Python DAY 1

Chapter 1. Introduction to Python

1.1 Introduction in Brief

Python is an open source programming language. Python can run on any environment and operating system including Windows, Linux, Mac OS, etc. Being trained in Python increases your programming efficiency and you can write function using less coding compared to other programming languages.

1.2 Advantages Of Python

1. Python programming increases the readability of code and helps to write small as well as large and complex functions.

2. Python programming is used to write scripting programs as well as non-scripting programs.

3. Python programming language has a vast library and facilitates programmers developing any program easily.

4. Python is an interpreted language.
When you run a python program an interpreter will parse python programs on a line by line basis, as compared to compiled languages like C or C++, where the compiler first compiles the program and then starts running.

5. Python is Dynamically Typed.
In python there is no need to define variable data types ahead of time, python can automatically guess the data type of the variable based on the type of value it contains.

6. You write less code and do more.
Python codes are usually 1/3 or 1/5 of java code. This means we need lesser code in Python to achieve the same thing when compared to Java.
In python to read a file you only need 1 line:
print(open('D:\\1.txt').read()))

-------Installation

Chapter 2 Variable and Data Types

2.1 Definition of Terms - Variable and Data Types

2.1.1 Variable

A Python variable is used to store values. It is a reserved memory location. To put it simply, a variable in a python program gives data for processing.

2.1.2 Data Types

Data types are the classification of data items. Data types signify the kind of value present in a variable and controls how the value can be used. Numeric, non-numeric and Boolean (true/false) data are the commonly used data types. There are several data types in Python. A few of the important types are listed below.

2.1.3 Numbers

Numeric literals constitute numbers. Numeric literals are immutable, which means that its value cannot be changed. Python has three numeric types: integers, floating point numbers(decimals), and complex numbers.

Example

```
age=38 #
Salary=70000.50 #Float represents fraction points data
Var1=7i #Complex type represents imaginary data
```

2.1.4 Boolean

Boolean is the simplest built-in type in Python. This type can hold only two values: True or False. An expression that defines a logical idea that equates to true or false can be created using this data type.
For example, you could say, isIt = 6 > 5, which would equate to True.

2.1.5 String

A string is a sequence (ordered left-to-right) of characters. Strings are enclosed between single or double quotes. In Python, strings are also immutable. Once a string is created, you cannot modify any character present in the string. Single and double quoted strings are the same. You can also use a single quote within a string, even when it is surrounded by double quotes. Defining a string is simple. For Example, message ="Python is a simple language."

2.1.6 Lists

A list is a container. It holds comma-separated values (items or elements) within square brackets. Lists do not require Items or elements to have the same data type. It is the most commonly used data type because it’s flexible, ordered and allows duplication. It’s similar to arrays as it can contain multiple data types at once as nested objects.

2.1.7 Tuples

A tuple is a series of immutable Python objects. Tuples are also sequences, just like lists. The differences between the two are, the tuple's objects cannot be changed and unlike lists and tuples use parentheses instead of square brackets.

2.1.8 Dictionary

In Python, a dictionary is a container of unordered sets of objects like lists. The objects are enclosed in curly braces { }. The items in a dictionary are a comma-separated list of key:value pairs where keys and values are Python data types. Each object or value is accessed by a key and each key is unique in the dictionary. As keys are used for indexing, they must be an immutable type (string, number, or tuple). You can also create an empty dictionary using empty curly braces.

Some commonly used data types are date and calendar.
Example of datetime.

```
>>> import datetime
>>> print(datetime.datetime.now())
2019-04-11 19:47:18.679253
>>>
```

You can format date pattern as below

```
>>> date=datetime.datetime.now()
>>> date.strftime('%d-%m-%y')
'11-04-19'
>>>
```

The exact year range for which strftime() works varies across different platforms. However, years before 1900 cannot be used.

