraction  -10
 Multiplication  200
 division  0.5
```

```
Floor division  0
Modulo division  10
Exponention   100000000000000000000
```

27 Assignment Operator

- single line assignment
- multi line assignment

  LHS = RHS

```
[ ]: # single line
a=10
b=20
c=a+b
print(a,b,c)

# multi line
a, b, c = 10, 20, 10+20
print(a,b,c)
c, d, e = a+b, a-b, a*b
print(c, d,e)
```

```
10 20 30
10 20 30
30 -10 200
```

```
[ ]: # swaping any type of values

a= input("enter the A value")
b = input ("enter the B value")

print("Value A BEfore Swap",a)
print("Value B Before Swap",b)
c = a
a = b
b = c
print("Value A after Swap",a)
print("Value B after Swap",b)
```

```
enter the A value15
enter the B value25
Value A BEfore Swap 15
Value B Before Swap 25
Value A after Swap 25
Value B after Swap 15
```

```
[ ]: # swaping any type of values

     a= input("enter the A value")
     b = input ("enter the B value")

     print("Value A BEfore Swap",a)
     print("Value B Before Swap",b)
     a, b = b, a
     print("Value A after Swap",a)
     print("Value B after Swap",b)
```

```
enter the A value25
enter the B value35
Value A BEfore Swap 25
Value B Before Swap 35
Value A after Swap 35
Value B after Swap 25
```

```
[ ]: # swaping any type of values

     a= int(input("enter the A value"))
     b = int(input ("enter the B value"))

     print("Value A BEfore Swap",a)
     print("Value B Before Swap",b)

     a = a + b
     b = a - b
     a = a - b
     print("Value A after Swap",a)
     print("Value B after Swap",b)
```

```
enter the A value66
enter the B value33
Value A BEfore Swap 66
Value B Before Swap 33
Value A after Swap 33
Value B after Swap 66
```

28 Relational Operators (Comparision)

```
[ ]: a= 10
     b= 20

     print(a==b)
     print(a > b)
     print(a < b)
     print(a != b)
     print(a >= b)
     print(a <= b)
```

```
False
False
True
True
False
True
```

[ ]:
```
a=int(input("Enter A value"))
b = int(input("Enter B value"))
c= int(input("Enter C value"))

if a > b  and a > c :
  print("A is Max ",a)
elif b > a and b > c:
  print("B is Max ",b)
else :
  print('C is Max',c)
```

```
Enter A value35
Enter B value100
Enter C value5
B is Max  100
```

29 Bitwise Operator (Most IMP)

- The purpose of Bitwise operators to perform operations on integer data in the form of bits
- Bitwise operators can applied only on integer data but not floating point values because integer values provides certainity where floating point values are not able to porivide certainity
- Bitwise operators first integer values into binary bits and perform the operation based on type of bitwise
- Since these opertots performs the operation in the form of bit by bit hence they named as bitwise
- type of Bitwise opertors
  - Bitwise Left Shift operator (<<)
  - Bitwise Right Shift Operator (>>)
  - Bitwise OR Opertor (|)
  - Bitwise AND operator (&)
  - Bitiwse Complement operator (~)
  - Bitwise XOR operator (^)

30 1. Bitwise Left shift operator (<<)

```
[ ]: a = 10
     b = 3
     c= a<<b
     print(c)
     print(3<<2)
     print(4<<1)
```

```
80
12
8
```

31 2. Bitwise Right shift operator (>>)

```
[ ]: a = 10
     b = 3
     c= a>>b
     print(c)
     print(3>>2)
     print(4>>1)
```

```
1
0
2
```

32 3. Bitwise OR opertor (|)

Varname1	VarName2	VarName1	VarName-2
1	0	1	
0	1	1	
0	0	0	
1	1	1	

```
[ ]: a=4
     print(bin(a), a)
     b=7
     print(bin(b), b)
     c=a|b
     print(bin(c)   , c)

     a=15
     print(bin(a), a)
     b=4
```

```
print(bin(b), b)
c=a|b
print(bin(c)  , c)
```

```
0b100 4
0b111 7
0b111 7
0b1111 15
0b100 4
0b1111 15
```

4. Bitwise AND operator (&)

Varname1	VarName2	VarName1 & VarName-2
1	0	0
0	1	0
0	0	0
1	1	1

[ ]:
```
a=4
print(bin(a), a)
b=7
print(bin(b), b)
c=a & b
print(bin(c)  , c)

a=15
print(bin(a), a)
b=4
print(bin(b), b)
c= a & b
print(bin(c)  , c)
```

```
0b100 4
0b111 7
0b100 4
0b1111 15
0b100 4
0b100 4
```

33 5. Biwise Complement Operator (~)

- Bitwise complement opertor internally inverted the bits of given number
- n= 4 will convert X=~n – 0100 inverting the above bits – 1011

```
[ ]: a = 4
     print(bin(a), a)
     x = ~ a
     print(bin(x), x)
     a = 16
     print(bin(a), a)
     x = ~ a
     print(bin(x), x)
```

```
0b100 4
-0b101 -5
0b10000 16
-0b10001 -17
```

6. Bitwise XOR opertor (^)

Varname1	VarName2	VarName1 ^ VarName-2
1	0	1
0	1	1
0	0	0
1	1	0

```
[ ]: a=4
     print(bin(a), a)
     b=7
     print(bin(b), b)
     c=a ^ b
     print(bin(c)  , c)

     a=15
     print(bin(a), a)
     b=4
     print(bin(b), b)
     c= a ^ b
     print(bin(c)  , c)
```

```
0b100 4
0b111 7
0b11 3
0b1111 15
0b100 4
0b1011 11
```

34 Membership Operators

- The purpose of membership operator is that to check the existance of a perticular value in an iterable object.

- An iterable object is one which contains multiple values (Sequence types , list , set, dict)
- types
  - in operator
  - not in operator

```
[ ]: # in opertor
s='python'
print(type(s), s)
print('y' in s)
print('o' in s)
print('X' in s)
print('yth' in s)

# not in

s='python'
print(type(s), s)
print('y' not in s)
print('o' not in s)
print('X' not in s)
print('yth' not in s)
```

