16/7/25 Om Sri Sai Ram (16)

1) Find Outputs

```
a = "Ramo Rao"          # Ramo Rao
print(a)                # Rama Rao
print(type(a))          # string is <class 'str'>
print(id(a))            # memory address
b = 'Hyd'               # 'Hyd' string can be in single quote
print(b)                # Hyd
c = """Hyb is green city.
    Hyb is hitec city
    Hyb is beautiful city"""
print(c)                #  Hyd is green city
                           Hyd is hitec city
                           Hyd is beautiful city
```

multi lines using triple quotes

multi line str " " comment triple quotes

Index demo program (1C)

```
a = 'Hyd'               # 'Hyd'        Hyd
                                       0 1 2
print("How to print 'H' of object 'a'") # print 'H'  print(a[0])
print("How to print 'y' of objet 'a')   # print 'y'  print(a[1])
print("How to print 'd' of object 'a')  # print 'd'  print(a[2])
print(a[3])             # invalid string index out of range
print("How to print 'd' of object 'a')  # print 'd'  (a[-1])   d y H
print("How to print 'y' of object 'a')  # print 'y'  (a[-2])   -1 -2 -3
print("How to print 'H' of object 'a')  # print 'H'  (a[-3])
print(a[-4])            # lid negative index
print(a[0]==a[-3])      # 'H' == 'H'
a[2] = 'c'              # strings are immutable string cannot be modified
print(25[0])            # index number invalid 25 interger non sequence is not index so error
print('25'[0])          # '2'
print(True[1])          # Boolens are not subscriptable non sequence
print('True'[1])        # prints 'r'   == objects are compared
                                       = assign reference to object
print(False[3])         # print 's'
```

16/7/25 Om Sri Sai Ram (1b)

① Find outputs

```
a = "Ramo Rao"          # Ramo Rao
print(a)                # Rama Rao
print(type(a))          # string is <class 'str'>
print(id(a))            # memory address
b = 'Hyd'               # 'Hyd' string can be in single quote
print(b)                # Hyd
c = """Hyb is green city.      # multi lines using thriple quotes
    Hyb is hitec city
    Hyb is beautiful city"""
print(c)                #  Hyd is green city    multi line
                           Hyd is hitec city      comm
                           Hyd is beautiful city  triple quot
```

② # Index demo program (1c)

```
a = 'Hyd'                                  # 'Hyd'        Hyd
print("How to print 'H' of object 'a')     # print 'H'    print(a[0])   0 1 2
print("How to print 'y' of object 'a')     # print 'y'    print(a[1])
print("How to print 'd' of object 'a')     # print 'd'    print(a[2])
print(a[3])                                # invalid string index out of rang
print("How to print 'd' of object 'a')     # print 'd'    (a[-1])      d y H
print("How to print 'y' of object 'a')     # print 'y'    (a[-2])      -1 -2 -3
print("How to print 'H' of object 'a')     # print 'H'    (a[-3])
print(a[-4])                               # lid negative index
print(a[0]==a[-3])                         # 'H' == 'H'
a[2] = 'c'                                 # strings are immutable string cannot be modified
print(25[0])                               # index number invalid 25 interger non sequence is not index so
print('25'[0])                             # '2'
print(True[1])                             # Booleans are not subscriptable non sequence
print('True'[1])                           # prints 'r'      == objects are compared
print('False'[3])                          # print 's'       = assign reference to object
```

③ # find output (1d)

```
a='Hyd'          # a='Hyd'
print(a*3)       # 'HydHydHyd'
print(a*2)       # 'HydHyd'
print(a*1)       # 'hyd'
print(a*0)       # empty string ''
print(a*-1)      # '' (negative repeat results in empty string
print(25*3)      # 75 . multiplication Non sequence
print('25'*3)    # '252525'        Sequence Repetation
print('25'*4.0)  # Error cannot multiply string by float
print(3*'Hyd')   # 'Hyd Hyd Hyd'
print('25'*True) # '25'   25*1 -> 25   * is overloaded because
                                        it performs many roles
```

④ # find outputs (1e)

```
a='Hyd'
print(a, id(a), sep=':')   # a:Hyd   memory address Hyd <space> Address of 'Hyd'
a=a*3                      # a='HydHydHyd'
print(a, id(a))            # id(a) will return new memory
```

a → Hyd

a → HydHydHyd

it provide space, not the space

Ref. modified to string HydHydHyd

⑤ # len() function (1f) sep=(' ')

```
print(len('Hyd'))          # 3
print(len('Rama Rao'))     # 8
print(len('9247'))         # 4
print(len(''))             # empty string
print(len(' '))            # 1
print(len(689))            # Error cannot be used on integer
                           # 689 is an integer, not a string, list or sequence
```

Non Sequence

⑥ # find outputs (1g)

```
a='''Hyd'''      # we cannot use start a string with 4 double
print(a)         # Hyd
print(len(a))    # 3, 4
print(a[0])      # H
print('''Hyd''') # the invalid syntax
```

```
b = """"Hyd""""  # invalid syntax
print(b)        # "Hyd
print(len(b))   # 5
```

