UNIT-1

Web resources

https://www.w3schools.com/python
Realpython.com

Python features

- It is a programming language
- Developed by guido van rossum in early 90s
- monty **python** flying circus

- Python is easy to learn as compared to other programming languages.
- lesser code required to solve a problem than c,c++ and java
- data type of a variable depends on value it is holding
- Its syntax is straightforward and much the same as the English language.
- There is no use of the semicolon or curly-bracket
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python is an interpreted language(it means the Python program is executed one line at a time)

To compile and run python code ,it uses "interpreter"(translator)
To compile c,c++ or java they use "compiler"(translator)

Compiled languages

C,c++ ,c# and java

Interpreted languages

Python
Perl
Php

Java script
ruby

- which softwares are using python currently

  youtube
  google(search engine)
  instagram
  spotify
  netflix
  quora
  pinterest
  uber
  bittorrent &
  to develop games

- https://www.python.org

Data Types and variables

```
>>> a=10
>>> type(a)
<type 'int'>
>>> a="bindhu"
>>> type(a)
<type 'str'>
>>> a=7.98
>>> type(a)
<type 'float'>
>>> a='#'
>>> type(a)
<type 'str'>
>>> 6<7
True
>>> 6>9
False
>>> a=True
>>> type(a)
<type 'bool'>
```

```
>>> x,y=10,20
>>> print(x)
10
>>> print(y)
20
>>> x,y,z=2,3,4
>>> print(z)
4
>>> x = y = z = "Orange"
>>> print(z)
```

```
Orange
>>> print(y)
Orange
```

operators

- Arithmetic
  - +
  - -
  - /
  - %
  - //(floor division)
    - 7//2
    - 3
  - **(power)
- relational
  - <
  - <=
  - ==
  - !=
  - >
- logical
  - and
  - or
  - Not

```
>>> 6<7 and 7>9
False
>>> 6<7 and 7<9
True
>>> 6>7 and 7>9
False
>>> 6>7 or 7>9
False
>>> 6>7 or 7<9
True
>>> not 6>7
True
>>> not (6>7 or 7>9)
True
```

- bitwise
  - |
  - &
  - ^
  - <<
  - >>
- Membership operators

in
not in

eg:

```
a='sai'
print('s' in a)
True
```

```
>>> a="halwa"
>>> "h" in a
True
>>> "w" in a
True
>>> "q" in a
False
>>> "w" not in a
False
>>> "u" not in a
True
```

Input and output

```
print("enter ur favourite dish:")
ilike=input()
print("i like:",ilike)
```

```
ilike=input("enter ur favourite dish:")
howmany=input("how many u eat:")
print("u like:",howmany," ",ilike)
```

Type casting:

```
a=int(input("enter num1:"))
b=int(input("enter num2:"))
c=a+b
print("sum of ",a,",",b,"is:",c)
```

```
a=float(input("enter num1:"))
b=float(input("enter num2:"))
c=a+b
print("sum of ",a,",",b,"is:",c)
```

Note

In python there is no character data type .a character is a string of length one.

String variables can be declared either by using single or double quotes

a=”raju” is as same as a=’raju’

```
>>> "sai"*2
'saisai'
>>> "sai"*4
'saisaisaisai'
>>>
```

---

control statements or conditional statements

if-else

elif-ladder

for in

while

```
a=10
if a<9:
    print("hello")
    print("bye")
```

```
a=8
if a<9:
    print("hello")
    print("bye")
else:
      print("fello")
      print("bye")
```

```
a=2
b=2
if a<b:
    print("a<b")

elif a>b:
      print("a>b")

else:
    print("a==b")
```

Iteration(looping)

```
for vada in range(1,10,3):
    print(vada)
```

1,4,7

```
for vada in range(10,0,-2):
    print(vada)
```

10,8,6,4,2

```
i=1
while i<6:
    print(i)
    i=i+1
print("ice")
```

```
i=1
while i<6:
    print(i,end=" kiran ")
    i=i+1
print()
```