```
<class 'str'> python
True
True
False
True
<class 'str'> python
False
False
True
False
```

35 Identity Operators

- the purpose of identity operators is to check the memory address of two objects
- in python programming we have 2 types of identity opertors
  - is opertor
    - ' is ' operator returns True provided both obj1 and obj2 must contains same address
    - ' is ' operator returns False provided both obj1 and obj2 must contains different address
  - is not operator
  - ' is not ' operator returns True provided both obj1 and obj2 must contains different address
  - ' is not ' operator returns False provided both obj1 and obj2 must contains same address

[ ]:
```python
a= 6
b= 6
print("id of A ", id(a))
print("id of B ", id(b))
print(a is b)
a= 6
b= 8
print("id of A ", id(a))
print("id of B ", id(b))
print(a is b)
a= 6
b= 8
print("id of A ", id(a))
print("id of B ", id(b))
print(a is not b)

a= 6
b= a
print("id of A ", id(a))
print("id of B ", id(b))
print(a is b)
```

```
id of A  139182511292816
id of B  139182511292816
True
id of A  139182511292816
id of B  139182511292880
False
id of A  139182511292816
id of B  139182511292880
True
id of A  139182511292816
id of B  139182511292816
True
```

36 Ternary Operator in python

37 if else

[ ]:
```python
# program for finding biggest of two number by using python ternary operator

a= int(input("Enter A number "))
b= int(input("Enter B number "))
```

```
if a > b:
  print("A is Biggest number ", a)
else:
  print("B is Biggest number ", b)
```

```
Enter A number 25
Enter B number 30
B is Biggest number  30
```

[ ]:
```
# program for finding biggest of two number by using python ternary operator

a= int(input("Enter A number "))
b= int(input("Enter B number "))

big= a if a > b else b
print(big)
```

```
Enter A number 25
Enter B number 15
25
```

[ ]:
```
# program for finding biggest of two number and check for equality by using␣
 ↪python ternary operator

a = int(input('Enter A number'))
B = int(input('Enter B number'))

big = a if a > b else b if b > a else 'Both are same values'
print('big ( {}, {} = {} )'.format(a,b,big))
```

```
Enter A number50
Enter B number50
big ( 50, 15 = 50 )
```

[ ]:
```
#program for accepting a word and decide whether the First and last letters are␣
 ↪same or not
#FirstLastCharsWord.py
word=input("Enter a word")

if word[0] == word[-1]:
  print("Word first letter {} and last letter  {} are same {}".format(word[0],␣
 ↪word[-1], word))
else:
  print("Word first letter {} and last letter  {} are not same {}".
 ↪format(word[0], word[-1], word))
```

```
res="first letter and last letter are same " if word[0] == word[-1] else 'Words␣
  ↪are not same'
print(res)
```

```
Enter a wordpYHTON
Word first letter p and last letter  N are not same pYHTON
Words are not same
```

[ ]:
```
#program for accepting a word and decide whether it is palindrome or not
#palindromeex1.py
# MOM

word = input("enter word to check palindrom")

res= "Palindrom " if word == word[::-1] else 'not palindrom'
print(res, word)
```

```
enter word to check palindromMOM
Palindrom  MOM
```

[ ]:
```
#PROGRAM FOR ACCEPTING A WORD AND CHECH WHETHER IT CONTAINS VOWELS ARE NOT
#VowelCheck.py
word = input("enter word to check palindrom")

res = 'Vovels present' if 'a' in word or 'e' in word or 'i' in word or 'o' in␣
  ↪word or 'u' in word else 'NO Vovels'
print(res, word)
```

```
enter word to check palindromabc
Vovels present abc
```

38 TYPES OF FLOW CONTROL STATEMENTS IN PYTHON

- **A. Condtional Or Selection Or Branching Statements**
  - 1. Simple if statement
  - 2. if else statement
  - 3. if elif else statement
  - 4. match case statement
- **B. Looping or Iterative or repetative statements**
  - 1. while loop or while..else loop
  - 2. for loop or for else loop
- **C. Transfer flow control statements**
  - 1. break
  - 2. continue
  - 3. pass
  - 4. return

- **D. inner Or nested loops**
  - 1. for loop in for loop
  - 2. while loop in while loop
  - 3. for loop in while loop
  - 4. while loop in for loop

39 A. Condtional Or Selection Or Branching Statements

39.1 Simple if statement

[ ]:
```
#moviee.py

tkt=input('Do you have ticket (Yes/No)')
if tkt == 'yes':
  print("Will go to Movie")
print("no Ticket sorry ")
```

```
Do you have ticket (Yes/No)no
no Ticket sorry
```

[ ]:
```
#posnegzero
num=int(input("enter number to check "))
if num < 0:
  print('Number is negative',num)
if num > 0:
  print('Number is positive ',num)
if num == 0:
  print('Num is zero', num)
print('Program execution completed')
```

```
enter number to check 0
Num is zero 0
Program execution completed
```

[ ]:
```
#langNamesif.py
langdict={"C":"Ritche","C++":"SUP","PYTHON":["RAVALI","ROSSUM","BHARAT"],"JAVA":
 ↪"GOSLING"}
lang=input("Enter your fevourate language ")
if lang in langdict:
  print("Favourate lang developed by ", lang, langdict.get(lang))
else:
  print('Lang is other {}'.format(lang))
```

```
Enter your fevourate language marathi
Lang is other marathi
```

40 if else statement

[ ]:
```
#dsEx1.py
d= int(input('enter d'))
if d == 0:
  print('Zero', d)
else:
  if d==1:
    print(d)
  else:
    if d==2 :
      print(d)
    else:
      if d == 3:
        print(d)
      else:
        if d == 4:
          print(d)
        else:
          if d == 5:
            print(d)
          else:
            if d == 6:
              print(d)
            else:
             if d == 7:
              print(d)
             else:
              if d == 8:
                print(d)
              else:
                if d == 9:
                  print(d)
```

```
enter d5
5
```

[ ]:
```
#EvenOddIfelse.py
d= int(input('enter d'))
if d %2 ==0 :
  print("{} is Even".format(d))
else:
  print("{} is Odd".format(d))
```

```
enter d6
```

```
6 is Even
```

41 if .. elif..else statement

```
[ ]: # Digit check
d= int(input("Enter digit in 0 to 9 "))
if d == 0:
  print("Zero", d)
elif(d==1):
  print('ONE',d)
elif(d==2):
  print('TWO', d)
elif(d==3):
  print('Three',d)
elif(d==4):
  print('FOUR', d)
elif(d==5):
  print('FIVE',d)
elif(d==6):
  print('SIX', d)
elif(d==7):
  print('Seven',d)
elif(d== 8):
  print('EIGHT', d)
elif(d==9):
  print("NINE", d)
else:
  print("Not found ", d)
```

