Tuples

Tuples

- Sequence of **immutable (unchangeable)** Python objects
- **Tuples cannot be changed** like lists and tuples use **parentheses ( )**, whereas lists use square brackets.
- Creating a tuple is as simple as putting different **comma-separated values**.
- Optionally you can put these comma-separated values between parentheses also.

For example

```
tup1 = ('physics', 'chemistry', 1997, 2000, 12.5);
tup2 = (1, 2, 3, 4, 5,’a’,’raju’ );
tup3 = "a", "b", "c", "d";
```

empty tuple

```
tup1 = ();
```

To write a tuple containing a **single value you have to include a comma:**

```
a = (50)                # an integer
tup1 = (50,);           # tuple containing an integer
```

tuple indices start at 0

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );


print ("tup1[0]: ", tup1[0])        # print physics

print ("tup2[1:5]: ", tup2[1:5])        # print (2,3,4,5)
```

Tuples in Action

```
>>> (1, 2) + (3, 4)          # Concatenation
(1, 2, 3, 4)

>>> (1, 2) * 4               # Repetition
(1, 2, 1, 2, 1, 2, 1, 2)

>>> T = (1, 2, 3, 4)

>>> T1=T[0], T[1:3]              # Indexing, slicing
>>>T1
(1, (2, 3))
```

Sorted method in Tuples

```
>>> tmp = ['aa', 'bb', 'cc', 'dd']

>>> T = tuple(tmp)    # Make a tuple from the list's items

>>> T

('aa', 'bb', 'cc', 'dd')

>>> sorted(T)

['aa', 'bb', 'cc', 'dd‘]
```

List comprehensions can also be used with tuples.

The following, for example, makes a list from a tuple, **adding 20** to each item along the way:

```
>>> T = (1, 2, 3, 4, 5)

>>> L = [x + 20 for x in T]
>>> L
[21, 22, 23, 24, 25]
```

Equivalent to:

```
>>> T = (1, 2, 3, 4, 5)
>>>L = []
>>>for x in T:
        L.append(x+20)
>>> L
[21, 22, 23, 24, 25]
```

Look the differences:

```
>>> t4=('bb', 'aa', 'dd', 'cc')

>>> L1 = [t4]
>>> L1
[('bb', 'aa', 'dd', 'cc')]
>>> print(L1[0])
('bb', 'aa', 'dd', 'cc')
>>> L2=[x for x in t4]          =>   L2=list(t4)
>>>L2
['bb', 'aa', 'dd', 'cc']
```

Index method can be used to find the **position of particular value** in the tuple.

```
>>> T = (1, 2, 3, 2, 4, 2)
>>> T.index(2)          # Offset of first appearance of 2
1

>>> T.count(2)            # How many 2s are there?
3
```

Nested Tuples

```
>>> T = (1, [2, 3], 4)
>>> T[1] = 'spam'
 # fails: can't change tuple itself
TypeError: object doesn't support item assignment
>>> T[1][0] = 'spam'
# Works: can change mutables inside
>>> T
(1, ['spam', 3], 4)
```

```
>>> bob = ('Bob', 40.5, ['dev', 'mgr'])
```

Tuple record

```
>>> bob
('Bob', 40.5, ['dev', 'mgr'])
>>> bob[0], bob[2]                    # Access by position
('Bob', ['dev', 'mgr'])
>>> t1=bob[0], bob[2]
>>t1
('Bob', ['dev', 'mgr'])
```

# Prepares a Dictionary record from tuple

```
>>> bob = dict(name='Bob', age=40.5, jobs=['dev', 'mgr'])

>>> bob
{'jobs': ['dev', 'mgr'], 'name': 'Bob', 'age': 40.5}

>>> bob['name'], bob['jobs']        # Access by key
('Bob', ['dev', 'mgr'])
```

Dictionary to Tuple

- We can convert parts of dictionary to a tuple if needed:

```
>>> tuple(bob.values())                # Values to tuple
(['dev', 'mgr'], 'Bob', 40.5)
>>> tuple(bob.items())
(('name', 'Bob'), ('age', 40.5), ('jobs', ['dev', 'mgr']))
>>> list(bob.items())                  # Items to list of tuples
[('jobs', ['dev', 'mgr']), ('name', 'Bob'), ('age', 40.5)]
```

Using Tuples

- **Immutable** which means you **cannot update** or change the values of tuple elements

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
tup1[0] = 100;
```

- You are able to take portions of existing tuples to create new tuples as the following example demonstrates

```
tup3 = tup1 + tup2;
print (tup3)
```

