Day Objectives:

Python Basics

- How to check your Python version
- Comments in Python (#, ctrl+/)

Data Types:

```
A. int,float,bool<br>
B. List,tuple,string,Dictionary, set
```

- variables
- Type Casting

OPERATORS

```
A. Arithmetic Operators ( +,-,*,%,/,//,**)

B. Assignment Operators ( =, +=, -=, *=, %=, /=, //=)

C. Comparison operators ( ==, >=, <=, !=, >, < )

D. Logical operators  ( and, or, not )

E. Bitwise Operators ( &, |, ~, ^, >>, <<)

F. Identity Operators ( is , is not )

G. Membership Operators ( in , not in )
```

variable Features

variable assignments

- single variable assignment
- multiple variable assignment

Dynamic input reading

Conditional Statements

```
- if
- if else
- if elif else
- Nested if
```

In [3]:
```
# How to check your Python version
import sys
sys.version
# shift+enter
# alt+enter
# cntrl+enter
```

Out[3]:
```
'3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]'
```

Comments in Python

In [5]:
```
# comments
# 132+244
132+3445
```

Out[5]:
```
3577
```

In [ ]:
```
# multi line comments cntrl+/

# sdhfhgddf
# dvfhdfgh
# dsfhdfh
# jdgjhfag
# jdaghfh
```

Data Types:

- int,float,char,double,bool,long, str, --c
- int,float,bool
- Iterables - str,list,set,tuple,dict

In [6]:
```
v = 100 # variable - it holds any type of data
type(v)
```

Out[6]:
```
int
```

In [7]:
```
d = 100.900
type(d)
```

Out[7]:
```
float
```

In [8]:
```
s = "string"  # strig is a collection char's
type(s)
```

Out[8]:
```
str
```

In [9]:
```
c = 's'
type(s)
```

Out[9]: str

In [14]:
```
b = False
type(b)
```

Out[14]: bool

In [11]:
```
l = ["lavanya",123,890.900] # list  - data structure
type(l)
```

Out[11]: list

In [12]:
```
s = set()  # sets
ss = {123,3455,566}
type(s)
```

Out[12]: set

In [13]:
```
type(ss)
```

Out[13]: set

In [15]:
```
d = dict() # dictionary - combination of both key, value pair
type(d)
```

Out[15]: dict

In [16]:
```
d = {"name":"lavanya",
     "role":"Technical skill Trainer",
     "Exp":2}
type(d)
```

Out[16]: dict

In [18]:
```
t = () # tuple
type(t)
```

Out[18]: tuple

In [19]:
```
t = (123,45,657,879,98)
type(t)
```

Out[19]: tuple

Variable

- it's a name
- It Holds any type of data

Type Casting

- Changing one data type into another data type

In [20]:
```
# string into int
s = "2021"
s+1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-07c52e59d0e8> in <module>
      1 # string into int
      2 s = "2021"
----> 3 s+1

TypeError: can only concatenate str (not "int") to str
```

In [21]:
```
s+"2"
```

Out[21]:
```
'20212'
```

In [22]:
```
int(s)+1 # str to int
```

Out[22]:
```
2022
```

In [23]:
```
# int to str
str(9000)
```

Out[23]:
```
'9000'
```

In [24]:
```
# float to int
int(345.565)
```

Out[24]:
```
345
```

In [25]:
```
# int to float
float(8000)
```

Out[25]:
```
8000.0
```

In [26]:
```
# str to int
int("hi") # h having some ASCII
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-26-2e82c4a8a041> in <module>
      1 # str to int
----> 2 int("hi")

ValueError: invalid literal for int() with base 10: 'hi'
```

In [27]:
```
int("0")
```

Out[27]: 0

OPERATORS

- Performs a particular operation on operands
- relation between 1 or 2 operands
- 6 different types of operatos: A. Arithmetic Operators (+,-,*,%,/,//,**)

  B. Assignment Operators (=, +=, -=, *=, %=, /=, //=)

  C. Comparison operators (==, >=, <=, !=, >, <)

  D. Logical operators (and, or, not)

  E. Bitwise Operators (&, |, ~, ^, >>, <<)

  F. Identity Operators (is , is not)

  G. Membership Operators (in , not in)

A. Arithmetic Operators (+,-,*,%,/,//,**)

In [33]:
```
234+345
a = 234
b = 355
a*b
```

Out[33]: 83070

In [29]:
```
345-345
```

Out[29]: 0

In [30]:
```
345*56
```

Out[30]: 19320

In [31]:
```
23/2
```

Out[31]: 11.5

In [32]:
```
23//2
```

Out[32]: 11

In [34]:
```
23%2
```

Out[34]: 1

B. Assignment Operators (=, +=, -=, *=, %=, /=, //=)

In [37]:
```
a = 100
```

In [41]:
```
a += 100
a
```

Out[41]: 500

In [43]:
```
a = a+100
a
```

Out[43]: 700

In [44]:
```
a -= 100
a
```

Out[44]: 600

In [47]:
```
2333/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-47-ee14f1f463c2> in <module>
----> 1 2333/0

ZeroDivisionError: division by zero
```

C. Comparison operators (==, >=, <=, !=, >, <)

- True or False

In [49]:
```
a  == 600
```

Out[49]: True

In [51]:
```
a = 1000
b = 500
print(a == b)
print(a >= b)
print(a <= b)
print(a != b)
print(a > b)
print(a < b)
```

```
False
True
False
True
True
False
```