```
Enter digit in 0 to 9 11
Not found  11
```

42 match case statement

- syntax

match(Choice Expr):

```
case Choice Label1:
    Block of Statements-1
case Choice Label2:
    Block of Statements-2
case Choice Label3:
    Block of Statements-3
...
case Choice Label-n:
    Block of Statements-n
case _: # Default Case Block
    Default Block of Statements
```

[ ]:
```python
#MatchCaseEx1.py
print('-'*50)
print("\t Arithmatic Operations")
print('-'*50)
print('\t 1.Addition')
print('\t 2.Substraction')
print('\t 3.Division')
print('\t 4.Multiplication')
print('-'*50)
A= int(input("ENTER A VALUE \t"))
B= int(input("ENTER B VALUE\t"))
ch=int(input("Enter your choice"))
print('-'*50)
match (ch):
        case 1:
          C= A+B
          print('Addition of {} and {}  = {}'.format(A,B,C))
        case 2:
          C= A - B
          print('Substraction of {} and {}  = {}'.format(A,B,C))
        case 3:
          C= A / B
          print('DIVISION of {} and {}  = {}'.format(A,B,C))
        case 4:
          C= A * B
          print('Multiplication of {} and {}  = {}'.format(A,B,C))
        case _:
          print('Wrong Selection of choice',ch)
print('-'*50)
print("Program Execution completed ")
```

```
--------------------------------------------------
         Arithmatic Operations
--------------------------------------------------
         1.Addition
         2.Substraction
         3.Division
         4.Multiplication
--------------------------------------------------
ENTER A VALUE   100
ENTER B VALUE   250
```

```
Enter your choice3
--------------------------------------------------
DIVISION of 100 and 250  = 0.4
--------------------------------------------------
Program Execution completed
```

43 looping or Iterative OR repetative statements

43.1 while OR while. Else loop

[ ]:
```python
#EvenNumGenEx.py
#Program for generating even numbers within the range
n =int(input('Enter number to find the even numbers'))

if n < 0 :
  print("Entered number is less than zero",n)
else:
  i = 2
  while i <= n:
    print(i)
    i += 2
```

```
Enter number to find the even numbers10
2
4
6
8
10
```

[ ]:
```python
#Program for generating 1 to n numbers where n is +ve
n =int(input('Enter number to find the even numbers'))
if (n <= 0 ):
  print("Number is less than zeor", n)
else:
  i=1
  while (i <= n ):
      print(i )
      i +=1
```

```
Enter number to find the even numbers5
1
2
```

```
3
4
5
```

```
[ ]: #Factorial Calculation:
     # This program calculates the factorial of a given number using a while loop.
     n =int(input('Enter number to find tFactorial'))
     if (n <= 0 ):
      print("Number is less than zeor", n)
     else:
       i = 1;
       while (i <= n):
         fact= n * n
         i +=1
     print(fact)
```

```
Enter number to find tFactorial4
16
```

```
[ ]: #Guessing Game:
     #This program implements a simple guessing game where the user has to guess a␣
      ↪randomly generated number within a specified range.
     #the game continues until the user correctly guesses the number.

     import random

     random_num=random.randint(1,10)
     guess=None

     while guess != random_num:
       guess=int(input('Enter number to check '))
       if guess < random_num:
         print("Too Low...! Try Again")
       else:
         print("Too High ..! Try Again")
     print('Congratulations your Guess is correct ', guess , random_num)
```

```
Enter number to check 1
Too Low…! Try Again
Enter number to check 2
Too Low…! Try Again
Enter number to check 3
Too Low…! Try Again
Enter number to check 4
Too Low…! Try Again
Enter number to check 9
Too High ..! Try Again
Enter number to check 8
```

```
Too High ..! Try Again
Enter number to check 7
Too High ..! Try Again
Enter number to check 6
Too High ..! Try Again
Enter number to check 5
Too High ..! Try Again
Congratulations your Guess is correct  5 5
```

44 for loop or for...else loop

[ ]:
```
#program for demonstrating for loop
#ForLoopEx1.py

for i in range(5):
  print(i)
```

```
0
1
2
3
4
```

[ ]:
```
#program for demonstrating for loop
#ForLoopEx1.py
lst=[10,"Rossum",23.45,True,2+3j]
i=0
print(lst)
print('-'*50)
while i < len(lst):
      print(lst[i])
      i = i+1
print('-'*50)
for i in lst:
  print(i)
```

```
[10, 'Rossum', 23.45, True, (2+3j)]
--------------------------------------------------
10
Rossum
23.45
True
(2+3j)
--------------------------------------------------
10
```

```
Rossum
23.45
True
(2+3j)
```

[ ]:
```
#ForLoopEx2.py
line=input("Enter the line of text")
if (len(line) <0 ):
  print("No text entered ")
else:
  i=0
  while i < len(line):
    print(line[i])
    i = i+1
print('-'*50)
for ch in line:
  print(ch)
```

```
Enter the line of textpams
p
a
m
s
--------------------------------------------------
p
a
m
s
```

[ ]:
```
#Sum of Numbers:
#Calculate the sum of all numbers from 1 to a given number n.
num=int(input("Enter the number to find sum"))
if (num <0 ):
  print("Entered values is less than 0  ")
else:
  sum=0
  for n in range(1, num+1):
    sum  += n
  print(sum)
```

```
Enter the number to find sum15
120
```

[ ]:
```
# Factorial Calculation:
# Calculate the factorial of a given number n.
#Sum of Numbers:
#Calculate the sum of all numbers from 1 to a given number n.
num=int(input("Enter the number to find sum"))
```

```
if (num <0 ):
  print("Entered values is less than 0  ")
else:
  fact=1
  for n in range(1, num+1):
    fact  *= n
  print(fact)
```

```
Enter the number to find sum5
120
```

[ ]:
```
# Count Characters in a String:
# Count the occurrences of each character in a given string.

st=input("Enter the string to count the char")
if (len(st) < 0):
  print("No String Entered")
else:
  count=0
  d1=dict()
  for ch in st:
    if ch in d1:
      d1[ch] +=1
    else:
      d1[ch] = 1
  print(d1)
```

