Introducción a Python

Python List

List is an ordered sequence of items. It is one of the most used datatype in Python and is very flexible. All the items in a list do not need to be of the same type.

Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [ ].

```
>>> a = [1, 2.2, 'python']
```

We can use the slicing operator [ ] to extract an item or a range of items from a list. Index starts form 0 in Python.

script.py | IPython Shell

```
a = [5,10,15,20,25,30,35,40]

# a[2] = 15
print("a[2] = ", a[2])

# a[0:3] = [5, 10, 15]
print("a[0:3] = ", a[0:3])

# a[5:] = [30, 35, 40]
print("a[5:] = ", a[5:])
```

Lists are mutable, meaning, value of elements of a list can be altered.

```
>>> a = [1,2,3]
>>> a[2]=4
>>> a
[1, 2, 4]
```

Python Tuple

Tuple is an ordered sequence of items same as list.The only difference is that tuples are immutable. Tuples once created cannot be modified.

Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically.

It is defined within parentheses () where items are separated by commas.

```
>>> t = (5,'program', 1+3j)
```

We can use the slicing operator [] to extract items but we cannot change its value.

script.py | IPython Shell

```
t = (5,'program', 1+3j)

# t[1] = 'program'
print("t[1] = ", t[1])

# t[0:3] = (5, 'program', (1+3j))
print("t[0:3] = ", t[0:3])

# Generates error
# Tuples are immutable
t[0] = 10
```

Python Strings

String is sequence of Unicode characters. We can use single quotes or double quotes to represent strings.

```
>>> s = "This is a string"
```

Like list and tuple, slicing operator [ ] can be used with string. Strings are immutable.

script.py | IPython Shell

```
s = 'Hello world!'

# s[4] = 'o'
print("s[4] = ", s[4])

# s[6:11] = 'world'
print("s[6:11] = ", s[6:11])

# Generates error
# Strings are immutable in Python
s[5] ='d'
```

We can perform set operations like union, intersection on two sets. Set have unique values. They eliminate duplicates.

```
>>> a = {1,2,2,3,3,3}
>>> a
{1, 2, 3}
```

Since, set are unordered collection, indexing has no meaning. Hence the slicing operator [] does not work.

```
>>> a = {1,2,3}
>>> a[1]
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
TypeError: 'set' object does not support indexing
```

Logical operators

Logical operators are the `and`, `or`, `not` operators.

Logical operators in Python

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Logical Operators in Python

```
x = True
y = False

# Output: x and y is False
print('x and y is',x and y)

# Output: x or y is True
print('x or y is',x or y)

# Output: not x is False
print('not x is',not x)
```