Delete Tuple Elements

- **Removing individual tuple elements** is **not possible**
- But possible to **remove** an **entire tuple**

```
tup = ('physics', 'chemistry', 1997, 2000);
print (tup)
del tup;
print ("After deleting tup : ")
print (tup)
Error
```

How to do modification in Tuple

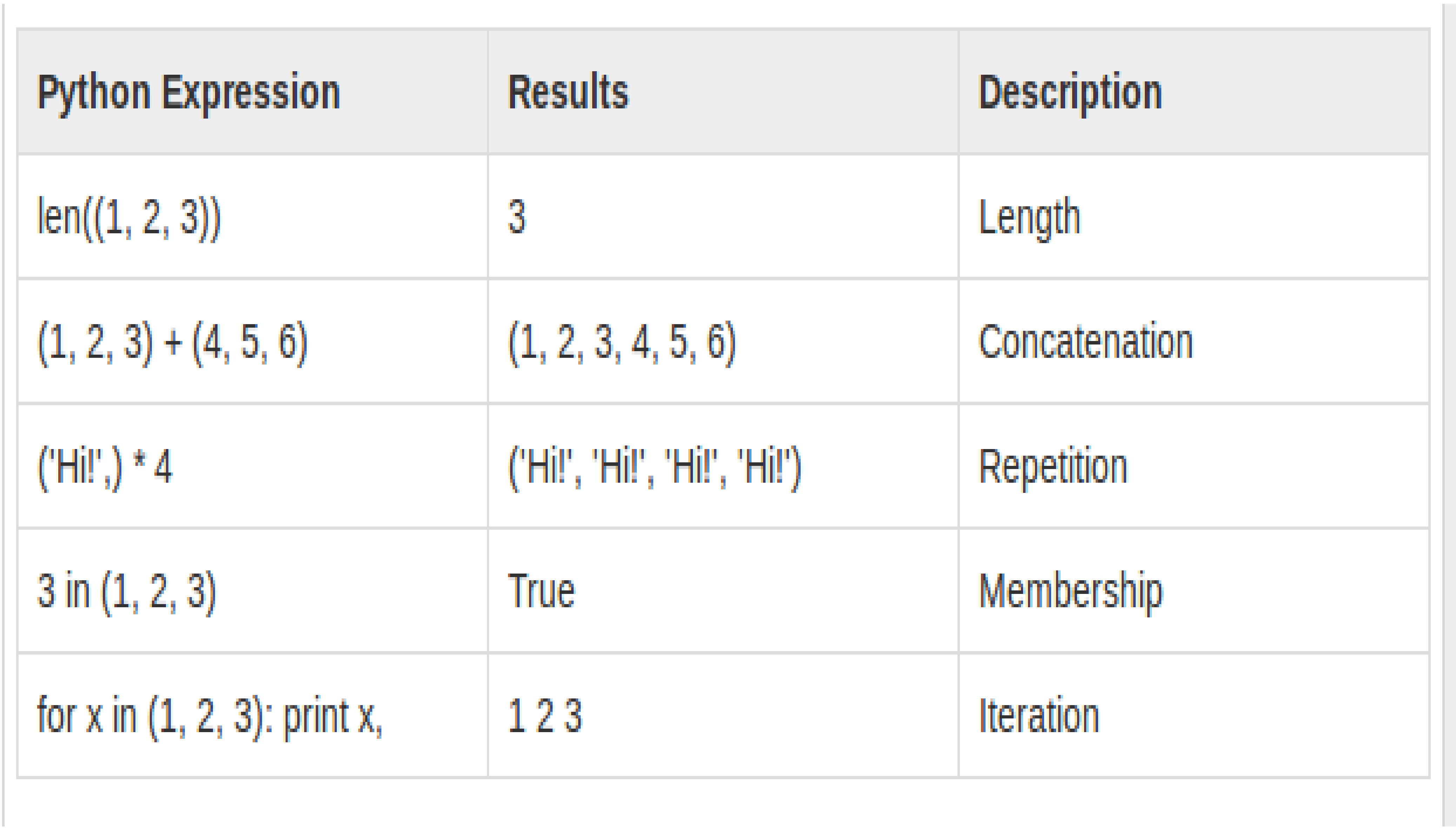
Python tuple is an immutable object.
Hence any operation that tries to modify it (like append, delete) is not allowed.
Then HOW?,

- Convert tuple to list by built-in function list().
- Append item to list object
- Use built-in function tuple() to convert this list object back to tuple.

