- **created by**: Sudip Ghimire
- **URL**: https://www.sudipghimire.com.np
- **GitHub**: https://github.com/ghimiresdp

go to course contents

Chapter 3.1. Lists

Table of Contents

Ref: https://docs.python.org/3.9/tutorial/introduction.html#lists

Python list is a compound data type that can store collection of data. List is written as comma separated values/items inside large brackets `[]`.

Some of the features of list are as follows:

1. Lists can store multiple items in a single variable
2. Lists are created using Square brackets `[]`
3. Items in list are ordered. i.e. they can be accessed using indexes.
4. The first item of list has index of 0
5. List with n number of items has last index of (n-1)
6. Lists are different than Arrays in other programming languages.
   - lists can have more than one type of data
   - lists are mutable

- lists do not need data to be unique
- In python, list items can also be accessed using negative indexes.

Creating a list

To create a list, we can simply add items separated by comma inside large brackets.

```
# list of numbers
numbers = [18, 24, 56, 21, 44, 27, 99, 100, 64, 44]

# list of strings
animals = ['cat', 'dog', 'tiger']

# list of multiple data types
random_lists = [1, 'John', 'Moon', True, 45.62]
```

In case of a list, new line do not terminate the statement until brackets are closed.

```
animals = ['cat', 'dog', 'tiger']

# can also be written as

animals_1 = [
    'cat',
    'dog',
    'tiger'
]
```

Accessing an item of a list

we can use index of an item inside a large bracket to access an item from the list. The index of the first item starts at `0` and increases as it moves towards the next item.

To access `cat` from the above list, we have to write the following statement

```
# assigning the value to a new variable
x = animals[0]

# printing the value of the item
print(animals[0])
```

We can also use negative index to access index of the list in reverse order.

For example, If we want to access the last item, we can use the index `-1`. Similarly for second last item, we can use the index `-2`.

```
print(animals[-1])  # tiger
```

Getting the slice of a list

To access a slice of items from the list, we can use start and end of the index inside large brackets separated by a colon `:`. In this case, starting index will be inclusive in the slice whereas ending index will be exclusive.

For example:

```
prime = [2, 3, 5, 7, 11, 13, 17, 19]

# Accessing [3, 5, 7]
slice_1 = prime[1:4]
```

Accessing first 5 items from the list:

```
slice_2 = prime[0:5]  # this takes out a slice from index 0 to 4
```

Note:: *If we do not provide the starting index, the slice will be taken from the first index and if we do not provide the ending index, the slice will be taken up to the last index of the list.*

Example:

```
prime = [2, 3, 5, 7, 11, 13, 17, 19]

# The following statement slices the list from first item up to index of 2
print(prime[:3])  # [2,3,5]

# The following statement slices the list from item of index 5 up to the last
item.
print(prime[5:])  # [13, 17, 19]

# The following statement takes all items since starting and ending indices are
not provided.
print(prime[:])  # [2, 3, 5, 7, 11, 13, 17, 19]
```

Updating values from a list

We can easily update a value from the list using its indices

```
list_1 = ['Atanasoff', 'Berry', 'Calculator']

# the last item should be Computer, so I need to change it
```

```
list_1[2] = 'Computer'

print(list_1)

# ['Atanasoff', 'Berry', 'Computer']
```

Updating a slice of a list

Sometimes, we want to update a slice of a list. In this case, updating each item from the index is more frustrating. Fortunately, Python has the capability to update a slice of a list.

```
prime_numbers = [2, 3, 5, 6, 12, 13, 17]
# I wrote 6, 12 instead of 7, 11 by mistake so I want to change them
prime_numbers[3:5] = [7, 11]
print(prime_numbers)
# [2, 3, 5, 7, 11, 13, 17]
```

We can even update the slice with less or more number of elements on the list.

```
odd_num = [1, 3, 5, 7, 8, 10, 15, 17]
# I wrote 8, 10 instead of 9, 11, 13, so I want to change them
odd_num[4:6] = [9, 11, 13]
print(odd_num)
# [1, 3, 5, 7, 9, 11, 13, 15, 17]
```

Finding out the length of a list

We can either use `len()` function or a `__len__()` method of a `list` to find out the length of the list.

```
odd = [1, 3, 5, 7, 9, 11, 13, 15, 17]

length = len(odd)
print(length)
# 9

# or
length = odd.__len__()
print(length)
# 9
```

Adding items to the list

We have multiple methods to add item to the list. Some of them are as follows

The `list.append()` method

We can use `list.append()` method to add new item to the list. With this method, a new item is always added at the end of the existing list.

```
animals = ['cat', 'dog', 'tiger']
animals.append('monkey')
print(animals)
# ['cat', 'dog', 'tiger', 'monkey']
```

if we want to add new item to the list at our desired index, then we should use `list.insert()` method instead