1 kiran 2 kiran 3 kiran 4 kiran 5 kiran

Strings operations

String is an array of characters

```
a="drum stick"

>>> a[0]
'd'
>>> a[0:3]
'dru'
>>> a[0:5]
'drum '
#it is called slicing
>>> a[-1]
'k'
>>> a[-4:-1]
'tic'
>>> a[-10:]
'drum stick'
```

```
>>> a[:6]
'drum s'
```

strings are immutable(cant change once assigned)

```
a[0]='s'

#it is not allowed
```

string methods(functions)

```
>>> a="drum stick"
>>> len(a)
10
>>> a.count('m')
1
>>> a.endswith('k')
True
>>> a.endswith('ck')
True
>>> a.find('s')
5
>>> a.isalpha()

False
>>> b="bun"
>>> b.isalpha()
True
>>> a.isdigit()
False
>>> a.upper()
'DRUM STICK'
>>> a.split()
['drum', 'stick']
>>> a.replace('d','s')
'srum stick'
>>> a
'drum stick'
>>> a.isalnum()
False
>>> a.capitalize()
'Drum stick'
>>> a
'drum stick'
>>> "@".join(a)
'd@r@u@m@ @s@t@i@c@k'
>>> "   ".join(a)
'd\tr\tu\tm\t \ts\tt\ti\tc\tk'
>>> a.split(“ “)
['drum', 'stick']
>>> b="i am undergoing surgery"
>>> b.split('r')
['i am unde', 'going su', 'ge', 'y']
>>> b.split(" ")
['i', 'am', 'undergoing', 'surgery']
```

```
>>> a.index('d')
0
>>> a.index('u')
2
```

```
# finding ASCII

>>> chr(48)
'0'
>>> chr(65)
'A'
>>> chr(44)
','
>>> chr(122)
'z'
>>> ord('A')
65
>>> ord(',')
44
>>> ord('z')
122

for tea in range(97,123):
    print(chr(tea))
```

a,b,-----z

ORD() means ordinal position

List

Lists are used to store multiple items(values) in a single variable.

Lists are used to store collections of data, the other 2 are Tuple and Dictionary.

Lists are created using square brackets:

eg:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

A list can contain values of different data types

a=[“sam”,4.5,99]

A list may contain another list

[2,3,[4,5,6],9]

Eg:

```
>>> a=[2,3,[4,5,6],7,8]
>>> a[0]
2
>>> a[2]
[4, 5, 6]
>>> a[2][0]
4
>>> a[2][2]
6
>>> a[-1]
8
>>> a[-3]
[4, 5, 6]
>>> a[-3][-3]
4
>>> a[-3][-1]
6
```

List slicing

```
>>> a[0:3]
[2, 3, [4, 5, 6]]
>>> a[0:2]
[2, 3]
```

```
>>> a[:]
[2, 3, [4, 5, 6], 7, 8]
```

List methods

```
>>> a=[3,4,5,6,7,8,9]
>>> a.append(11)
>>> a
[3, 4, 5, 6, 7, 8, 9, 11]
>>> b=['g','a','t','v','u']
>>> b.sort()
>>> b
['a', 'g', 't', 'u', 'v']
>>> sum(a)
53
>>> a.pop()
11
>>> a
[3, 4, 5, 6, 7, 8, 9]
>>> a.pop(2)
5
>>> a
[3, 4, 6, 7, 8, 9]
>>> len(a)
6
>>> a.remove(3)
>>> a
[4, 6, 7, 8, 9]
[4, 6, 7, 8, 9]
```

```
>>> del a[2:4]
>>> a
[4, 6, 9]
>>> s="hello ramu where are you"
>>> wow=s.split()
>>> wow
['hello', 'ramu', 'where', 'are', 'you']
>>> wow=s.split("r")
>>> wow
['hello ', 'amu whe', 'e a', 'e you']
>>> s=["sai","sam","siri","vini"]
>>> delimiter="@"
>>> q=delimiter.join(s)
>>> q
'sai@sam@siri@vini'
fruits = ['apple', 'banana', 'cherry']
fruits.insert(1, "orange")
['apple', 'orange', 'banana', 'cherry']
>>> y=[3,2,4,5,7,1]
>>> y
[3, 2, 4, 5, 7, 1]
>>> y.clear()
>>> y
[ ]
```

eg:

```
names=["raj","sam","nani","sonu","vini"]
llen=len(names)
```

```
for i in range(llen-1,-1,-1):
        print(names[i])
print()


names=["raj","sam","nani","sonu","vini"]
llen=len(names)
for i in range(0,llen):
        t=list(names[i])
        t.pop()
        s="".join(t)
        names[i]=s
        print(names[i])
print()
```

Tuple

Tuples are used to store multiple items in a single variable.

fruit = ("apple", "banana", "cherry", "apple", "cherry")

Tuple is one of 4 built-in data types in Python used to store collections(set) of data

A tuple is a collection which is ordered and **unchangeable(immutable)**.

Tuples are written with round brackets.