⑦ find output

```
a = 'Sankar Dayal Sarma'   # 'Sankar Dayal Sarma'
print(a[7:12])             # dayal
print(a[7:])               # dayal sarma
print(a[:6])               # sankar        # a[0:6:1] 0 to 5
print(a[:]) # a[0:18:1]    # sankar dayal sarma  # a[0:18:1]
print(a[::])               # sankar dayal sarma  # a[0:18:1]
print(a[1:10:2])           # anrD
print(a[0::2]) # a[0:18:2] # snkrdysrm  # a[0:18:2]
print(a[-5:-1]) # a[7:18:2] # sarm
print(a[::-1])  # a[-5:-1:1] # amrasdayarkrakna
print(a[-1:-19:-1])        # amrs
print(a[-1:-19:-2])        # aam ylasarkns
```

positive 4-1
negative 4+1

```
print(a[3:-3])   # kar dayal sa   a[3:-3:1]
print(a[2:-5])   # nkar Dayal     a[2:-5:1]  default step 1
print(a[-1:-5])  # empty string   a[-1:-5:1] -1 0 2 away from
print(a[3:3])    # from index 3 to 3 is empty slice.
                                  a[3:3:1] 2,3,4,5 away from 2
```

```
#  0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17
#  S  a  n  k  a  r     D  a  y  a  l     S  a  r  m  a
# -18-17-16-15-14-13-12-11-10 -9 -8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3
```

When slicing there will be no error.

Reverse string ::: - string

⑧ # find Outputs

```
a='A'          # 'A'
print(a[1])    # Tries to access the second char doesn't exist  string index out of range
print(a[1:])   # empty string
```

⑨ # int() function demo program

```
print(int(10.6))      # Convert float 10.6 to int => 10
print(int(True))      # True => 1
print(int(False))     # false => 0
print(int('25'))      # '25' => 25
print(int('0075'))    # string with leading zeros => 75
print(int(0B11010))   # Binary 11010 => 26
print(0B11010)        # same as above => 26
print(int(0O6247))    # Octal 6247 => 3247
print(int(0XA7B9))    # same => 3247
print(0O6247)         # same => 3247
print(int(0XA7B9))    # Hex A7B9 => 42937
print(0XA7B9)         # same => 42937
print(int(3+4j))      # error can't convert complex to int
print(int('25.4'))    # error invalid literal for int() with base 10  string float can
print(int('Ten'))     # error invalid literal for int() with base 10
```

16 8 4 2 1
1 1 0 1 0

-512 64 8 1
6 2 4 7

4096 256 16 1
A 2 B 9

10

⑩ # complex float() function demo

```
print(complex(3,4))        # (3+4j)
print(complex(0,4))        # 4j
print(complex(3))          # (3+0j)
print(complex(3.6, 4.6))   # (3.6+4.6j)
print(complex(3.8))        # (3.8+0j)
print(complex(3,4.5))      # (3+4.5j)
print(complex(True,False)) # (1+0j)
```

```
print (complex (True))      # 1+0j
print (complex (False))     # 0+0j
print (complex (True, 4))   # 1+4j
print (complex ('3'))       # (3+0j)
print (complex ('3.8'))     # (3.8+0j)
print (complex ('3', '4'))  # error 'second argument must be numbe
print (complex ('3', 4))    # error both must be argument if 2 args
print (complex ('8', '4'))  # error can't use 2 string
print (complex ('Ten')      # error can't convert "Ten" to number
```

(11) # bool

```
print (bool (0)) # false
print (bool (10)) # True; 10 is non zero
print (bool (-25)) # True - non-zero (even negative) is true
print (bool (0.0)) # False - zero float is false
print (bool (0.1)) # True - non-zero float is true
print (bool (0+0j)) # false
print (bool (10+20j)) # True
print (bool (-15j)) # True
print (bool ('false')) # True Non empty string false
print (bool ('')) # false
print (bool ('Hyd')) # True
print (bool (' ')) # True
print (bool ('True')) # True
```

(12) # str ()

```
print (str (25))    # integer 25 to string '25'
print (str (10.8))  # converts 10.8 to string '10.8'
print (str (3+4j))  # converts complex to string '3+4j'
print (str (True))  # converts Bool True to string 'True'
print (str (false)) # converts Bool false to string 'False'
```

```
print (str (None))          # converts None to string 'None'
```

(13) # oct()

```
print (oct (195))           # Decimal to octal
print (oct (0B10101110010)) # Binary to octal
print (oct (0XA7B91))       # Hexa decimal to octal
```

(14) # hex()

```
print (hex (25))            # converts decimal 25 to hexadecimal
print (hex (0B10101111010111)) # converts binary to hexadecimal
print (hex (0062477))       # converts octal to hexadecimal
```

