Introduction to Python

Data types

Apart from numeric, string and boolean data types, Python supports storing of collection of values. This built-in data structures can hold any **collection** of objects. These are:

1. List – []
2. Tuple – ()
3. Set- {}
4. Dictionary – {key:value,…}

<u>List</u>

It is an ordered mutable collection of elements. It can be considered as an array in other programming language. It can be **heterogeneous** that means List can have elements of different data types (integer, string, etc). So it is useful in grouping mixed data types. List are **mutable** objects i.e. its elements can be modified. It has elements kept in an **ordered** fashion. List are dynamic object as number of elements can increased or decreased.

Introduction to Python

List

The **list** is initialized by:

a = list()

a = []

a = [1,2,3] (elements are enclosed in square brackets [] and separated by ‘,’)

```
varA=[]
print(type(varA))

varB=list()
print(type(varA))

varC=[1,2,3,4]
print(type(varC))
```

```
<class 'list'>
<class 'list'>
<class 'list'>
```

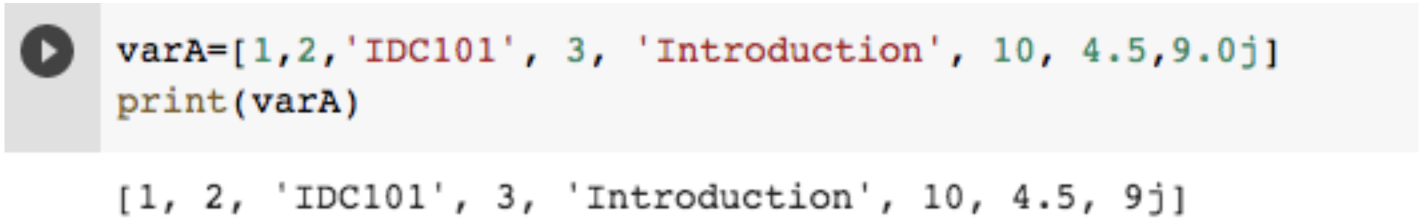
Ordered collection of elements

```
varA=[1,2,3,4]
varB=[3,4,1,2]
varC=[1,2,3,4]

## Content of list varA and varB but are not same as per list

print(varA==varB)

print(varC==varA)
```

```
False
True
```

Introduction to Python

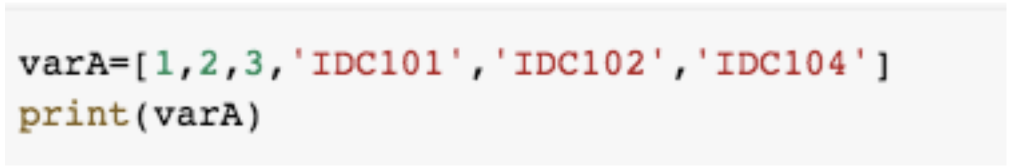
Mixed data types



```
varA=[1,2,'IDC101', 3, 'Introduction', 10, 4.5,9.0j]
print(varA)
```

```
[1, 2, 'IDC101', 3, 'Introduction', 10, 4.5, 9j]
```

Mutable



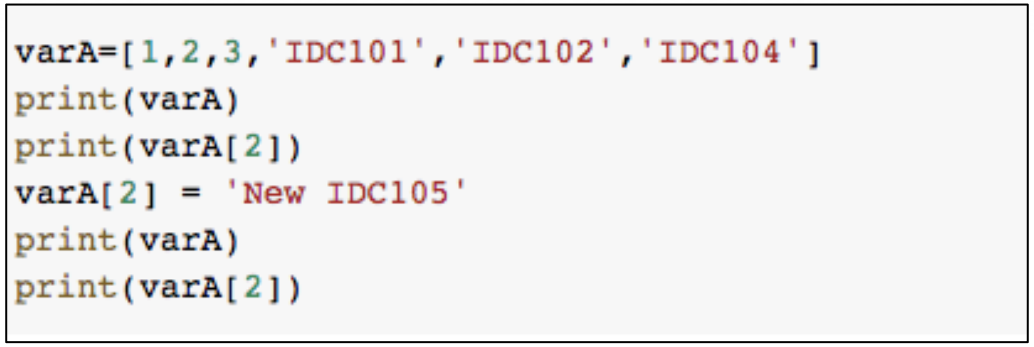
```
varA=[1,2,3,'IDC101','IDC102','IDC104']
print(varA)
```

Access elements of list using index, same as defined for string.