The `list.insert()` method

The `insert()` method is more flexible than the `append()` method, but we are allowed to pass an index where we want to add an item.

```
animals = ['cat', 'dog', 'tiger']
animals.insert(2, 'monkey')
print(animals)
# ['cat', 'dog', 'monkey', 'tiger']
# previously it was ['cat', 'dog', 'tiger', 'monkey']
```

The `list.extend()` method

If we want to add more than one item to the list at once, then we can use `list.extend()` method.

```
animals = ['cat', 'dog', 'tiger']
more_animals = ['Chimpanzee', 'Fox', 'Deer']
animals.extend(more_animals)
print(animals)
# ['cat', 'dog', 'tiger', 'Chimpanzee', 'Fox', 'Deer']
```

Removing items from the list

Similar to appending items, removing items is also possible with various ways. Some of them are as follows:

The `list.pop()` method

Similar to `list.append()`, the `pop()` method pops out the last item from the list if we pass no arguments to the method.

```
animals = ['cat', 'dog', 'tiger', 'Chimpanzee', 'Fox']
animals.pop()
```

```
print(animals)
# ['cat', 'dog', 'tiger', 'Chimpanzee']
```

If we want to remove an item at nth index, then we need to specify an index of the item.

```
animals = ['cat', 'dog', 'tiger', 'Chimpanzee', 'Fox']
# I want to remove 'tiger' which is at index 2
animals.pop(2)
print(animals)
# ['cat', 'dog', 'Chimpanzee', 'Fox']
```

The `list.remove()` method

Sometimes, we do not know the index of an item we want to remove. At this point, we can use `remove()` method. With this method, we can directly pass the value as an argument.

```
animals = ['cat', 'dog', 'tiger', 'Chimpanzee', 'Fox']
animals.remove('dog')
print(animals)
# ['cat', 'tiger', 'Chimpanzee', 'Fox']
```

The `list.clear()` method

The `clear()` method removes all items from the list and makes it an empty list.

```
animals = ['cat', 'dog', 'tiger', 'Chimpanzee', 'Fox']
animals.clear()
print(animals)
# []
```

Some Useful List Methods

`copy()`

generally whenever we assign a value of list to another one, it references the older list. Whenever we try to change the content of a list, it eventually changes the value of the older list too. to avoid this, we need to create a shallow copy of the list. To do so, we need to use `copy()` method.

```
list1 = [1, 2, 3, 4, 5]
print('original list: ', list1)
list2 = list1
print('list 2 before modification: ', list2)
list2.pop()
print('list 2 after modification: ', list2)
print('list 1 after modification: ', list1)

# original list:  [1, 2, 3, 4, 5]
# list 2 before modification:  [1, 2, 3, 4, 5]
# list 2 after modification:  [1, 2, 3, 4]
# list 1 after modification:  [1, 2, 3, 4]
```

Here, we can see, `list2.pop()` method affects `list1` too. but if we use `copy()` method, it does not affect the original list.

```
list1 = [1, 2, 3, 4, 5]
print('original list: ', list1)
list2 = list1.copy()
print('list 2 before modification: ', list2)
list2.pop()
print('list 2 after modification: ', list2)
print('list 1 after modification: ', list1)

# original list:  [1, 2, 3, 4, 5]
# list 2 before modification:  [1, 2, 3, 4, 5]
# list 2 after modification:  [1, 2, 3, 4]
# list 1 after modification:  [1, 2, 3, 4, 5]
```

`reverse()`

We can reverse the list using `reverse()` method.

```
list1 = [1, 2, 3, 4, 5]
list1.reverse()
print(list1)
# [5, 4, 3, 2, 1]
```

`sort()`

We can sort the list in ascending or descending order

```
list1 = [1, 5, 2, 7, 3, 9, 0, 6]
list1.sort()
print(list1)
# [0, 1, 2, 3, 5, 6, 7, 9]

list1.sort(reverse=True)
print(list1)
# [9, 7, 6, 5, 3, 2, 1, 0]
```

`index()`

we can find the index of an item of the list using `index()` method.

```
animals = ['cat', 'dog', 'cow', 'donkey']
index = animals.index('dog')
print(index)   # 1
```

`count()`

We can count the number of occurrences of an element of a list using the `count()` method

```
numbers = [1, 4, 2, 6, 4, 3, 2, 6, 4, 3, 1, 2, 5, 3, 2, 7, 4, 2, 8, 2, 10, 5]

count_2 = numbers.count(2)
print(count_2)   # 6
```