(15) # float

```
print (float (25))          # converts int 25 to 25.0
print (float (True))        # converts True to float 1.0
print (float (False))       # converts false to float 0.0
print (float ('92'))        # converts string to float 92.0
print (float ('36.4'))      # converts decimal to float 36.4
print (float ('0075'))      # leading zero is fine in string => 75.0
print (float (0B10101011))  # Binary literal = 85 => 85.0
print (float (0062477))     # Octal = 3.247 => 3247.0
print (float (0XA7B91))     # Hex = 42937 => 42937.0
print (float (3+4j))        # error can't convert complex to float
print (float ('Ten'))       # error invalid string for float
```

Day 3:-

① # Find Outputs

```
a = range (10,50,5)        ## range (10, 50, 5)
print(type(a))             # <class 'range'>
print (a)                  # range (10,50,5)
print (*a)                 # 10 15 20 25 30 35 40 45
print(id(a))               # memory address
print (len(a))             # 8
print (*a[2:7], sep=',')   # slicing range object from index 2 to 7 not included
print (*a [::-1])          ## error doesn't suppor because it goes away to -1
a [4] = 32                 # error range object are immutable
print (a*2)                ## error not excuted because the program already crashed at the previous line
```

(*a -> unpacked)

replaced 32 but it is error because it is immutable

10 15 20 25 30 35 40 45
0 1 2 3 4 5 6 7

20, 25, 30, 35

repeated is not permitted

We cannot range order

② # find Outputs

```
a = range (10, 20)         ## 10,11,12,13,14,15,16,17,18,19   [10:19]
print (*a, sep=',')        ## 10,11,12,13,14,15,16,17,18,19
b = range (5)              # rang [0:5] -> 0 to 4 in steps of 1
print (*b)                 # 0,1,2,3,4   0 1 2 3 4
c = range (10, 1, -1)      ## [10 : 2 : -1]
print (*c, sep='...')      # 10...9...8...7...6...5...4...3...2
d = range (-10, 0)         # [-10 : -1]
print (*d)                 # -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
e = range (-10)            # [0 : -10 : 1]   0 >= -10
print (*e)                 # empty range
f = range (2,2)            # empty rang
print (*f)                 # empty
g = range (10, 11, 0.1)    # float object cannot be interpreted as an integer
h = range ('A', 'F')       # 'str' object cannot be interpreted as an integer
```

space

beginning should be greater than or equal end

begining

hold

③ r = range(10, 17, 3) # [10:13:16]

a, b, c = r # a=10, b=13 c=16 unpacked range objects 3 elem

print(a, b, c) # 10, 13, 16 10 13 16

r = range(3) # [0:1:2]

x, y = r # error, Because unpack 3 values (0, 1, 2)

p, q, r, s = r # Error, Be cause Not enough values to unpack (4, 3)

a, b, c = *r # error, Because can not use * when left side already 3 elements.

Both side should be equal.

m = r # Ref 'm' points to object 'r'

print(id(r)) # address of range object 'm'

print(id(m)) # # same address

18/7/25

day:-4,

find outputs

a = [25, 10.8, 'Hyd', True, 3+4j, None, 'Hyd', 25]

print(a) # [25, 10.8, 'Hyd', True, 3+4j, None, 'Hyd', 25]

print(*a) # 25 10.8 Hyd True 3+4j None Hyd 25

print(type(a)) # list

print(id(a)) # memory address

print(len(a)) # 8

a[2] = 'sec' # 25, 10.8, 'sec', True, 3+4j, None, 'Hyd', 25]

print(a) # [25, 10.8, 'sec', True, 3+4j, None, 'Hyd', 25]

print(a[2:5]) # ['sec', True, (3+4j)]

(2) # append() and remove() methods

```
a = []          # empty list
print(a)        # []
a.append(25)    # Adds the integer 25 to the list -> now [25]
a.append(10.8)  # float 10.8 -> [25, 10.8]
a.append('Hud') # string 'Hud' -> [25, 10.8, 'Hud']
a.append(True)  # boolen 'true' -> [25, 10.8, 'Hud', True]
print(a)        # [25, 10.8, 'Hud', True]
a.remove('Hud') # remove first occurrence of 'Hud' (a string) [25, 10.8, True]
print(a)        # [25, 10.8, True]
a.remove('25')  # trying to remove '25' string but 25 is integer not string
print(a)        #                                  in the list.
```

(3) # find outputs

```
a = [25, 10.8, 'Hud']  # created 3 elements
print(a)               # [25, 10.8, 'Hud']
print(id(a))           # memory address
print(a*3)             # [25, 10.8, 'Hud', 25, 10.8, 'Hud', 25, 10.8, 'Hud']
print(a*2)             # [25, 10.8, 'Hud', 25, 10.8, 'Hud']
print(a*1)             # [25, 10.8, 'Hud']
print(a*0)             # [] empty
print(a*-1)            # 'empty string
print(a*4.0)           # can't multiply the sequence by non-integer of float
a = a*3)               # reference is modified to 9 elements
print(a)               # [25, 10.8, 'Hud', 25, 10.8, 'Hud', 25, 10.8, 'Hud']
                       # [25, 10.8, 'Hud', 25, 10.8, 'Hud', 25, 10.8, 'Hud']
print(id(a))           # a is new list (result of a*3), so Id change different from previous
a = [25]               # 25   # print(a, id(a))
print(a*a)             # can't multiply sequence by non-int of type 'list'
                       # sequence  Integer
```