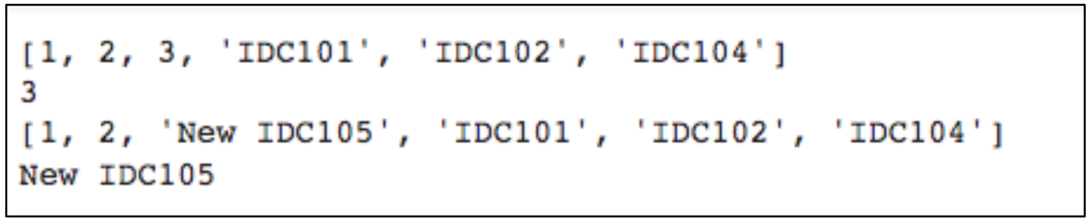
Indexes	0	1	2	3	4	5
List elements	1	2	3	IDC101	IDC102	IDC104
Negative Indexes	–5	–4	–4	–3	–2	–1

Introduction to Python

Mutable



```
varA=[1,2,3,'IDC101','IDC102','IDC104']
print(varA)
print(varA[2])
varA[2] = 'New IDC105'
print(varA)
print(varA[2])
```



```
[1, 2, 3, 'IDC101', 'IDC102', 'IDC104']
3
[1, 2, 'New IDC105', 'IDC101', 'IDC102', 'IDC104']
New IDC105
```

Access multiple elements of list is using slices following the syntax:

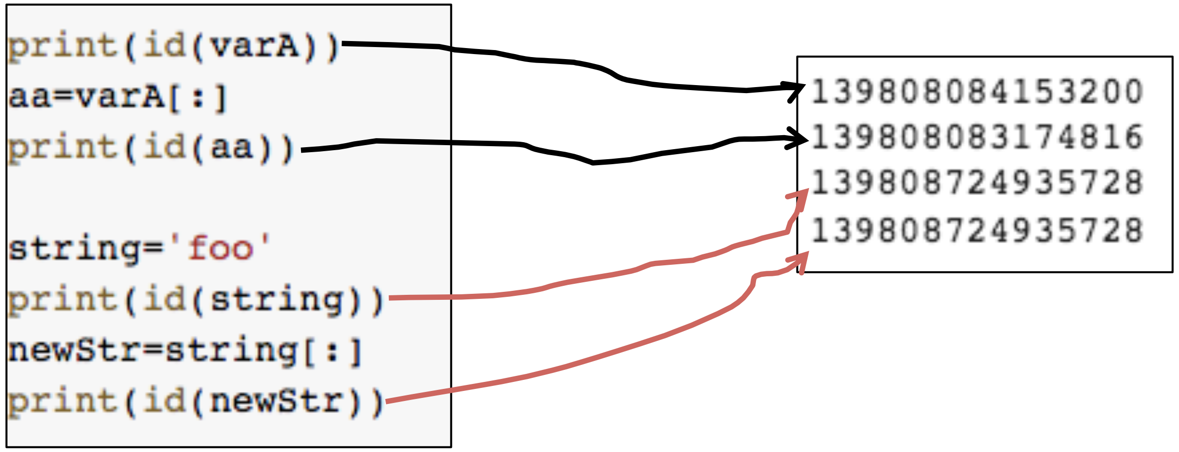
listVariable[startIndex:stopIndex:step]

Everything is same as in string **BUT, when we use list with [:], it returns a new object!!** unlike string

Introduction to Python

Mutable

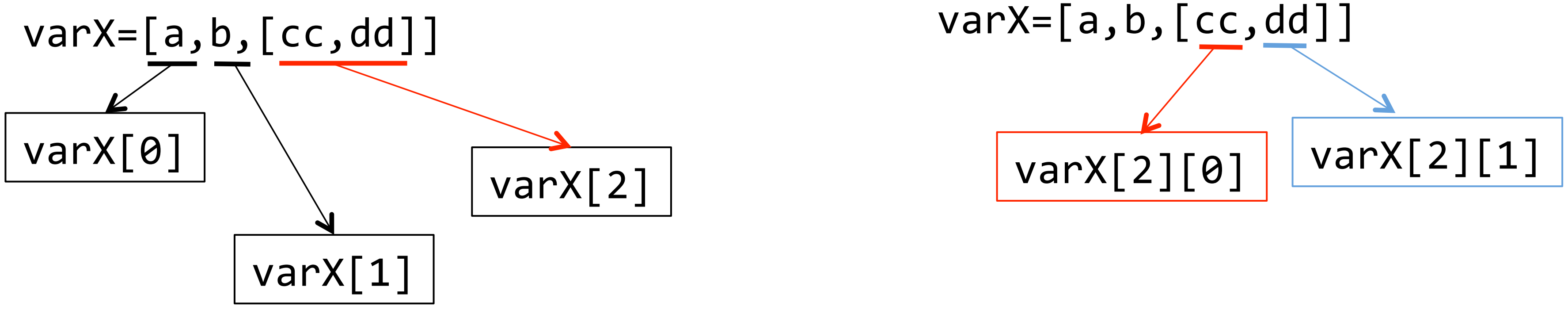
Everything is same as in string **BUT, when we use list with [:], it returns a new object!!** unlike string



```
print(id(varA))
aa=varA[:]
print(id(aa))

string='foo'
print(id(string))
newStr=string[:]
print(id(newStr))
```

```
139808084153200
139808083174816
139808724935728
139808724935728
```