```
Enter the string to count the charAABBCCDDDDEEE
{'A': 2, 'B': 2, 'C': 2, 'D': 4, 'E': 3}
```

[ ]:
```
#Program for accepting a line of text and disply in reverse order without using␣
 ↪slicing
#LineReverseEx1.py

line=input("Enter the line of text : ")
if (len(line) <0 ):
  print("No text entered ")
else:
  revstr=''
  for l in range(len(line)-1,-1,-1):
    revstr = revstr+line[l]
print("Inserted String : ",line)
print("Revered String : ", revstr)
```

```
Enter the line of text : pams
Inserted String :  pams
Revered String :  smap
```

[ ]:
```python
#Program for deciding whether the given number is perfect or not
#PerfectEx1.py
number = 6
divisors_sum = 0

for i in range(1, number):
    if number % i == 0:
        divisors_sum += i

if divisors_sum == number:
    print(number, "is a perfect number.")
else:
    print(number, "is not a perfect number.")
```

```
6 is a perfect number.
```

45 Transfer flow control statements

- break
- continue
- pass
- return

[ ]:
```python
#Program for demonstrating break statement
#breakex1.py
s=input("Enter string")
br=input('enter break char')
for ch in s:
  print(ch)
print('-'*50)
for ch in s:
  if ch == br:
        break
  print(ch)
```

```
Enter stringPYTHON
enter break charO
P
Y
T
H
O
N
--------------------------------------------------
P
Y
T
H
```

```
[ ]: #
     # Find Prime Numbers:
     # This program finds prime numbers within a given range.

     number=int(input("Enter the range to get prime number"))

     if (number < 1):
       print("Number is less than 0 ")
     else:
         for num in range(1, number+1):
           if num >1:
             for i in range(2, num):
               if num % i == 0:
                 break
             else:
               print(num)
```

```
Enter the range to get prime number11
2
3
5
7
11
```

```
[ ]: #WAP which will accept a numerical integer value and decide whether it is prime␣
      ↪or not.
     #PrimeEx1.py
     num=int(input("Enter a number"))
     if num < 0:
       print('Number less than 0')
     else:
       res='PRIME'
       for i in range(2,num):
         if num % i ==0:
             res='NOT PRIME'
             break

       if (res =="PRIME"):
         print('{} is {} '.format(num, res))
       else:
         print('{} is {} '.format(num, res))
```

```
Enter a number11
11 is PRIME
```

```
[ ]: #program for finding sum of digits of given +ve number ---while loop
     #NumDigitsSEmEx1.py
```

```
num=int(input("Enter a number"))
if num < 0:
  print('Number less than 0')
else:
  sum=0
  for i in range(num+1):
    sum = i+sum
print(sum)
```

```
Enter a number6
21
```

[ ]:
```
#program for finding sum of digits of given +ve number ---while loop
#NumDigitsSEmEx1.py
# eg. digit 2356
num=int(input("Enter a number"))
if num < 0:
  print('Number less than 0')
else:
  sum=0
  while (num >0):
    r=num % 10
    sum +=r
    num=num //10

print("Sum of digits are ",sum)
```

```
Enter a number12345
Sum of digits are  15
```

45.1 Continue

[ ]:
```
#Program for demonstarting continue statement
#continueex1.py

str=input("Enter a number")
if len(str) < 0:
  print('Empty String')

else:
  for ch in str:
    if ch == 't':
      continue
    else:
      print(ch)
```

```
Enter a numberpython
p
```

```
y
h
o
n
```

[ ]:
```python
#Program for demonstarting continue statement
#continueex1.py
str=input("Enter a number")
if len(str) < 0:
  print('Empty String')

else:
  for ch in str:
    if ch == 'a' or ch == 'e':
      continue
    else:
      print(ch)
```

```
Enter a numberprashant sundge
p
r
s
h
n
t
 
s
u
n
d
g
```

[ ]:
```python
#Program for displaying only vowels from line of text
#continueex3.py
str=input("Enter a number")
if len(str) < 0:
  print('Empty String')
lst=['a','e','i','o','u','A','E','I','O','U']
for ch in str:
  if ch  not in lst:
    continue
  else:
    print(ch)
```

```
Enter a numberprashant sundge
a
a
u
```

e

46 inner or Nested loops

```
[ ]: #Program for demonstarting Inner loops--for loop in for loop
     #InnerLoopEx1.py

     for i in range(5):
       print('outer loop', i)
       for j in range(2):
         print('\t inner loop ', j)
```

```
outer loop 0
         inner loop  0
         inner loop  1
outer loop 1
         inner loop  0
         inner loop  1
outer loop 2
         inner loop  0
         inner loop  1
outer loop 3
         inner loop  0
         inner loop  1
outer loop 4
         inner loop  0
         inner loop  1
```

```
[ ]: #Program for demonstarting Inner loops--while loop in while loop
     #InnerLoopEx2.py
     i=0

     while i < 5:
       print('outer while loop ',i)
       j=0
       while j <= 2:
         print('\t inner while loop ', j)
         j +=1
       i +=1
```

```
outer while loop  0
         inner while loop  0
         inner while loop  1
         inner while loop  2
outer while loop  1
         inner while loop  0
         inner while loop  1
         inner while loop  2
outer while loop  2
         inner while loop  0
         inner while loop  1
         inner while loop  2
outer while loop  3
         inner while loop  0
         inner while loop  1
         inner while loop  2
outer while loop  4
         inner while loop  0
         inner while loop  1
         inner while loop  2
```

47 PATTERN

[ ]:
```
n=5
for i in range(n):  # row
  for j in range(n): # columns
    print('*' , end=' ')
  print()
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

[ ]:
```
n=5
for i in range(n):  # row
  for j in range(i+1): # columns
    print('*' , end=' ')
  print()
```

```
*
* *
* * *
* * * *
* * * * *
```

[ ]:
```
n=5
for i in range(n):  # row
  for j in range(i, n): # columns
    print('*' , end=' ')
  print()
```

```
* * * * *
* * * *
* * *
* *
*
```

[ ]:
```
n=5
for i in range(n):  # row
  for j in range(i, n): # columns
    print(' ', end='')
  for j in range(i+1):
    print('*', end = '')
  print()
```

```
    *
   **
  ***
 ****
*****
```

[ ]:
```
n=5
for i in range(n):  # row
  for j in range(i+1): # columns
    print(' ', end='')
  for j in range(i, n):
    print('*', end = '')
  print()
```

```
*****
 ****
  ***
   **
    *
```

[ ]:
```
n=5
for i in range(n):  # row
  for j in range(i, n): # columns
    print(' ', end='')
  for j in range(i):
    print('*', end = '')
  for j in range(i+1):
    print('*', end = '')
  print()
```

```
    *
   ***
  *****
 *******
*********
```