③ ④ # list() function demo

```
a = list('H4d')   # ['H', '4', 'd']
print(a)          # ['H', '4', 'd']
print(type(a))    # list
print(len(a))     # 3
b = (10, 20, 15, 18)  # A tuple is defined
print(list(b))    # [10, 20, 15, 18]
print(list(range(5)))  # [0, 1, 2, 3, 4]
print(list(25))   # int object is not iterable  it is not sequence integer 25
```

Sequence

⑤ # Find outputs

```
a = []          # empty list
print(type(a))  # list type
print(a)        # empty list
print(len(a))   # 0.
b = list()      # empty list
print(b)        # empty list
print(len(b))   # 0
```

slice = list, not modified
ref. "slice new list"

⑥ # slice

```
#  0   1   2   3   4   5   6   7
list = [25, 10.8, 3+4j, 'H4d', True, None, 10.8, 'H4d']
#  -8  -7  -6  -5  -4  -3  -2  -1
print(list[2:7])  # [3+4j, 'H4d', True, None, 10.8]
print(list[:])    # [25, 10.8, (3+4j), 'H4d', True, None, 10.8, 'H4d']
print(list[:])    # list[0:8:1] -> list index 0 to 7 in step 1 [25, 10.8, 3+4j, 'H4
print(list[::-1])  # ['H4d', 10.8, None, True, 'H4d', (3+4j), 10.8, 25]
print(list[::2])   # [25, (3+4j), True, 10.8]
print(list[1::2])  # [10.8, 'H4d', None, 'H4d']
```

True, None as

print(list[::-2]) # list[-1:-9:-2] -> from index -1 to -8 in steps of -2
['H4d', None, 'H4d', 10.8]
print(list[-2::-2]) # index -2 to (10.8) back towards 2 [10.8, True, (3+4j), 25]
print(list[1:4]) # [index 1 to 3] -> [10.8, (3+4j), 'H4d']
print(list[-4:-1]) # index -4 to -2 [True, None, 10.8]
print(list[3:-3]) # ['H4d', True]
print(list[2:-5]) # [(3+4j)]
print(list[-1:-5]) # empty list

⑦ find outputs

```
#   0      1      2      3      4      5      6      7
list = [25, 10.8, 3+4j, 'H4d', True, None, 10.8, 'H4d']

x,4 = list[3:5] # index 3 to 5-1 ['H4d', True]
print('x :', x) # x: H4d
print('4 :', 4) # 4: True
for x in list[2:7]: # inder 2 to 6-1 [3+4j, 'H4d', True, None, 10.8]
    print(x)    : # [3+4j, 'H4d', True, None, 10.8]
```

⑧ outputs

```
#   0   1   2   3   4
a = [10, 20, 30, 40, 50]
print [a, id(a)]  # memory address                    [10, 60, 70, 50]
a[1:4] = [60, 70] # replacing from 1 to 3 (20, 30, 40) with 20 (60, 70
print (a, id(a)) # lis is modified [10, 60, 70, 50]
a[2:4] = {100, 200, 300} # [10, 60, 100, 200, 300]
print(a, id(a)) # in-place modification [10, 60, 100, 200, 300]
```

⑨ # find o/ps

```
a = [25]   # [25]
print (a[1]) # list out of range
print [a[1:]) # empty list
```

(10) find. o/ps

```
a = (25) : # (25) is just 25 (no comma) int
b = 25,    # comma makes it a tuple
c = 25     # int regular integers
d = (25,)  # (25,) tuple with one element (25)
print(type(a))  # <class 'int'>
print(type(b))  # <class 'tuple'>
print(type(c))  # <class 'int'>
print(type(d))  # <class 'tuple'>
print(a*4)      # 25*4=100
print(b*4)      # (25,)*4 = (25, 25, 25, 25)
print(c*4)      # 25*4=100
print(d*4)      # (25,)*4 = (25, 25, 25, 25)
```

(11) # tuple

```
a = tuple('H4d')  # ('H', '4', 'd') type casting
print(a)          # ('H', '4', 'd')
print(type(a))    # tuple
print(len(a))     # 3
b = [10, 20, 15, 18]  # list created
print(tuple(b))       # (10, 20, 15, 18)
print(tuple(range(5))) # (0, 1, 2, 3, 4)
print(tuple(25))      # 'int' object is not iterable
```

(12)

```
a = ()          # empty tuple
print(type(a))  # tuple
print(a)        # () empty
print(len(a))   # () empty 0
b = tuple()     # creates another empty tuple ()
print(b)        # () empty
print(len(b))   # 0
```