Nested list



Introduction to Python

List operations

Concatenation - (+)
Repetition - (*)
Membership – x in xList (find whether x is member of xList). In nested list, the nested element need to checked in appropriate list element index
del varList – deletes the List varList
len(varList) – returns the length of list
varList.append() – appends list with one variable, if a list is provided it becomes sublist in a list
varList.extend() – appends the list with ANOTHER list
varList.index() – returns the index of the first occurrence of element in bracket
varList.pop() – By default use returns the last element in the list (LILO), this also shortens the list!! If, an index is specified that elements if removed.

for j in varList – iterates elements in varList

for j in range(len(varList)) – iterates elements in varList **but** access value by index

Tuples

It is an ordered mutable collection of elements. It can be considered as an array in other programming language. It can be **heterogeneous** that means List can have elements of different data types (integer, string, etc). So it is useful in groping mixed data types. List are **immutable** objects **i.e.** elements **cannot** be modified after initializing a tuple. It has elements kept in an **ordered** fashion.

The **tuple** is initialized by:

```
a = tuple()
a = ()
```

Elements are enclosed in brackets () and separated by ‘,’

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

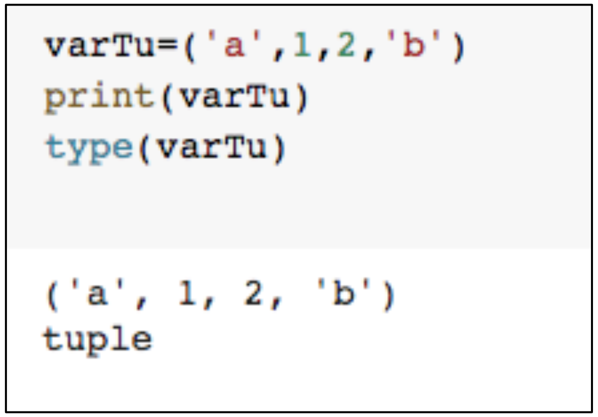
Tuple initialized with single element must be followed by ‘,’ else it will take primitive data type.

```
a=(1,)
```

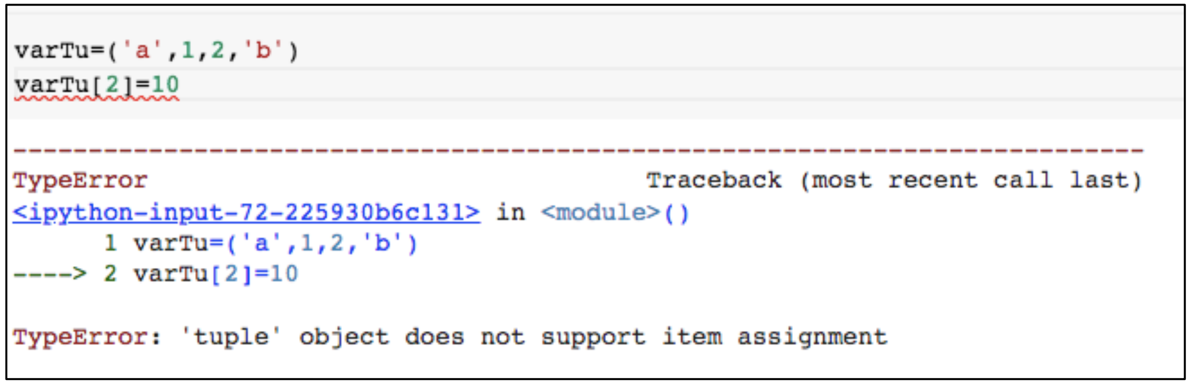
Introduction to Python

Immutable

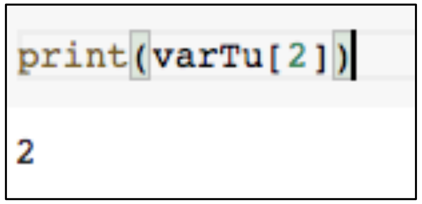
The tuple, indexing, slicing, membership, iteration, concatenation and repetition are as defined in list.