Directive	Meaning	Example	Notes
%a	Weekday as abbreviated names.	Sun, Mon, …, Sat (en_US); So, Mo, …, Sa (de_DE)	(1)
%A	Weekdays as full names.	Sunday, Monday, …, Saturday (en_US); Sonntag, Montag, …, Samstag (de_DE)	(1)
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, …, 6	
%d	Day of the month as a zero-padded decimal number.	01, 02, …, 31	
%b	Month as abbreviated names.	Jan, Feb, …, Dec (en_US); Jan, Feb, …, Dez (de_DE)	(1)

%B	Month as full names.	January, February, …, December (en_US); Januar, Februar, …, Dezember (de_DE)	(1)
%m	Month as a zero-padded decimal number.	01, 02, …, 12	
%y	Year without century as a zero-padded decimal number.	00, 01, …, 99	
%Y	Year with century as a decimal number.	1970, 1988, 2001, 2013	
%H	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, …, 23	
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, …, 12	
%p	Equivalent of either AM or PM.	AM, PM (en_US); am, pm (de_DE)	(1), (2)
%M	Minute as a zero-padded decimal number.	00, 01, …, 59	
%S	Second as a zero-padded decimal number.	00, 01, …, 59	(3)
%f	Microsecond as a decimal number, zero-padded on the left.	000000, 000001, …, 999999	(4)
%z	UTC offset in the form +HHMM or -HHMM (empty string if the object is naive).	(empty), +0000, -0400, +1030	(5)
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, EST, CST	
%j	Day of the year as a zero-padded decimal number.	001, 002, …, 366	
%U	Week number of the year (Sunday as the first day of the week) as a zero padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.	00, 01, …, 53	(6)

%W	Week number of the year (Monday as the first day of the week) as a decimal number. All days in a new year preceding the first Monday are considered to be in week 0.	00, 01, …, 53	(6)
%c	Appropriate date and time representation.	Tue Aug 16 21:30:00 1988 (en_US); Di 16 Aug 21:30:00 1988 (de_DE)	(1)
%x	Appropriate date representation.	08/16/88 (None); 08/16/1988 (en_US); 16.08.1988 (de_DE)	(1)
%X	Appropriate time representation.	21:30:00 (en_US); 21:30:00 (de_DE)	(1)
%%	A literal '%' character.	%	

Format may differ per OS and time zone.

Calendar is also available in python

```
>>> import calendar
>>> m=calendar.month(2019,4)
>>> print(m)
     April 2019
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30

>>>
```

Full year calendar can be display using code:

```
print(calendar.calendar(2019))
                              2019

      January                   February                  March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6                1  2  3                  1  2  3
 7  8  9 10 11 12 13       4  5  6  7  8  9 10       4  5  6  7  8  9 10
```

14 15 16 17 18 19 20 11 12 13 14 15 16 17 11 12 13 14 15 16 17
21 22 23 24 25 26 27 18 19 20 21 22 23 24 18 19 20 21 22 23 24
28 29 30 31 25 26 27 28 25 26 27 28 29 30 31

2.2 Collections of Data

Python data collections are container data types, namely lists, sets, tuples, dictionary.

1. A list is modifiable, it stores duplicate values and elements. They can be accessed using indexes.

2. A tuple is ordered and unchangeable. Duplicate entries can be present inside a tuple.

3. A set is an unarranged collection of items. A set created using set is mutable while the ones created with frozenset is immutable. A set is not indexed and does not hold duplicate elements.

4. A dictionary has key value pairs and is mutable.
In addition to these, python also has a module named collections which has specialized data structures. The collections module offers high-performance data types that enhances your code and makes things cleaner and easier.