(13) Trickly

```
a = (10, 20, 30)  # (10, 20, 30)
print(a)          # (10, 20, 30)
print(id(a))      # memory address
a = a*2           # (10, 20, 30, 10, 20, 30)
print(a)          # (10, 20, 30, 10, 20, 30)
print(id(a))      # new memory ID a*2
```

(14) # set — duplicate value will be deleted

```
a = {25, 10.8, 'H4d', True, 3+4j, None, 25, 'H4d'}  # {True, 10.8, 'H4d', 3+4j, None, 25}
print(a)          # set # {True, 10.8, 'H4d', 3+4j, None, 25}
print(type(a))    # set
print(len(a))     # 6
print(a[2])       # set object is not subscriptable  → index, there is no index
print(a[1:4])     # set object is not subscriptable  not slicable, no slice
a[2] = 'sec'      # 'set' object does not support item assignment  immutable
print(a*2)        # set can't be multiplied repeated  no value when, their is no slice & index
print(a*a)        # set can't be multiplied repeated
```

(15) # Tricky

```
a = {1, 'H4d', False, True, 0.0, '', 1.0, 0}
print(a)          # {'H4d', False, True, ''}
print(len(a))     # 4
print(type(a))    # class set
```

(16) set()

```
a = set('Rama Rao')  # Rama Rao is a string, unique characters
print(a)             # {'a', 'R', 'O', 'A', 'r', 'm', ' '} in any order
print(len(a))        # 7
print(set([10, 20, 15, 20]))  # duplicate value 20 → {10, 20, 15}
print(set({25, 10.8, 'H4d', 10.8}))  # set removes {25, 10.8, 'H4d'}
```

print(set(range(10, 20, 3))) #range(10,20,3) operands [10,13,16,19]
{10,13,16,19}
print(set(25)) # error, Because int object is not iterable
print(set([25, 10.8, [], 'Hyd'])) # error, Because unhashable type 'list'

set cannot have mutable element []

⑦ #find

```
a = []  # empty
b = ()  # empty tuple
c = {}  # empty set, but it's a dict
d = set()  # Empty set
print(type(a)) # list
print(type(b)) # tuple
print(type(c)) # dic
print(type(d)) # set
print(a)  # []
print(b)  # ()
print(c)  # {}
print(d)  # set()
```

{} dic

{10,20} set

⑧ Tricky

add() & remove()

```
a = set()  # set()
a.add(25) # Adds 25 insert 25 into set
a.add(10.8) # Adds 10.8
a.add('Hyd') # Adds 'Hyd'
a.add(True) # True is equivalent to 1
a.add(None) # Add None
a.add('Hyd') # duplicate - ignored
a.add(1)  # Already present as True - duplicate, ignore
print(a)  # {None, 10.8, 'Hyd', 25, True}
print(len(a)) # 5
a.remove(25) # Remove 25
print(a)  # {None, 10.8, 'Hyd', True}
a.append(100) # Error, because set object has no attribute 'append'
```

a.add(set(1)) # Error, because list is unhanshable mutable
a.add((1)) # Allow, Tuple is immutable
a.add([]) # Error, because list is unhanshable mutable
print(a) # {None, 10.8, 'Hyd', True}
a.add({3}) # Error, because dict is unhanshable mutable

19)
How to print set in 2 different ways

a = {25, True, 'Hyd', 10.8} print(c)

print('set with print function') # print(a)

print('How to print set') # set with for loop

print('Iterate thru set with for loop')

How to iterate thru set with for loop

```
for x in a
    print(x) # Hyd <next line> 25 <next> 10.8 <next> True <next
```

day:-5

a = {10: 'Ramesh', 20: 'Kiran', 15: 'Amar', 18: 'sita'}

print(a) # {10: 'Ramesh', 20: 'Kiran', 15: 'Amar', 18: 'sita'}

print(type(a)) # <class 'dict'>

print('How to obtain value key 10') # print(a[10]) # Ramesh

print('How to obtain value key 20') # print(a[20]) # Kiran

print('How to obtain value key 15') # print(a[15]) # Amar

print('How to obtain value key 18') # print(a[18]) # Sita

print(a[19]) # error 19 not in dict

print(a[0]) # 0 not an dict

print(a['Amar']) # 'Amar' - key must match the type (here, key is integers from Amar)

How to modify value of key 15 to 'Krishna' # changes key 15's value

How to remove 20: 'Kiran' from dict 'a' # Remove key 20 & its value Kiran

How to append 25: 'Ramsi' to dict 'a' # Appends a new entry

```
print(a)    # final state of the dict
print(len(a)) # Number of key-value pairs 4
print(a*2) # error unsupport operand type for *: dict & int
```