[ ]:
```
n=5
for i in range(n-1):  # row
  for j in range(i, n): # columns
    print(' ', end='')
  for j in range(i):
    print('*', end = '')
  for j in range(i+1):
    print('*', end = '')
  print()
for i in range(n):  # row
  for j in range(i+1): # columns
    print(' ', end='')
  for j in range(i, n-1):
    print('*', end = '')
  for j in range(i, n):
    print('*', end = '')
  print()
```

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

48 Student Report Card

Write a python program which will implement the following let us assume thier esist a student which contain stdNo stdName Marks in 3 Subjects CM CppM PyM Read stdNo,stdName and marks in 3 subject 1. Calculate total marks (totalMark=CM+CppM+PyM) 2. Calculate Percentage of marks =(totalMarks/300)*100 3. Decide the Grade of Student - if the student got marks less than 40 in any subject then grade is failed - if total marks with in the range of 300 to 250 grade should be DISTINCTION - if total marks with in the range of 249 to 200 grade should be FIRSTCLASS - if total marks with in the range of 199 to 150 grade should be SECONDCLASS - if total marks with in the range of 149 to 120 grade should be THIRDCLASS

Print the Total Mark Report

[ ]:
```
#StudentReportcard.py
print("*"*50)
while(True):
  stno=int(input("Enter Student Roll Number"))
  if (stno >= 100) and (stno <= 200):
    break
  else:
    print('Please enter student Roll Number proerly'.format(stno))
stName=input("Enter the Name of Student")
print("*"*50)
while(True):
  cm=int(input('Enter marks for CM'))
  if (cm >= 0 ) and (cm <= 100):
      break
  else:
      print('Please enter marks in range 0 to 100')
while(True):
  cpp=int(input('Enter marks for cpp'))
  if (cpp >= 0 ) and (cpp <= 100):
      break
  else:
      print('Please enter marks in range 0 to 100')
while(True):
  pym=int(input('Enter marks for cpp'))
  if (pym >= 0 ) and (pym <= 100):
      break
  else:
      print('Please enter marks in range 0 to 100')

total=cm+cpp+pym
avg =total/3

print('-----Report Card------')
print("*"*50)
print('Roll No. :',stno)
print('Student Name :',stName)

print('-----Marks Obtained------')
print("*"*50)
print('C Language  :',cm)
print('C++ Language  :',cpp)
print('Python Language  :',pym)
print("*"*50)
print('-----Score------')
print("*"*50)
print("Total Score {} out of 300".format(round(total,2)))
print("Total Percentage  {} % ".format(round(avg,2)))
print('-----Result------')
result = 'PASS'
if cm <= 40 or cpp <=40 or pym <= 40 :
      result = 'FAIL'
      print('YOU FAILED..! BEST OF LUCK', result)
else:
  if result == 'PASS':
    if total >= 250:
      result = 'DISTINCTION'
    if total <= 250 and total > 200:
      result = 'FIRST CLASS'
    if total <= 200 and total > 149:
      result = 'SECOND CLASS'
    if total <= 149 and total > 120:
      result = 'THIRD CLASS'

print("Grade :  {} % ".format(result))
```

```
**************************************************
Enter Student Roll Number152
Enter the Name of Studentpams
**************************************************
Enter marks for CM85
Enter marks for cpp96
Enter marks for cpp85
-----Report Card------
**************************************************
Roll No. : 152
Student Name : pams
-----Marks Obtained------
**************************************************
C Language  : 85
C++ Language  : 96
Python Language  : 85
**************************************************
-----Score------
**************************************************
Total Score 266 out of 300
Total Percentage  88.67 %
-----Result------
Grade :  DISTINCTION %
```

49 HACKERRANK PROBLEMS

Task Given an integer, , perform the following conditional actions:

If is odd, print Weird If is even and in the inclusive range of to , print Not Weird If is even and in the inclusive range of to , print Weird If is even and greater than , print Not Weird Input Format

A single line containing a positive integer, .

Constraints

Output Format

Print Weird if the number is weird. Otherwise, print Not Weird.

Sample Input 0

3 Sample Output 0

Weird Explanation 0

is odd and odd numbers are weird, so print Weird.

Sample Input 1

24 Sample Output 1

Not Weird Explanation 1

and is even, so it is not weird.

[ ]:
```
import math
import os
import random
import re
import sys

if __name__ == '__main__':
  n = int(input().strip())

  if n % 2 == 1:
    print('Weird')
  else:
    if n % 2 == 0 and n >2 and n < 5:
      print('Not Weird')
    else:
      if n % 2 == 0 and n > 6 and n < 20:
        print('Weird')

      else:
        if n % 2 == 0 and n > 20:
          print('Not Weird')
```

```
3
Weird
```

[ ]:
```
for i in range(5):
  i  *= i
  print(i)
```

```
0
1
4
9
16
```

```
[ ]: def is_leap_year(year):
         if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
             return True
         else:
             return False

     year = int(input("Enter a year: "))

     if is_leap_year(year):
         print(f"{year} is a leap year.")
     else:
         print(f"{year} is not a leap year.")
```

```
Enter a year: 2024
2024 is a leap year.
```

```
[ ]: if __name__ == '__main__':
       n= int(input())
       lst=list()
       for i in range(1,n+1):
         print(i,end='')
```

```
2
12
```

50 FUNCTION

```
[ ]: #approachex.py
     # INPUT : Taking input from function call
     # PROCESSING : done in the function  body
     # RESULT : given to function call

     def sumop(a,b):
       c=a+b   # C is local variable
       return c  # return is keyword used for giving results

     # main program
     a=float(input('Enter First value'))
     b=float(input('Enter second value'))
     c = sumop(a,b)
     print('sum(  {} , {} ) = {}'.format(a,b,c))
     print('_'*50)
     print('type of sumop =', type(sumop))
```

```
Enter First value52
Enter second value20
sum(  52.0 , 20.0 ) = 72.0
--------------------------------------------------
type of sumop= <class 'function'>
```

[ ]:
```python
# approachex.py

def sumop():
  a=float(input('Enter First value'))
  b=float(input('Enter second value'))
  c=a+b   # C is local variable
  print('sum(  {} , {} ) = {}'.format(a,b,c))

#main program
sumop()
```

```
Enter First value10
Enter second value25.
sum(  10.0 , 25.0 ) = 35.0
```