Example

```
>>> names=['Raj','Naaz','Jay','Shiam']
>>> names
['Raj', 'Naaz', 'Jay', 'Shiam']
>>> names.sort()
>>> names
['Jay', 'Naaz', 'Raj', 'Shiam']
>>> names.reverse()
>>> names
['Shiam', 'Raj', 'Naaz', 'Jay']
>>> names.append('Vani')
>>> names
['Shiam', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names.insert(1,'Reev')
>>> names
['Shiam', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names[0]
'Shiam'
>>> names[0:3]
['Shiam', 'Reev', 'Raj']
```

```
>>> names[2:4]
['Raj', 'Naaz']
>>> names[:]
['Shiam', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> nm=names
>>> nm
['Shiam', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> nm[0]='Shan'
>>> nm
['Shan', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names
['Shan', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> nm=names.copy()
>>> nm
['Shan', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names
['Shan', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> nm[0]='Saral'
>>> nm
['Saral', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names
['Shan', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> nm=names[:]
>>> nm
>>> nm[0]='Surya'
>>> nm
['Surya', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names
['Shan', 'Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names.count('Shan')
1
>>> names.count('Shani')
0
>>> len(names)
6
>>> dir(names)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__',
'__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__',
'__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append',
'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>> names.remove('Shan')
>>> names
```

```
['Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names.index('Raj')
1
>>> names
['Reev', 'Raj', 'Naaz', 'Jay', 'Vani']
>>> names[-1]
'Vani'
>>> m=['Raju','Suraj']
>>> names.extend(m)
>>> names
['Reev', 'Raj', 'Naaz', 'Jay', 'Vani', 'Raju', 'Suraj']
>>> names.__add__('Teena')
Traceback (most recent call last):
  File "<pyshell#57>", line 1, in <module>
    names.__add__('Teena')
TypeError: can only concatenate list (not "str") to list
>>> names.__add__(['Teena'])
['Reev', 'Raj', 'Naaz', 'Jay', 'Vani', 'Raju', 'Suraj', 'Teena']
>>> names
['Reev', 'Raj', 'Naaz', 'Jay', 'Vani', 'Raju', 'Suraj']
>>> names=names.__add__(['Teena'])
>>> names
['Reev', 'Raj', 'Naaz', 'Jay', 'Vani', 'Raju', 'Suraj', 'Teena']
>>> names.pop()
'Teena'
>>> names
['Reev', 'Raj', 'Naaz', 'Jay', 'Vani', 'Raju', 'Suraj']
>>> names.__doc__
'Built-in mutable sequence.\n\nIf no argument is given, the constructor creates a new empty
list.\nThe argument must be an iterable if specified.'
>>> type(m)
<class 'list'>
>>> n=[1,344,45,6,56,5,5,6,76767,9]
>>> n
[1, 344, 45, 6, 56, 5, 5, 6, 76767, 9]
>>> n.sort()
>>> n
[1, 5, 5, 6, 6, 9, 45, 56, 344, 76767]
>>> 1 in n
True
>>> n.count(1)
1
>>> n.count(6)
2
```

```
>>> if 1 in n:
        print('Found')
else:
        print('Not Found')

        Found
>>>
>>> if 99 in n:
        print('Found')
else:
        print('Not Found')

        Not Found
>>> a=6
>>> if n.count(a)>0:
        print('Found ',n.count(a),' Times')
else:
        print('Not Found')

        Found  2  Times
>>> a=5
>>> if n.count(a)>0:
        print('Found ',n.count(a),' Times')
else:
        print('Not Found')

        Found  2  Times
```

Task

1. Write a program to print COMPUTER on one line and DEVELEARN on the next line
2. Write a Python program to print the following string in a specific format (see the output). Sample String : "Twinkle, twinkle, little star, How I wonder what you are! Up above the world so high, Like a diamond in the sky. Twinkle, twinkle, little star, How I wonder what you are" Output :

```
Twinkle, twinkle, little star,
   How I wonder what you are!
          Up above the world so high,
          Like a diamond in the sky.
 Twinkle, twinkle, little star,
          How I wonder what you are
```

3. WAP to accept 2 numbers. Interchange the two numbers and then print them.
4. Write a Python program to get the Python version you are using.
5. Write a Python program which accepts the user first and last name and print them in reverse order with a space between them.

6. Write a Python program to display the current date and time.
7. Write a Python program to print the calendar of a given month and year.
   Note : Use 'calendar' module.