Unchangeable

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Eg:

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

(2, 3, 4, 5, 6)

>>> a[0]=99

'tuple' object does not support item assignment

Allow Duplicates

Since tuple are indexed, tuples can have items with the same value:

>>> b=(2,2,3,3,4,4)

>>> b

(2, 2, 3, 3, 4, 4)

Tuple methods

>>> q=(3,2,5,4,7,6)

>>> q

(3, 2, 5, 4, 7, 6)

>>> q[2:5]

(5, 4, 7)

>>> q.index(3)

0

>>> len(q)

6

>>> q=(3,2,5,4,7,6)

>>> del q[1:3]

'tuple' object does not support item deletion

>>> del q

>>> q

name 'q' is not defined

Looping through tuple

a=(2,4,5,6,7)

```
for i in a:
    print(i)
```

Dictionary

collection of "key-value" pairs

```
siri={"age":26,"sal":20000,"height":5.7}
```

age is key,26 is value

It should not have duplicate keys

eg 1:

```
>>> siri={"age":26,"sal":20000,"height":5.7}

>>> siri["age"]
26
>>> siri["sal"]
20000
>>> siri
{'age': 26, 'height': 5.7, 'sal': 20000}
```

eg 2:

```
>>> a={1:11,2:22,3:33}
```

```
>>> a[1]
11
>>> a[2]
22

>>> a={1:11,2:22,3:33}
>>> a[1]=111  #modifying values
>>> a
{1: 111, 2: 22, 3: 33}
```

methods of dictionary

```
>>> a={1:11,2:22,3:33}
>>> a[1]=111
>>> a
{1: 111, 2: 22, 3: 33}
>>> len(a)
3
>>> a.get(1)
111
>>> a.values()
[111, 22, 33]
>>> a.pop(2)
22
>>> a
{1: 111, 3: 33}
>>> a.keys()
[1, 3]
```

```
>>> a.has_key(2)
False
>>> a.clear()
>>> a
{ }
>>> s={ }
>>> type(s)
<type 'dict'>
```

Looping through dictionary

```
siri={"age":26,"sal":20000,"height":5.7}
for i in siri:
    print("key:",i," ,value:",siri[i])
```

Finding frequency count of each character

```
string="hello,potato i will eat you"
freq={}
for i in string:
    if i in freq:
            freq[i]=freq[i]+1
    else:
            freq[i]=1
print(freq)
```

```
>>> a={}
>>> a['h']=1
>>> a
{'h': 1}
>>> a['h']
1
>>> a['h']=2
>>> a
{'h': 2}
>>> a['y']=8
>>> a
{'y': 8, 'h': 2}
>>> a['u']=90
>>> a
{'y': 8, 'h': 2, 'u': 90}
>>> len(a)
3
>>> a.keys()
['y', 'h', 'u']
>>> a.values()
[8, 2, 90]
>>> a['h']=a['h']+1
>>> a
{'y': 8, 'h': 3, 'u': 90}
>>> a['u']=a['u']+5
```

```
>>> a
{'y': 8, 'h': 3, 'u': 95}
```

Note

```
>>> a=[2,3,4,5]
>>> type(a)
<type 'list'>
>>> a=(2,3,4,5)
>>> type(a)
<type 'tuple'>
>>> a={2,3,4,5}
>>> type(a)
```

<type 'set'>

```
s="win"
d="root"
for i in s:
  if i in d:
        print(i,"is in ",d)
  else:
        print(i,"is not in ",d)

a=[2,4,5,6,7,8]
alen=len(a)
for i in range(alen):
   a[i]=a[i]**2
for i in range(alen):
   print(a[i],end=" ")
print()

food={"poori":"kurma","pani":"poori","idly":"vada","tea":"pakoda"}
for i in food:
        print(i,":",food[i])
```

```
>>> food={"poori":"kurma","pani":"poori","idly":"vada","tea":"pakoda"}
>>> food
{'poori': 'kurma', 'tea': 'pakoda', 'pani': 'poori', 'idly': 'vada'}
```

Type casting

```
d=[3,5,6]
>>> d
[3, 5, 6]
>>> type(d)
<type 'list'>
>>> tuple(d)
(3, 5, 6)
>>> d
[3, 5, 6]
>>> d=tuple(d)
>>> d
(3, 5, 6)
>>> type(d)
<type 'tuple'>
>>> d=list(d)
>>> d
[3, 5, 6]
>>> type(d)
<type 'list'>
>>> g="i like eating"
>>> z=list(g)
>>> z
['i', ' ', 'l', 'i', 'k', 'e', ' ', 'e', 'a', 't', 'i', 'n', 'g']
```

```
names=["raj","sam","nani","sonu","vini"]
llen=len(names)
```

```
for i in range(0,llen):
        t=list(names[i])
        t.pop()
        s="".join(t)
        names[i]=s
print(names)
```