[ ]:
```python
n=int(input("Enter number to find odd even"))
def odd_even(n):
  if n % 2 ==0:
    print('EVEN', n)
  else:
    print("ODD ", n)


# main program
odd_even(n)
```

```
Enter number to find odd even56
EVEN 56
```

[ ]:
```python
#funex01.py
print("Line : 2 im from the begining of function Def")
def hello():
  print('Hello Python World line 4:')
#main program
print('line 6 Im after function def')
hello() # funciton call
print('line 8 im after fucniton call')
```

```
Line : 2 im from the begining of function Def
line 6 Im after function def
Hello Python World line 4:
line 8 im after fucniton call
```

```
[ ]: #Program for cal area and perimeter of Circle by using Functions
     #CircleAreaPeri.py
     def area(r):
       ac=3.14*r**2
       print('Area Of Circle ={}'.format(ac))

     def perimeter():
       r=float(input('Enter Radious for cal permiter:'))
       pc=2*3.14*r
       print('Perimeter of circle ={}'.format(round(pc,2)))
     #main program
     r =float(input('Enter radious for calculating Area'))
     area(r)
     print('_'*50)
     perimeter()
```

```
Enter radious for calculating Area5
Area Of Circle =78.5
__________________________________________________
Enter Radious for cal permiter:3
Perimeter of circle =18.84
```

```
[ ]: #Calculate Simple interest and total amount to pay
     #compaundinterest.py
     #Simple Interest = Principal x Interest Rate x Time.

     def simple_interest(p,i, t):

       si=(p*i*t)/100
       totalamt=si+p
       return totalamt

     def compound_interest(p,i,t,n):
       #A=p*(1+i/n)**(n*t)
       A = p * (pow((1 + i / 100), t))
       interest = A- p

       return interest

     def accept_inputs():
       p=float(input('Enter Priniciple ammount '))
       i=float(input('Enter interest rate applied'))
       t=float(input('Enter number of Years '))
       n=float(input('Enter number of times interest applied per time period'))
       print('-'*50)
```

```
  return p,i,t,n

# main program
p,i,t,n = accept_inputs()
totalamt = simple_interest(p,i,t)
interest = compound_interest(p,i,t,n)
print('-'*50)
print("Principle Amount : {}".format(p))
print("Interest Rate Applied : {}".format(i))
print('Duration : {}'.format(t))
print('-'*50)
print("Simple Interest applied : {}".format(round(totalamt,2)))
print("Compound Interest applied : {}".format(round(interest,2)))
```

```
Enter Priniciple ammount 50000
Enter interest rate applied10
Enter number of Years 3
Enter number of times interest applied per time period3
--------------------------------------------------
--------------------------------------------------
Principle Amount : 50000.0
Interest Rate Applied : 10.0
Duration : 3.0
--------------------------------------------------
Simple Interest applied : 65000.0
Compound Interest applied : 16550.0
```

[ ]:
```
#Program swap case of text / word by using functions
#SwapCaseEx1.py

def swap_word(word):

  swap=''
  for w in word:
    if w.isupper():
      swap=swap+w.lower()
    elif (w.islower()):
      swap=swap+w.upper()
    elif (w.isspace()):
      swap=swap+w
  return swap


#main program
word=input('Enter word to swap ')
swap=swap_word(word)
print("Original Word : {} and Swapped word : {}".format(word, swap))
```

```
Enter word to swap pRaShAnT SunDGE
Original Word : pRaShAnT SunDGE and Swapped word : PrAsHaNt sUNdge
```

[ ]:
```
#Program finding number of occurences of Each Letter in a word / line
#WordCharOccurenceEx.py
def occ_word(word):
  d=dict()
  for ch in word:
    if ch not in d:
      d[ch] = 1
    else:
      d[ch]=d[ch]+1
  return d

#main program
word=input('Enter word to swap ')
d=occ_word(word)
print("Original Word : {} and Swapped word : {}".format(word, d))
```

```
Enter word to swap PRASHANT
Original Word : PRASHANT and Swapped word : {'P': 1, 'R': 1, 'A': 2, 'S': 1,
'H': 1, 'N': 1, 'T': 1}
```

51 Positional Arguments

- The number of Arguments of function call must be equal to the number of formal parameters in function heading
- This parameter mechanism also recommends order and meaning of parameters for higher accuracy
- the default argument passing mechanism is positional aruments or parameters
- syntax fun defination [def fun_name(param1, param2, param-n):]
- syntax fun call [fun_nam(arg1,arg2, …arg-n]

[ ]:
```
#Program for Demonstarting Possitional Arguments
#PosArgsEx1.py

def studinfo(sno, sname, smarks):
  print("\t{}\t{}\t{}".format(sno,sname,smarks))

# main program

print('+'*50)
print('\tRoll No.\tName\t Marks')
print('+'*50)
studinfo(10, 'PRASHANT', 70.5)
studinfo(11, 'Ranjana', 85.90)
studinfo(12, 'Srikant', 65.5)
studinfo(13, 'Sachin', 72.35)
print('+'*50)
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++
        Roll No.        Name     Marks
++++++++++++++++++++++++++++++++++++++++++++++++++
        10      PRASHANT        70.5
        11      Ranjana 85.9
        12      Srikant 65.5
        13      Sachin  72.35
++++++++++++++++++++++++++++++++++++++++++++++++++
```

[ ]:
```
#Program for Demonstarting Possitional Arguments
#PosArgsEx2.py

def studinfo(sno, sname, smarks,crs):
  print("\t{}\t{}\t{}\t{}".format(sno,sname,smarks,crs))

# main program

print('+'*50)
print('\tRoll No.\tName\tMarks\tCourse')
print('+'*50)
studinfo(10, 'PRASHANT', 70.5, 'Python')
studinfo(11, 'Ranjana', 85.90, 'Python')
studinfo(12, 'Srikant', 65.5, 'Python')
studinfo(13, 'Sachin', 72.35, 'Python')
print('+'*50)
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++
        Roll No.        Name     Marks   Course
++++++++++++++++++++++++++++++++++++++++++++++++++
        10      PRASHANT        70.5    Python
        11      Ranjana 85.9    Python
        12      Srikant 65.5    Python
        13      Sachin  72.35   Python
++++++++++++++++++++++++++++++++++++++++++++++++++
```

52 Default paramtere

- when we use default parameter in the function defination they must be used as last paramter otherwise we get error (SytaxError - non-defualt argumnet follows default argument)

