Constant, variable and Type:

A **constant** has a value which cannot be changed.
A **variable** has a value which can be changed.
A **type** is a mechanism used for classification.

We classify the automobiles as cars.
We classify the cars as sedans, SUV, hatchbacks and MUV.
We classify the subjects as physics, chemistry, zoology and so on.
It becomes easier to comprehend based on classification.

We know the car and the calculator are not of the same type.
They have different behaviours. We can navigate a car; we can evaluate expression on a calculator.

Characteristics depend on the type.
The type is mathematically a set.
It specifies the **values and operations**. **It does not specify how the operation is carried out.**
We can slowdown a car by applying brakes. Whether the brakes are hydraulic or mechanical or is some other mechanism could vary from car to car. But all cars do support braking!

Run this program and find out about various types Python supports.

```
# file: 1_variable_type.py
# discussed:
#       constant (literal)
#       variable
#       type
#               has set of values
#               has set of operators
#               specified what happens when the operators are applied
#               does not specify how operators are applied
#       any value has a fixed type
```

```
#      type of a variable depends on the value assigned to it

a = 10
print(a, type(a))  # int
a = "fool"
print(a, type(a)) # str
a = 3.14
print(a, type(a)) # float
a = True
print(a, type(a))  # values of bool type : False True
a = 2 + 3j
print(a, type(a)) # complex
#---------------------
a = (11, 22, 33, 44)
print(a, type(a)) # tuple

a = [11, 22, 33, 44]
print(a, type(a)) # list

a = {11, 22, 33, 44}
print(a, type(a)) # set

a = {11 : 22, 33 : 44}
print(a, type(a)) # dict
```

We classify the types into basically two categories.

1. simple or scalar types:

have a single value.

Types are : int float bool complex

2. structured or reference types:

have more than one value. There are like a hostel with # of rooms.

Types are : list tuple set dict

We will discuss more of these in the next lectures.

Variable and id:

In Python, variables having the same value may share the location. we can examine by checking the id. Check the program 2_variable_id.py

Test 1:

In the first part, we find that the id of a, b, c, d[0] are all same. They all refer to a single value 100.

What happens if one of them is changed say c to 1000.

Only c changes and others still remain at 100.

Test 2:

We have a list x with two elements. This is called a structured type as it has # of elements.

When we assign x to y, both will have the same id.

When we change x[0], we are changing not x, but a component of x, y will also change.

Test 3:

We have a list x with two elements.

When we assign x to y, both will have the same id.

When we reassign to x, we are changing x. So y will not be changed.

This is a confusing topic. We will revisit this topic throughout the course.

Rules:

1. Changing a scalar variable will not affect others which have the same value.
2. **Changing a structured variable** will not affect others which have the same value.
3. **Changing part of a structured variable** will affect all others which have the same value.

```
# file: 2_variable_id.py

# Test 1
# int : has only one element; scalar or simple or primitive type
a = 100
b = a
c = 50 + 50
d = [100, 200]
print(id(a))
print(id(b))
print(id(c))
print(id(d[0]))
# all have the same id

c = 1000
print("a : ", a) # has not changed
# only c changed, a, b, d[0] did not change.

# Test 2
# list : has # of elements ; structured or reference type

x = [11, 22]
y = x
print(x, id(x))
print(y, id(y))

x[0] = 33;  # y also changed
print(x, id(x))
print(y, id(y))

# Test 3:
x = [11, 22]
y = x
print(x, id(x))
```

```
print(y, id(y))

x = [33, 44]; # y does not change
print(x, id(x))
print(y, id(y))
```