D. Logical operators (and, or, not)

In [52]:
```
True and False
```

Out[52]: False

In [53]:
```
True and True  # True Statement
```

Out[53]: True

In [54]:
```
True or False
```

Out[54]: True

In [55]:
```
False or False  # False
```

Out[55]: False

In [56]:
```
not True
```

Out[56]: False

In [57]:
```
not False
```

Out[57]: True

In [58]:
```
3445 and 45
```

Out[58]: 45

In [59]:
```
34 and 10
```

Out[59]: 10

In [60]:
```
54 and 4354647657765
```

Out[60]: 4354647657765

In [61]:
```
234 or 23
```

Out[61]: 234

In [62]:
```
23 and 0
```

Out[62]: 0

In [63]:
```
0 or 34
```

Out[63]: 34

In [64]:
```
0 or 234
```

Out[64]: 234

In [65]:
```
234 or 0  # True and False -- True
```

Out[65]: 234

In [66]:
```
0 and 345
```

Out[66]: 0

In [67]:
```
3445 and 0 # True and False -- false
```

Out[67]: 0

In [68]:
```
not(-1)
```

Out[68]: False

In [69]:
```
not(0)
```

Out[69]: True

In [72]:
```
not True
```

Out[72]: False

E. Bitwise Operators (&, |, ~, ^, >>, <<)

In [71]:
```
10 & 2
```

Out[71]: 2

10 -- 1010 2 -- 0010 & -- 0010 -- 2

In [73]:
```
1234 & 3
```

Out[73]: 2

In [74]:
```
133 & 0
```

Out[74]: 0

In [75]:
```
12 | 2
```

Out[75]: 14

12 -- 1100 2 --- 0010 |---- 1110 - 14

In [76]:
```
~ 1
```

Out[76]: -2

In [77]:
```
~ -2
```

Out[77]: 1

In [83]:
```
# -(n+1)

~ -1
-(-1+1)
```

Out[83]: 0

In [84]:
```
~ 2
-(2+1)
```

Out[84]: -3

In [80]:
```
~ 3
```

Out[80]: -4

In [85]:
```
~ 4
-(4+1)
```

Out[85]: -5

In [86]:
```
~ -4
-(-4+1)
```

Out[86]: 3

In [94]:
```
# >> right shift -- div by 2 with given number of times
# << left shift  -- Multiply by 2 with given number of times

print(12 >> 2)
12//2
6 // 2
```

```
3
```

Out[94]: 3

In [88]:
```
12 >> 1
```

Out[88]: 6

In [98]:
```
print(12 >> 3)
12 // 2
6 // 2
3 //2
```

```
1
```

Out[98]: 1

In [106]:
```
print(12 << 2 )
12 * 2
24 * 2
```

```
48
```

Out[106]: 48

In [100]:
```
12 << 1
```

Out[100]: 24

In [101]:
```
12 << 3
```

Out[101]: 96

In [102]:
```
12 << 4
```

Out[102]: 192

In [107]:
```
12 ^ 2
```

Out[107]: 14

12 -- 1100 2 -- 0010 1110 - 14

F. Identity Operators (is , is not)

In [108]:
```
True is True  # ==
```

Out[108]:
```
True
```

In [109]:
```
123 is not 123 # !=
```

```
<>:1: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:1: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<ipython-input-109-adfa2fcc4105>:1: SyntaxWarning: "is not" with a literal. D
id you mean "!="?
  123 is not 123
```

Out[109]:
```
False
```

In [110]:
```
# == it's comapring the values only
# is - it's comparing both value and address location
```

G. Membership Operators (in , not in)

In [111]:
```
"sanna" in "Prasanna"
```

Out[111]:
```
True
```

In [113]:
```
478 in [23,34,56,67,78,845,34,32,23, 478]
```

Out[113]:
```
True
```

In [120]:
```
"23" not in [23,34,54,6]
```

Out[120]:
```
True
```

In [121]:
```
"s" in "apssdc"
```

Out[121]:
```
True
```

variable Features

- do's
  - char
  - string
  - string+number
  - _
- don't
  - number
  - Spl Char's

In [127]:
```
v = 100
# 100 = "V"
st10 = 100
_ = 100
# $ = 90
```

```
  File "<ipython-input-127-9073cbe0e916>", line 5
    $ = 90
    ^
SyntaxError: invalid syntax
```

In [ ]:

In [ ]:

variable assignments

- single variable assignment
- multiple variable assignment

In [129]:
```
v = 10000  # single variable Assignment
v1,v2,v3 = 100
print(v1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-129-d3797d0c457e> in <module>
      1 v = 10000  # single variable Assignment
----> 2 v1,v2,v3 = 100
      3 print(v1)

TypeError: cannot unpack non-iterable int object
```

In [130]:
```
v1,v2,v3 = 100,200,300 # multi variable assignment
v3
```

Out[130]: 300

In [131]:
```
v1 = 100,200,300
v1
```

Out[131]: (100, 200, 300)

In [132]:
```
type(v1)
```

Out[132]: tuple

Dynamic Reading

In [136]:
```
age = input("enter your age")
print("my age is ", age)
```

```
enter your age24
my age is  24
```

In [138]:
```
age = int(input("enter your age"))
print("my age is ", age)
```

```
enter your age345
my age is  345
```

In [139]:
```
age = float(input("enter your age"))
print("my age is ", age)
```

```
enter your age34
my age is  34.0
```

In [143]:
```
ph = input("enter your phone number")
print("your ph number",ph)
```

```
enter your phone number8765453219
your ph number 8765453219
```

In [ ]:
```

```