**LECTURE-11 (Saturday 12-July-2025)**

* Test
* Test Questions:

1. Unzip:

Code:

x,y = (20,5)

print(x,y)

Output:

20 5

1. Comprehensive Dictionary:

Code:

dictionary1 = { "name" : "ABC", "f\_name" : "DEF", "age" : 25}

print(dictionary1)

print("\n")

print(dictionary1.items())

print("\n")

print({k:v for k,v in dictionary1.items()})

print("\n")

print({v:k for k,v in dictionary1.items()})

Output:

{'name': 'ABC', 'f\_name': 'DEF', 'age': 25}

dict\_items([('name', 'ABC'), ('f\_name', 'DEF'), ('age', 25)])

{'name': 'ABC', 'f\_name': 'DEF', 'age': 25}

{'ABC': 'name', 'DEF': 'f\_name', 25: 'age'}

1. Code:

True or True or 10/0

Output:

True

OR

True and True and 10/0

Output:

---------------------------------------------------------------------------

ZeroDivisionError Traceback (most recent call last)

[/tmp/ipython-input-2-4123072656.py](https://localhost:8080/) in <cell line: 0>()

----> 1 True and True and 10/0

ZeroDivisionError: division by zero

1. Code:

list1 = []

for i in list1:

  print(i)

Empty output:

1. Code:

list1 = []

for i in list1:

  print(i)

else:

  print("P")

Output:

P

Here,

“else” block will always executes

OR

list1 = [1,2]

for i in list1:

  print(i)

else:

  print("P")

Output:

1

2

P

1. Code:

def gen(): pass

print(gen())

Output:

None

1. Code:

list1 = [1,2,3,4,5]

for i in list1:

  print(i)

  if i == 2:

    break

else:

  print("P")

Output:

1

2

Here,

“else” block not executed because “for” loop block breaks the loops.

1. Code:

x = 5

y = 3

print(x&y)

print(x|y)

Output:

1

7

Here,

3 = 11 (binary value)

5 = 101 (binary value)

So,

101 – 5

011 – 3

----------

001 – 1 (&)

111 – 7 (|)

1. Slicing: In Python, slicing is a technique used to extract a subset of elements from sequences like strings, lists, or tuples.

Syntax: sequence[start:stop:step]

Here,

start - index to start the slice but starting index element is included in selected items

stop - index to end the slice but stopping index element is excluded in selected items

step - (optional) how many elements to skip

Code:

                    # 0   1   2   3

                    # -4 -3  -2  -1

names : list[str] = ['a','b','c','d']

print(names)

print(names[1::])

print(names[1:3:])

print(names[-3:-1:])

print(names[::-1])

print(names[-1:-4:-1])

print(names[3:-4:-1])

Output:

['a', 'b', 'c', 'd']

['b', 'c', 'd']

['b', 'c']

['b', 'c']

['d', 'c', 'b', 'a']

['d', 'c', 'b']

['d', 'c', 'b']

Here,

“[1::]” - it will read items from index “1” (b) till end as other values are not provided.

“[1:3:]” – it will read items from index “1” (b) and include “b” till index “3” (d) but will exclude “d”.

“[-3:-1:]” – it will read items from index “-3” (b) and include “b” till index “-1” (d) but will exclude “d”.

“[::-1]” – it will read items from index “-1” (d) end till start as other values are not provided.

“[-1:-4:-1]” - it will read items from index “-1” (d) end till start (a) but exclude “a”, in reverse order because “-1” value is present in “step”.

“[3:-4:-1]” - it will read items from index “3” (d) end till index “-4” (a) but exclude “a”, in reverse order because “-1” value is present in “step”.