(2)
```
a={10:'Hyd',10:'Sec'} # In a dictionary.
print(a) # {10:'Sec'}
print(len(a)) # 1
b={'R':'Red','G':'Green','B':'Blue','Y':'Yellow','G':'Gray','B':'Black'}
print(b) # {'R':'Red','G':'Green','B':'Blue','Y':'Yellow','G':'Gray','B':'Black'}
print(len(b)) # 4
```

(3)
```
a={True:'Yes',1:'No',1.0:'May be'} # {True:'may be'}
print(a) # {True:'Maybe'}
print(len(a)) # 1
```

(4)
```
a={[]:25} # Error. Because unhanshable, cannot be used as dict
b={():25} # Empty tuple, Immutable
print(b)  # {():25}
c={{}:25} # error, Dic {} are also mutable
d={'Ramesh':[9948250500,9848565090,9440250404]} # valid
print(d) # {'Ramesh':[9948250500,9848565090,9440250404]}
print(len(d)) # 1.
e={set():10.6} # Error, Because: A set is mutable
```

(5)
```
a={}          # empty dictionary
print(type(a)) # <class 'dict'>
print(len(a))  # empty 0
print(a)       # {} empty dict
b=dict()       # empty dic dict()
print(type(b)) # <class 'dict'>
print(len(b))  # 0
print(b)       # {} empty
```

find outputs

a-[25, 10.8, 'Hyd', True, 3+4j, None, 'Hyd', 25]

print(a) # [25, 10.8, 'Hyd', True, 3+4j, None, Hyd, 25]
(0, 1, 2, 3, 4, 5, 6, 7)

print(*a) # 25 10.8 Hyd True 3+4j None Hyd 25

print(type(a) # list

print(id(a)) # memory address

print(len(a)) # 8

a[2] = 'sec' # 25 10.8 'sec' True 3+4j None Hyd 25

print(a) # [25, 10.8, 'sec', True, 3+4j, None, Hyd, 25

print(a[2:5] # ['sec', True, (3+4j)]

Om Sri Sai Ram

28) Write a program to add, sub, multiply and divide two Complex number

1st comp number → 3+4j

2nd Complex number → 5+6j

What is the sum? → (3+4j) + (5+6j) = 8+10j

What is the difference? (3+4j) + (5+6j) = 2-2j

What is the product? → (3+4j) * (5+6j) = 15+18j+20j-24 = -9+38j

What is the division? (3+4j) / (5+6j) = (3+4j) * (5-6j) / (5+6j) * (5-6j)

= (15+18j+20j+24) / (25+36) = 39/61 + 2j/61

(27) Enter $C_1 = 3+4j$

$C_2 = 5+6j$

sum_result = C1+C2 (3+4j)+(5+6j)

diff_result = C1-C2 (3+4j)-(5+6j)

product_result = C1*C2 (3+4j)*(5+6j)

division_result = C1/C2 (3+4j)/(5+6j)

```
print("first complex number :", (3+4j))
print("second complex number :"(5+6j))
print("sum :" (3+4j)+(5+6j))  x+y)
print("difference :" (3+4j)-(5+6j))  x-y)
print("product :" (3+4j)*(5+6j))  x*y)
print("division :" (3+4j)/(5+6j))  x/y)
```

Wrong

(2) Enter first complex number : 3+4j
Enter Second complex number = 5+6j
Sum : (8+10j)
difference : (-2-2j)

(2) # identify error
else :
    print ('else suite')
print ('outside')

Inualid because else cannot identify without if (or) elif

(3) # Identify error
if 9>5
    print ('Hello')
    print ('Bye')

Inualid, Because colon (:) is missing after the if

(4) # Identify error
if 9>12:
    print ('Hyd')
else
    print ('sec')

Inualid, Because colon (:) is missing after else.

(5) Identify error. if 9>12:
print ('Hyd'

identify error

```
if (10,20,15):
print('Hyd')
else:
print('Sec') # indentation error did not given spac before print
```

Identify error

```
if ():
    print ('Hyd')
else:
    print('Sec')
print ('Bye')
```

Syntax error. Because their is empty condition () and if and else should be same position.

Identify error

```
if { }:
{
  print('one')          # Invalid syntax: { } is not a valid
  print ('Two')            condition or block
  print ('Three')
}
else :
{
   print ('four')
   print ('five')
   print ('six')
}
print ('Bye')
```

Identify error:

```
if ():
    print('one')
    print ('Two')
    print ('Three')
else:
    if []
        print('four')
        print('Five')
        print('six')
    else:
    if{}:
        Print ('seven')
        print('Eight')
        print ('Nine')
    else:
        print('Hyd')
        print('sec')
        print ('Lyb')
print('Bye')
```

() is an empty tuple, which is false.

their is no need of another if

else: if, which is Invalid - it should be 'elif'

Indentation is inconsistent or missing

Write a program to determine a number is even and odd with if statement?

```
num = int (input ("Enter a number: "))

if num % 2 == 0:
    print ("The number is even.")
else:
    print ("The numbu is odd.")
```

find outputs

```
if():
    print('Hyd')   # you are passing an empty tuple () : false
    print('sec')        does not excute
    print('cyb')
else : print ('one')    The else block is excute
    print('Two')
    print ('Three')
print ('Bye')
```