[ ]:
```
#Program for Demonstarting Possitional Arguments
#PosArgsEx2.py
```

```
#def studinfo(sno, sname, crs='Python',smarks): # SyntaxError: non-default␣
↪argument follows default argument
def studinfo(sno, sname, smarks,crs='Python'):
  print("\t{}\t{}\t{}\t{}".format(sno,sname,smarks,crs))

# main program

print('+'*50)
print('\tRoll No.\tName\tMarks\tCourse')
print('+'*50)
studinfo(10, 'Prashant', 70.5 )
studinfo(11, 'Ranjana',  85.90 )
studinfo(12, 'Srikant',  65.5)
studinfo(13, 'Sachin',  72.35)
print('+'*50)
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++
        Roll No.        Name    Marks   Course
++++++++++++++++++++++++++++++++++++++++++++++++++
        10      Prashant        70.5    Python
        11      Ranjana 85.9    Python
        12      Srikant 65.5    Python
        13      Sachin  72.35   Python
++++++++++++++++++++++++++++++++++++++++++++++++++
```

[ ]:
```
#Program for Demonstarting Default Arguments
#DefArgsEx2.py
def studinfo1(sno,sname,marks,crs="PYTHON",cnt="INDIA"):
    print("\t{}\t{}\t{}\t{}\t{}".format(sno,sname,marks,crs,cnt))
def studinfo2(sno,sname,marks,crs="JAVA",cnt="INDIA"):
  print("\t{}\t{}\t{}\t{}\t{}".format(sno,sname,marks,crs,cnt))
#main program
print("-"*50)
print("\tSNO\tNAME\tMARKS\tCOURSE\tCOUNTRY")
print("-"*50)
studinfo1(10,"RS",23.45) # Function call--Specific data
studinfo1(20,"TR",34.56) # Function call--Specific data
studinfo1(30,"MC",14.56,crs="JAVA") # Function call--Specific data
studinfo1(40,"DR",64.56) # Function call--Specific data
print("-"*50)
print("\tSNO\tNAME\tMARKS\tCOURSE\tCOUNTRY")
print("-"*50)
studinfo2(50,"DT",11.56,cnt="USA") # Function call--Specific data
studinfo2(60,"ST",21.16) # Function call--Specific data
studinfo2(70,"UT",21.56) # Function call--Specific data
print("-"*50)
```

```
--------------------------------------------------
        SNO     NAME    MARKS   COURSE  COUNTRY
--------------------------------------------------
        10      RS      23.45   PYTHON  INDIA
        20      TR      34.56   PYTHON  INDIA
        30      MC      14.56   JAVA    INDIA
        40      DR      64.56   PYTHON  INDIA
--------------------------------------------------
        SNO     NAME    MARKS   COURSE  COUNTRY
--------------------------------------------------
        50      DT      11.56   JAVA    USA
        60      ST      21.16   JAVA    INDIA
        70      UT      21.56   JAVA    INDIA
--------------------------------------------------
```

53 keyword paramteres (or) Arguments

- in some of the cicumstances we know the function name and formal paramter names and we dont know the order of formal paramter names and to pass the data/values accurately we must use the concept of keyword paramters

[ ]:
```
#keywordarugment

def keywordargument(a,b,c,d,sub='Python'):
  print(f'\t{a}\t{b}\t{c}\t{d}\t{sub}')

#main program
print('+'*50)
print("\tA\tB\tC\tD\tSubject")
print('+'*50)
keywordargument(25,25,25,25) # positional arugments
keywordargument(a=50,b=60,c=90,d=76) # funciton with keyword arguments
keywordargument(d=45,c=40,b=35,a=30)# funciton with keyword arguments
keywordargument(b=50,a=60,d=70,c=80)# funciton with keyword arguments
#keywordargument(a=50,b=60,58,36)  #SyntaxError: positional argument follows␣
  ↪keyword argument
keywordargument(50,60,c=90,d=76) # function call with positional arguments and␣
  ↪keyword argumnets
keywordargument(sub='MARATHI',b=50,a=60,d=70,c=80) # function call with defualt␣
  ↪argument and keyword args
keywordargument(45,sub='Defualt',b=50,d=70,c=80) # function call with postional␣
  ↪, defualt argument and keyword args
keywordargument(45,b=50,d=70,sub='HINDI',c=80) # function call with postional ,␣
  ↪defualt argument and keyword args
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++
        A       B       C       D       Subject
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++
        25      25      25      25      Python
        50      60      90      76      Python
        30      35      40      45      Python
        60      50      80      70      Python
        50      60      90      76      Python
        60      50      80      70      MARATHI
        45      50      80      70      Defualt
        45      50      80      70      HINDI
```

54 Variable length paramters

- when we have family of multiple function calls with variable number of values/argumnets then with normal python programming. we must define multiple function definations.
- this process leads to more development time to overcome this process we must use concept called *variable length paramters*
- Here **param* is called varible length paramter and it can hold any number of argument values or varible number of argument values
- *prams is* class tuple*
- the **param* must always written at last part of function heading and it must be only one (not multiple params)
- Rule: When we use variable length and default paramters in function heading. we use default paramter as last and before we use variable length paramter and in funciton calls we should not use default paramter as key word argument because variable number of values are treated as positional argument values()

