1. A literal

is a constant value assigned to a variable.

Types:

1. Numeric Literals
2. String Literals

1.1 **Numeric Literals** may contain:

- Digits (0-9). Eg. 267, 56
- +or - signs. Eg. +267 , -56
- Letter e for exponential notation. Eg. 4.23e-4 ⇒ 4.23 X 10-4
- Decimal point. Eg. 2.67
- Comma is not allowed.

```
2.34e-3
```

```
0.00234
```

```
# Comma is not allowed. With comma literal is not taken as a single numeber.
2,300
```

```
(2, 300)
```

For **Integer Literals**, there is no limit for the value.

For **Floating Point** value, Python uses double-precision standard format (IEEE 754). It provides a range of **10 power -308** to **10 power 308** with 16 to 17 digits of precision.

Eg. 1.00342e+6 ⇒1.00342X10+6 , 6 digits of precision

```
# output will have 16 digits after decimal and hence is said to be 16 digits
# precision

1/3
```

```
0.3333333333333333
```

```
# Arithmetic Overflow: It occurs when result of arithmetic operation goes be
# upper limit.
# Eg. 2.0e200 * 3.0e300 ⇒ 6.0e500 (expected)  ⇒ inf (in Python)

2.0e200 * 3.0e300
```

```
inf
```

```
# Arithmetic Underflow: It occurs when result of arithmetic operation goes
# beyond lower limit.
# Eg. 2.0e-200 / 1.0e300 ⇒ 2.0e-500 (expected)  ⇒ 0.0 (in Python)

2.0e-200 / 1.0e300
```

```
0.0
```

```
# 1/3  ⇒ .3333333333333333 (16 Digit precision)

5/3
```

```
1.6666666666666667
```

1.2 String Literals

It represents a sequence of characters delimited (surrounded) by a matching pair of either single or double quotes (and sometimes triple quotes). Eg. “Hello” , ‘Hello’

```
#Displays double quotes
print('Hell"o')
```

```
Hell"o
```

```
#Displays single quotes
print("Hell'o")
```

```
Hell'o
```

```
# Displays both single and double quotes
print('''Hell"o'n''')
```

```
Hell"o'n
```

Empty String: is a string with only a pair of matching quotes with nothing in between. It is different from the string containing one blank character.

```
# variable a is empty string
a = ''
len(a)  #len function determines the length of the string variable
```

```
0
```

```
# variable b consists of single space
b=' '
len(b)  #len function determines the length of the string variable
```

```
1
```

Characters are encoded within a compute using coding scheme. Unicode is the universal encoding scheme utilizing 8 or 32 bits for each character. By default, Python uses **UTF-8** which is compatible with **ASCII encoding scheme**.

ord function: gives the UTF-8 encoding of a given character.

Eg. ord('A') ⇒ 65

chr function: gives the character for a given encoding scheme.

Eg. chr(65) ⇒ 'A'

```
# Prints ASCII code of character '4'
ord('4')
```

```
52
```

```
# Prints character corresponding to ASCII code passed to chr function
chr(67)
```

```
'C'
```

2. Control Character

are used to control the display of output. They themselves are not displayed on the screen. Eg. Escape sequence.

Escape sequence: is a string that contains a backslash() followed by one escape sequence characters. **Eg. \n**

```
# Without \n
print("abcde")
```

```
abcde
```

```
# With \n
print("abc\nde")
```

```
abc
de
```

3.Variable

is a name (Identifier) associated with a value.

Value of a Variable can be changed during a program execution.

Mathematically, num = num +1 does not make sense, however in programming it increments the value of num by 1.

In python, statement k = num will not allocate new memory space to variable k, both num and k will refer to same memory location.

However, if in the next statement we execute k=20, then new memory will be allocated to variable k.

```
# Since both variables num and k store the same value , so separate memory
# is not assigned to two variables.
# Function id determines the momery location referenced by variable num
# and k.
# It can be noted that both have same momory location and hence proved that
# separate memory is not allocated to two variables
num = 10
k=num
print(id(num))
print(id(k))
```

```
11256352
11256352
```

```
# Since variable num is assigned with value 10 and variable k is assigned
# with value 20, so separate memory is assigned to two variables.
# Using id function, it can be noted that both have different momory
# location & hence proved that separate memory is allocated to two variables
k=20
print(id(num))
print(id(k))
```

```
11256352
11256672
```

✓ 59s completed at 8:43 PM ● ✕