Output :- One
Two
Three
Bye

find outputs

```
if{10:20, 30:40}:   non-empty dict which is true & excuted
    print ('Hyd')
    print ('sec')
    print ('cyb')
print ('Bye')
```

O/p :- Hyd
sec
cyb
Bye

Write a program to print digit in word's with else and if (do not use elif)

```
digit = int(input("Enter a digit (0-9): "))
if digit == 0:
    print("zero")
else:
    if digit == 1:
        print("one")
    else:
        if digit == 2:
            print("Two")
        else:
            if digit == 3:
                print("Three")
            else:
                if digit == 4:
                    print("Four")
                else:
                    if digit == 5:
                        print("Five")
                    else:
                        if digit == 6:
                            print("six")
                        else:
                            if digit == 7:
                                print("Seven")
                            else:
                                if digit == 8:
                                    print("Eight")
                                else:
                                    if digit == 9:
                                        print("Nine")
                                    else:
                                        print("Invalid")
```

Write a program to print month name for input month numbe by using if and elif

*) Enter month number (1-12): 6 June

* Enter month numbes (1:12): 13 Input should be b/w 1 and 12

* Enter month number (1-12): 10.8 Input should be an integer

```
User_input = input ("Enter month number (1=12)
if not user_input.isdigit():
    print("Input should be an integer")
else:
    month = int (user_inputs)
    if month == 1:
        print ("January")
    elif month == 2:
        print ("February")
    elif month == 3:
        print ("March")
    elif month == 4:
        print ("April")
    elif month == 5:
        print ("May")
    elif month == 6:
        print ("June")
    elif month == 7:
        print ("July")
elif month == 8:
    print ("August")
```

```
elif month == 9:
    print ("september")
elif month == 10:
    print ("october")
elif month == 11:
    print ("November")
elif month == 12:
    print ("December")
else:
    print("Input should be between 1 and 12")
```

29/7/25

① Write a program to test year is leap year or not

Hint: Single with 3 conditions and else.

```
year = int (input ("Enter a year : "))
if (year % 4 == 0 and year % 100 != 0
    print (year, "is a leap year")
else:
    print (year, "is Not a leap year")
```

* If we write only 1st condition it shows all the years as leap year or year % 400 == 0.

I/p : 2024 → o/p : leap year
1900 → Not a leap year
2000 → o/p → leap year

* If we do not write the 3rd condition it shows as not leap year

② Write a program to determine the largest of three numbers with if and else

Hint :- Write multiple condition

```
a = int( input ( "Enter first numbes : ")
b= int ( input ("Enter second numbes: ").
c: int ( input ("Enter third numbes: ")

    if a>=b and a>=c
        print (a, "is the largest")
    elif b>=a and b>=c:
        print (b, "is the largest")
    else:
        print (c, "is the largest")
```

③ Write a program to conuert celsius temp to farenheit and vice: versa.

```
print (". Temperature conuersion")
print ("1. celsius to fahrenheit")
print ("2. fahrenheit to celsius")

choice = int (input ("Enter your choice (1 or 2):"))
temp = float ( input ( input ("Enter temperature value:"

if choice ==1:
    celsius = float (input ("Enter celsius temperature:"))
    fahrenheit = 1.8 * celcius + 32
    print ("fores celcius equivalent:", farehneit)
if choice == 2
    fahrenheit = float ( input ("enter Farenheit temperature:"))
```

```
celsius = (fahrenheit - 32) / 1.8
print ("celcius equivalent:", round (celsius, 2))
else:
    print ("Invalid Input")
```

(4) Write a program to test a point (x,4) lies in 1st quadrant, 2nd and 3rd and 4th, x-axis, y-axis or origin

Hint: Use if... elif

```
x = float (input ("Enter x coordinates :"))
y = float (input ("Enter y co-ordinates :"))
if x>0 and y>0:
    print ("point lies in 1st quadrant")
elif x<0 and y>0:
    print ("point lies in 2nd quadrant")
elif x<0 and y<0:
    print ("point lies in 3rd quadrant")
elif x>0 and y<0:
    print ("point lies in 4th quadrant")
elif x!=0 and y==0:
    print ("point lies on x-axis")
elif x==0 and y!=0:
    print ("point lies on y-axis")
elif x==0 and y==0:
    print ("point is at the origin")
```

(5) Write a program to determine ... middle of three numbers?

a = 10 max = 20
b = 20 min = 7
c = 7 mid = (10+7+20) − (20+7) = 10

```
a = int(input("Enter first number:"))
b = int(input("Enter second number:"))
c = int(input("Enter third number:"))

max = a
min = a

if b > max:
    max = b
if c > max:
    max = c
if b < min:
    min = b
if c < min:
    min = c
mid = (a+b+c) - (max-min)

print("Maximum (max) =", max)
print("minimum (min) =", min)
print("Middle (mid) =", mid)
```