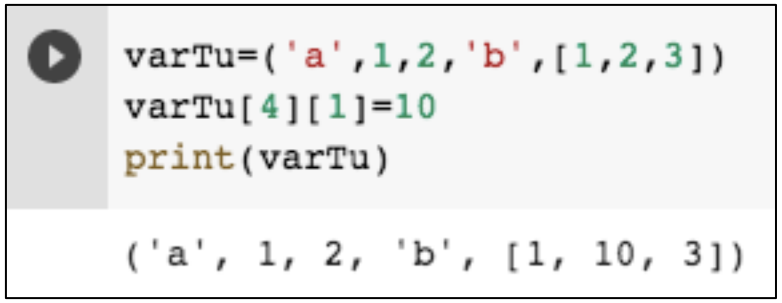
```
varTu=('a',1,2,'b')
print(varTu)
type(varTu)

('a', 1, 2, 'b')
tuple
```



```
varTu=('a',1,2,'b')
varTu[2]=10

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-72-225930b6c131> in <module>()
      1 varTu=('a',1,2,'b')
----> 2 varTu[2]=10

TypeError: 'tuple' object does not support item assignment
```



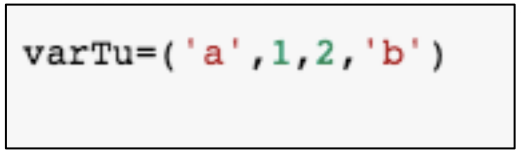
```
print(varTu[2])

2
```



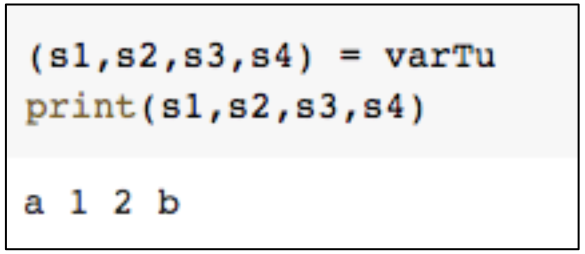
```
varTu=('a',1,2,'b',[1,2,3])
varTu[4][1]=10
print(varTu)

('a', 1, 2, 'b', [1, 10, 3])
```

Introduction to Python

Packing and unpacking tuple

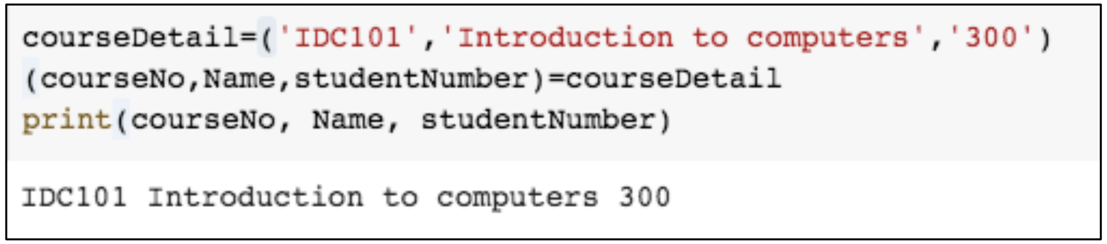


```
varTu=('a',1,2,'b')
```



```
(s1,s2,s3,s4) = varTu
print(s1,s2,s3,s4)
```

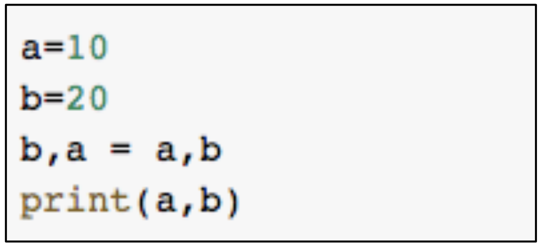
```
a 1 2 b
```



```
courseDetail=('IDC101','Introduction to computers','300')
(courseNo,Name,studentNumber)=courseDetail
print(courseNo, Name, studentNumber)
```

```
IDC101 Introduction to computers 300
```

Swapping



```
a=10
b=20
b,a = a,b
print(a,b)
```