Ex.

```
>>> T1=(10,50,20,9,40,25,60,30,1,56)
>>> L1=list(T1)
>>> L1
[10, 50, 20, 9, 40, 25, 60, 30, 1, 56]
>>> L1.append(100)
>>> T1=tuple(L1)
>>> T1
(10, 50, 20, 9, 40, 25, 60, 30, 1, 56, 100)
```

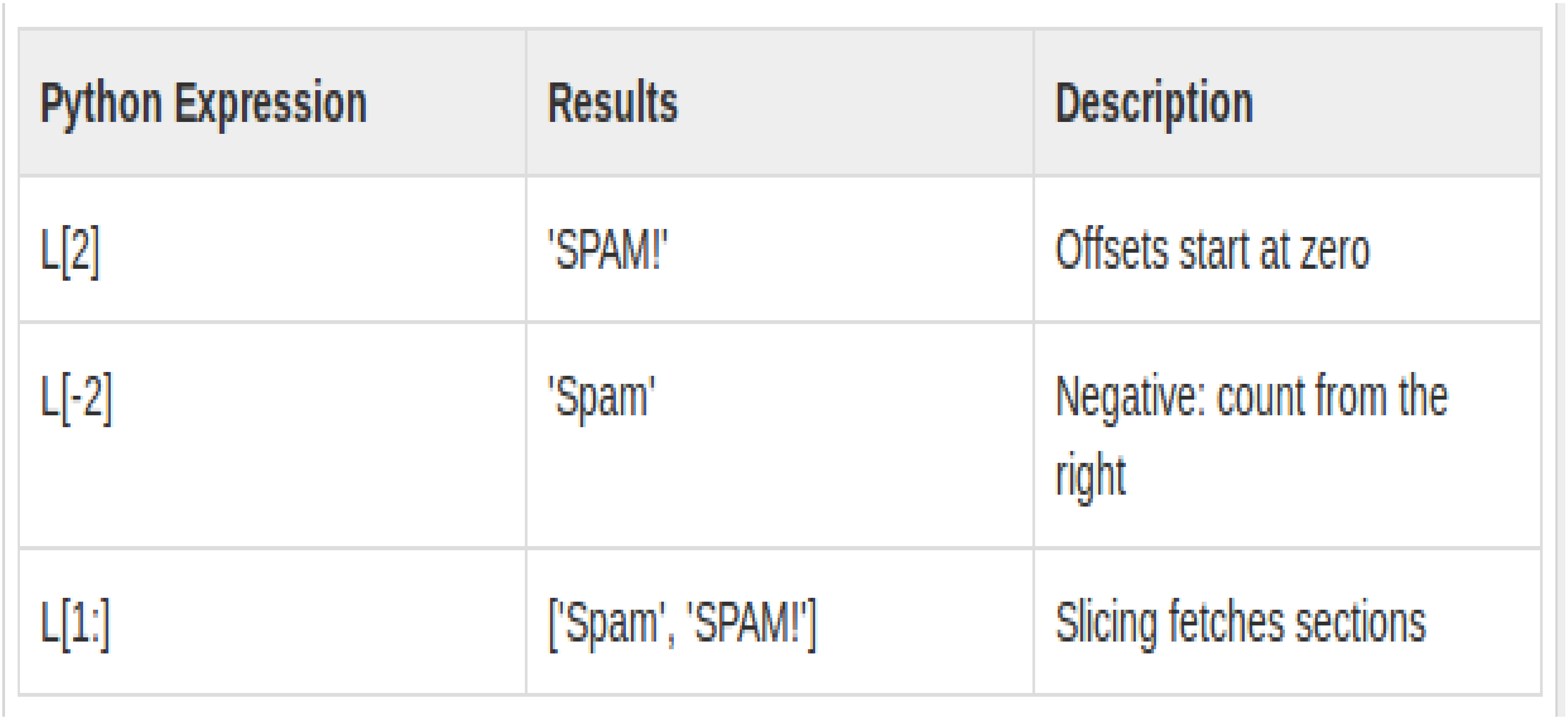
Basic Tuples Operations



Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Indexing, Slicing

L = ('spam', 'Spam', 'SPAM!')



Python Expression	Results	Description
L[2]	'SPAM!'	Offsets start at zero
L[-2]	'Spam'	Negative: count from the right
L[1:]	['Spam', 'SPAM!']	Slicing fetches sections

Built-in Tuple Functions

```
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')
```

```
len(tuple1)
2
```

When we have numerical tuple:

```
t1 = (1,2,3,7,4)
```

```
max(t1)    #prints 7

min(t1)    #prints 1
```

List of Tuples

```
>>>a=[]
>>>b=(2,5),(3,6)
>>>a=[b]
>>>a
[((2, 5), (3, 4))]

>>> q1=1,2,3
>>> q1
(1, 2, 3)
```

q1 is a tuple