[ ]:
```python
#Program for demonstrating Variable arguments
#PureVarAgrsEx1.py

def disvalues(*kvr): # kvr is varible length param.
  print(kvr,type(kvr))


#main program
disvalues(10) # func call with 1 arguments
disvalues(10,20) # func call with 1 arguments
disvalues(10,20,30) # func call with 1 arguments
disvalues(10,20,30,40) # func call with 1 arguments
disvalues()
disvalues('PRASHANT','PYTHON',23.45,True,2+3j)
```

```
(10,) <class 'tuple'>
(10, 20) <class 'tuple'>
(10, 20, 30) <class 'tuple'>
(10, 20, 30, 40) <class 'tuple'>
() <class 'tuple'>
('PRASHANT', 'PYTHON', 23.45, True, (2+3j)) <class 'tuple'>
```

```
[ ]: #PureVarAgrsEx1.py

     def disvalues(*kvr): # kvr is varible length param.
       print(kvr,type(kvr))
       print('Number of values ={} and its type {}'.format(len(kvr),type(kvr)))
       print('+'*50)



     #main program
     disvalues(10) # func call with 1 arguments
     disvalues(10,20) # func call with 1 arguments
     disvalues(10,20,30) # func call with 1 arguments
     disvalues(10,20,30,40) # func call with 1 arguments
     disvalues()
     disvalues('PRASHANT','PYTHON',23.45,True,2+3j)
```

```
(10,) <class 'tuple'>
Number of values =1 and its type <class 'tuple'>
++++++++++++++++++++++++++++++++++++++++++++++++++
(10, 20) <class 'tuple'>
Number of values =2 and its type <class 'tuple'>
++++++++++++++++++++++++++++++++++++++++++++++++++
(10, 20, 30) <class 'tuple'>
Number of values =3 and its type <class 'tuple'>
++++++++++++++++++++++++++++++++++++++++++++++++++
(10, 20, 30, 40) <class 'tuple'>
Number of values =4 and its type <class 'tuple'>
++++++++++++++++++++++++++++++++++++++++++++++++++
() <class 'tuple'>
Number of values =0 and its type <class 'tuple'>
++++++++++++++++++++++++++++++++++++++++++++++++++
('PRASHANT', 'PYTHON', 23.45, True, (2+3j)) <class 'tuple'>
Number of values =5 and its type <class 'tuple'>
++++++++++++++++++++++++++++++++++++++++++++++++++
```

55 Key Word Variables Length Parameters (or) arguments

```
[ ]: #Program for demonstrating Keyword Variable Length Arguments
     #PureKwdVarLenArgsEx1.py
     def dispvalues(**hyd): # here **hyd is called kwd Var leng para and its type is␣
      ↪dict
       print(hyd,type(hyd))
     #main program
     dispvalues(sno=10,sname="avinash") # Function Call-1 with 2 Kwd args
     dispvalues(eno=20,ename="Rossum")# Function Call-2 with 2 Kwd args
     dispvalues(tno=30,tname="Rajesh",sub="Python")# Function Call-3 with 3 Kwd args
     dispvalues(tid=40,name="Ramesh",sub="java",exp=25)# Function Call-4 with 4 Kwd␣
       ↪args
```

```
{'sno': 10, 'sname': 'avinash'} <class 'dict'>
{'eno': 20, 'ename': 'Rossum'} <class 'dict'>
{'tno': 30, 'tname': 'Rajesh', 'sub': 'Python'} <class 'dict'>
{'tid': 40, 'name': 'Ramesh', 'sub': 'java', 'exp': 25} <class 'dict'>
```

[ ]:

```
```

per

[ ]:

```
# write a function to print the length of the list

def len_list(l):
  print(len(l))

cities=['Mumbai', 'Pune', 'Nashik', 'Nanded', 'Amravati', 'Nagpur']


len_list(cities)
```

```
6
```

[ ]:

```
# write a program to print the elements of a list in sigle line

def print_list(l):
  for item in l:
    print(item , end =' ')

cities=['Mumbai', 'Pune', 'Nashik', 'Nanded', 'Amravati', 'Nagpur']

# main program
print_list(cities)
```

```
Mumbai Pune Nashik Nanded Amravati Nagpur
```

[ ]:

```
# write a program to find the factorial of n

def fact(n):
  f=1
  for i in range(1, n+1):
    f = f*i

  return f

n=int(input('Enter number to find Factorial '))
```

```
fact = fact(n)
print(fact)
```

```
Enter number to find Factorial 5
120
```

[ ]:
```
#write a funciton to conver usd to inr

def converter(usd):
  print("USD {} = {} INR ".format(usd, usd*83))

usd=int(input('Enter USD ammount to convert '))

converter(usd)
```

```
Enter USD ammount to convert 1
USD 1 = 83 INR
```

56 Recursion

[ ]:
```
def natural_number(n):
  sum =0
  for item in range(n+1):
    sum= sum+item
  return sum

sum=natural_number(3)
print(sum)
```

```
6
```

[ ]:
```
def natural_number(n):
  sum =0
  if (n==0):
    return
  for item in range(n+1):
    sum= sum+item
  return sum
  natural_number(n-1)

sum=natural_number(6)
print(sum)
```

```
21
```

```
[ ]: def natural_number(n):
       sum =0
       if (n==0):
         return 0
       return natural_number(n-1) + n

     sum=natural_number(10)
     print(sum)
```

```
55
```

```
[ ]: # write a recrusive function to print all the items from the list
     # use list and index as parameter
     def list_items(lst, idx):
       if (idx == len(lst)):
           return
       print(lst[idx])
       list_items(lst, idx+1)

     cities=['Mumbai', 'Pune', 'Nashik', 'Nanded', 'Amravati', 'Nagpur']

     list_items(cities, 0)
```

```
Mumbai
Pune
Nashik
Nanded
Amravati
Nagpur
```

57 Local variables and Global Variables

- local variables
  - the variable used inside of function body are called local variables
  - the purpose of local variables is that To store the temporary results
- global variables
  - global variables are those which are common values for different function calls
  - in the other words if the value is common for all the different function calls then such type of values must be taken as global variables

```
[ ]: #Program for demonstrating local and global variables
     #LocalGlobalVarEx1.py

     def fun1():
       subject1='ML'
       print("To Learn {} we must first learn the {}".format(subject1, lang)) # lang␣
      ↪is global variable subject is local variable
```

```
def fun2():
  subject2='DEEP LEARNING'
  print("To Learn {} we must first learn the {}".format(subject2, lang))

def fun3():
  subject3='AI'
  print("To Learn {} we must first learn the {}".format(subject3, lang))


# main program
lang='Python'
fun1()
fun2()
fun3()
```

```
To Learn ML we must first learn the Python
To Learn DEEP LEARNING we must first learn the Python
To Learn AI we must first learn the Python
```

[ ]:

```
#Program for demonstrating local and global variables
#LocalGlobalVarEx1.py

def fun1():
  subject1='ML'
  global lang
  lang='JAVA'
  print("To Learn {} we must first learn the {}".format(subject1, lang)) # lang␣
 ↪is global variable subject is local variable

def fun2():
  subject2='DEEP LEARNING'
  print("To Learn {} we must first learn the {}".format(subject2, lang))

def fun3():
  subject3='AI'
  print("To Learn {} we must first learn the {}".format(subject3, lang))


# main program
lang='Python'
fun1()
fun2()
fun3()
```

```
To Learn ML we must first learn the JAVA
To Learn DEEP LEARNING we must first learn the JAVA
To Learn AI we must first learn the JAVA
```

[ ]:
```
```