PYTHON PROGRAMMING

MODULE 4 – PART 3

TUPLES AND OPERATIONS ON TUPLES

OUTLINE

- Tuples
- □ Create and Assign
- □ Tuple Packing and Unpacking
- □ Access elements and Slicing
- Basic Operations
- Delete and Update
- Reverse
- Built-in Functions and Methods

Tuples

- Indexed
- Immutable (cannot update, add or delete elements)
- Can have heterogeneous elements (different data types)
- Defined by enclosing elements in parentheses () or simply by separating elements with comma only
- Iterating over elements is faster in a tuple than in a list

Creating Tuple

```
#Empty tuple:
tup=()
#Single element :
#without a comma tup will be considered as integer type
tup=(11,)
#Multiple elements:
tup=(1,2,('a','b'),[3,4])
or
tup=1,2,('a','b'),[3,4] #without parenthesis
```

Tuple Packing and Unpacking

When several tuple values are assigned to a single variable it is known as packing

```
tup=(1,2,('a','b'),[3,4])
```

When packed variable or a tuple is assigned to another tuple of same number of variables, it is known as unpacking

```
(a,b,c,d)=tup
print(a)
print(b)
print(c)

#each variable in tuple can be accessed individually
print(d)
OUTPUT
1
2
('a', 'b')
[3, 4]
```

Swapping made easy with tuple unpacking:

Accessing tuple and its elements

```
tup=(1,2,('a','b'),[3,4])
print(tup)
print(tup[0])
print(tup[2:4])
print(tup[-4:-2])
print(tup[:])
(1, 2, ('a', 'b'), [3, 4])
(('a', 'b'), [3, 4])
(1, 2)
(1, 2, ('a', 'b'), [3, 4])
```

Basic Operations

len(), concatenation(+), repetition(*), membership(in, not in), iteration

```
s=(1,2,3)
t=(1,2,3)
print(len(s))
print(s+t)
print(s*2)
for i in s:
    print(i)
print(1 in s)
print(1 not in s)

OUTPUT
3
(1, 2, 3, 1, 2, 3)
(1, 2, 3, 1, 2, 3)
(1, 2, 3, 1, 2, 3)
True
False
```

Delete and update (mutable elements only)

```
tup=(1,2,3)

'''shows error,
tuple is immutable'''
del tup[1]

#deletes whole tuple
del tup

tup=(1,2,('a','b'),[3,4]

'''deletes since this
element is a list which
is mutable'''
del tup[3][0]
print(tup)

OUTPUT
(1, 2, ('a', 'b'), [4])
```

```
4]
h
```

```
tup=(1,2,('a','b'),[3,4])

'''shows error, tuple is
immutable'''
tup[1]=3

tup=(1,2,('a','b'),[3,4])

'''updates, this tuple
element is a list which
is mutable'''
tup[3][0]=4
print(tup)

OUTPUT
(1, 2, ('a', 'b'), [4, 4])
```

Reverse of a tuple (using slicing and reversed())

```
tup=(98,9,86,7644,7)
print(tup)
                                    <u>OUTPUT</u>
#slicing
                                    (98, 9, 86, 7644, 7)
print(tup[::-1])
                                    (7, 7644, 86, 9, 98)
                                    (98, 9, 86, 7644, 7)
print(tup)
                                    (7, 7644, 86, 9, 98)
#reversed()
print(tuple(reversed(tup)))
```

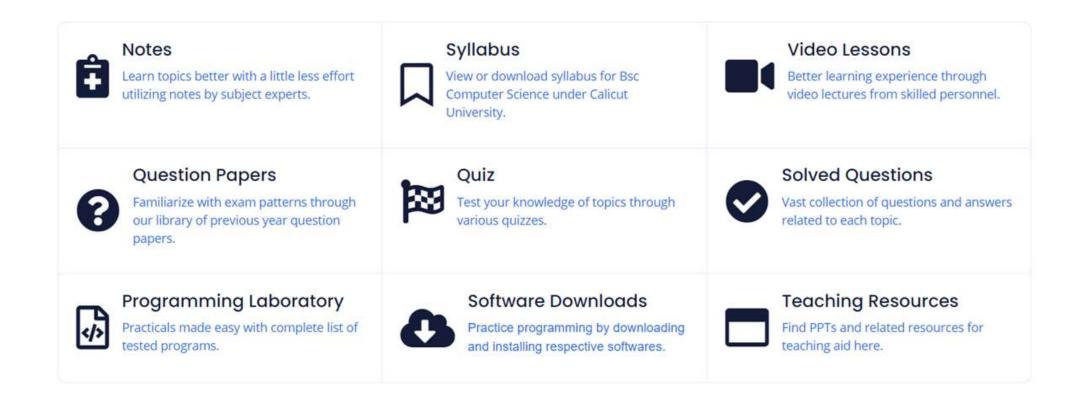
Functions Discussed

- 1) zip(iterable1, iterable2,...)
- 2) max(_tuple_)
 3) min(_tuple_)
- 4) tuple(sequence)
- 5) index(element) 6) count(element)

Tuple built-in functions and methods

```
s=(1,2,3)
t=(1,2,3)
a="hello"
'''takes 0 or more iterables and
                                                                      OUTPUT
combines corresponding elements into tuples.
                                                                      ((1, 1), (2, 2), (3, 3))
Must be passed to tuple, list or set to display resulting tuples.'''
print(tuple(zip(s,t)))
                                                                      ('h', 'e', 'l', 'l', 'o')
print(max(s), min(s))
#converts given sequence to tuple
print(tuple(a))
tup=((1,2,2,2, ('a', 'b'), [3, 4]))
print(tup.index(2))
print(tup.count(2))
```

Study at Home



www.teachics.org

teachics.

The Computer Science Learning Platform.

Thank You