(6) Write a program to determine three sides from a triangle or not

Hint : Use nested if

```
import math
a = float(input("Enter side a: "))
b = float(input("Enter side b: "))
c = float(input("Enter side c: "))
```

```
if (a+b>c) and (a+c>b) and (b+c>a):
    print("The sides form a triangle")
    if a==b and b==c:
        print("It is an equilateral Triangle")
        area = (math.sqrt(3)/4)*a**2
        print("Area =", round(area, 2))
    elif a==b or b==c or a==c:
        print("It is an Isosceles triangle")
        perimeter = a+b+c
        print("perimeter =", perimeter)
    else:
        print("It is a scalene Triangle")
        s = (a+b+c)/2
        area = math.sqrt(s*(s-a)*(s-b)*(s-c))
        perimeter = a+b+c
        print("Area =", round(area, 2))
        print("perimeter =", perimeter)
else:
    print("The sides do not form a triangle:")
```

(7) Write a program to determine roots of a quadratic equation $a*x^2+b*x+c=0$ where $a!=0$

```
a = float(input("Enter a (!=0): "))
b = float(input("Enter b: "))
c = float(input("Enter c: "))
if a!=0:
    d = b**2 - 4*a*c
    print("Discriminant =", d)
```

```
if d>0:
    r1 = (-b + math.sqrt(d)) / (2*a)
    r2 = (-b - math.sqrt(d)) / (2*a)
    print("Real and distinct roots:", r1, "and", r2)
elif d==0:
    r = -b / (2*a)
    print("Real and same root:", r)
else:
    real = -b / (2*a)
    imag = math.sqrt(-d) / (2*a)
    print("Complex Roots :", complex(real, imag), "and",
          complex(real, -imag))
else:
    print("'a' must not be 0.")
```

(8) Write a program to determine a point (x,y) lies inside, outside or on the circle.
Center is origin and radius is 'r'

```
x = float(input("Enter x coordinate: "))
y = float(input("Enter y coordinates:"))
r = float(input("Enter radius of the circle:"))
distance = math.sqrt(x**2 + y**2)
if distance > r:
    print("point is outside the circle")
elif distance < r:
    print("point is inside the circle")
else
    print("point is on the circle")
```

9) m = 4

```
match m:
 case 1:
  print('One')
 case 2:
  print('Two')
 case 3:
   print('Three')
 print('Bye')
```

case 1 : does not match
case 2 = "
case 3 = "

m = 4 does not match any case
no case block runs
Then print('Bye') runs
O/p = Bye

10) # i = 2

```
match i:
 case 1:
  print('One')
 case _:
  print('None of the above')
 case 2:
  print('Two')
 print('Bye')
```

O/p : Two
Bye

case:_ is in the middle of the conditions it should be, in the end.

11) # find outputs

```
m = 2
match m:
 case 1:
  print('One')
 case _:
   print('Hello')
 case _:
   print('Bye')
 print('Bye')
print('End')
```

same

O/p :- '_' it is used only once

Hello
End

(12) # find outputs

```
m = 1
match m:
    case 1:
        print('Hyd')
    case 1:
        print('sec')
    case 1:
        print('cyb')
print('Bye')
```

```
print('Hyd')
```

in case: 1 match found it doesn't match execute 2nd one

O/p: Hyd
Bye

if first case is executed and remaining cases are skipped

(13) find outputs

```
ch = 'B'
match ch:
    case 'A':
        print('Apple')
    case 'B':
        print('Book')
    case 'C':
        print('cafe')
    case _:
        print('None of the above')
print('Bye')
```

O/p = case 'A' = Not match
case 'B' = match print ('Book') is executed

Book
Bye

(14)

```
x = eval(input('Enter any number:
match x:
    case 7 | -6 | 0:
        print('Hyd')
        print('sec')
        print('cyb')
```

```
case -10 | 15:
    print ('One')
    print ('Two')
    print ('Three')
case _:                    None of above 5 values
    print ('India')
    print ('China')
    print ('USA')
print ('Bye')              all the above
```

| = bitwise (or) operator

1) if i/p is -6 and the out is Hyd
   sec
   Cyb
   Bye

2) if input is 15 : Matched in case -10 | 15 . o/p :- One Two Three Bye

3) if input is 10.6 :- No match in any of case → goes to case _ India, China USA Bye

4) if input is 0: Matched in case 7 | -6 | 0 o/p → Hyd sec Cyb Bye

5) if input is -10: Matched in case -10 | 15 → One two three Bye

6) if input is 7:- Matched in case 7 | -6 | 0 → Hyd sec Cyb Bye

(15)
```
tpl : eval ( input ('Enter any point. in the form of (x,y) : '))

match tpl:
    case (0,0):
        print ('origin')
    case (0,y):
        print ('y-axis')
    case (x,0):
        print ('x-axis')
    case (x,y):
        print ('Quadrant')
    case _:
        print ('Not a point')
```