```
## Till Now
```

- variables
- basic data types
  - string
  - integer
  - float
  - bool
- data structures - inbuilt in Python
  - list
  - tuples (covered in Intermediate)
  - dictionaries (covered in Intermediate)
  - sets (covered in Intermediate)

```
x = 3
print(type(x))
```

```
<class 'int'>
```

```
y = "hello"
print(type(y))
```

```
<class 'str'>
```

```
z = True
print(type(z))
```

```
<class 'bool'>
```

```
a = 0.55
print(type(a))
```

```
<class 'float'>
```

```
l = [1, 5, -1, 2]
print(type(l))
```

```
<class 'list'>
```

```
l.append(10)
```

```
print(l)
```

```
[1, 5, -1, 2, 10]
```

```
print(len(l))

    5
```

```
# Everything we defined above, is an object in python like int, str, float
```

Banking Application

- Customers
- Bank Accounts
- Transactions

Object Oriented Programming (OOPs)

- map real world objects in our code

What are the advantages?

- Better readability
- Better maintenance
- Modular: easy to reuse, replace, debug, etc.
- More flexible

School

- Students

```
### 2 Things: Class, Object
```

```
class Student:
  name = 'Sai' # data
  roll_number = 321 # data
```

```
s1 = Student() # an object of Student class

s2 = Student() # an object of Student class

print(type(s1))
print(type(s2))

    <class '__main__.Student'>
```

```
<class '__main__.Student'>
```

```
## How to access this data?

print(s1.name)
```

```
Sai
```

```
print(s2.name)
```

```
Sai
```

```
print(s1.roll_number)
print(s2.roll_number)
```

```
321
321
```

```
print(s1)
```

```
<__main__.Student object at 0x7effd324de90>
```

```
print(s2)
```

```
<__main__.Student object at 0x7effd322f1d0>
```

```
s1.name = 'Riyon'

s2.name = 'Sahil'

print(s1.name, s1.roll_number)
print(s2.name, s2.roll_number)
```

```
Riyon 321
Sahil 321
```

Requirement: To store the name of the student while creating the object

```
class Student:

  def __init__(self, name):
    self.name = name
```

```
s1 = Student("Mahesh") # able to initialize the name, create object with name

s2 = Student("Rajat")

print(s1.name)
print(s2.name)
```

```
Mahesh
Rajat
```

```
class Student:

  # this special function: dunder init (double underscore init double underscore)
  # --> called as constructor

  # self is nothing but the object on which you are working
  def __init__(myownself, name):
    print('Printing self inside the constructor:')
    print(myownself)
    myownself.name = name


s1 = Student("Mahesh") # able to initialize the name, create object with name
print()

print('Printing s1 outside:')
print(s1)
print(s1.name)
```

```
Printing self inside the constructor:
<__main__.Student object at 0x7effd316b750>

Printing s1 outside:
<__main__.Student object at 0x7effd316b750>
Mahesh
```

```
class Student:

  # this special function: dunder init (double underscore init double underscore)
  # --> called as constructor
  # automatically called when you create an object

  # self is nothing but the object on which you are working
  def __init__(self, input_name):
    # self WILL ALWAYS BE the first argument
    print('Printing self inside the constructor:')
    print(self)
    # assigning the data
    self.name = input_name


s1 = Student("x") # able to initialize the name, create object with name
print()

print('Printing s1 outside:')
print(s1)

print()
print(s1.name)
```

```
Printing self inside the constructor:
<__main__.Student object at 0x7effd31b7090>

Printing s1 outside:
<__main__.Student object at 0x7effd31b7090>

x
```

```
class Student:

  def __init__(self, input_name, input_roll_number = -1):
    self.name = input_name
    self.roll_number = input_roll_number


s1 = Student('Sahil', 8)

s2 = Student('Sanjana', 4)
```

```
print(s1.name, s1.roll_number)
```

```
Sahil 8
```

```
print(s2.name, s2.roll_number)
```

```
Sanjana 4
```

```
s1.roll_number = 17
print(s1.name, s1.roll_number)
```

```
Sahil 17
```

```
s1 = Student('Sahil')
print(s1.name, s1.roll_number)
```

```
Sahil -1
```

Quiz

```
class Vehicle:
    def __init__(self, name):
        self.name = name

v = Vehicle()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-53-64cd109d6101> in <module>()
      3         self.name = name
      4
----> 5 v = Vehicle()

TypeError: __init__() missing 1 required positional argument: 'name'
```

SEARCH STACK OVERFLOW

```
class Vehicle:
    def __init__(self, name):
        self.name = name

v = Vehicle.create('minivan')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-54-3d3497382660> in <module>()
      3         self.name = name
      4
----> 5 v = Vehicle.create('minivan')

AttributeError: type object 'Vehicle' has no attribute 'create'
```

SEARCH STACK OVERFLOW

```
class Vehicle:
    def __init__(self, name):
        self.name = name

v = Vehicle('minivan')
print(v.name)
```

```
minivan
```

Some behavior

```
class Student:

  # default constructor if you don't provide it
  def __init__(self):
    pass # it's a blank function - does nothing
```

```
s = Student()
```

```
# Introduction behavior / method

class Student:

  def __init__(self, input_name):
    self.name = input_name

  # who is the student giving the introduction
  def intro(self):
    print('Hey Everyone! I am', self.name)
```

```
s = Student('Sahil')
s.intro()
```

```
    Hey Everyone! I am Sahil
```

```
l = list([1, 2, 3])
l.append(5)

print(l)
```

```
    [1, 2, 3, 5]
```

```
# Class = Data + Methods
```

```
# Introduction behavior / method

class Student:

  def __init__(self, input_name):
    self.name = input_name

  # who is the student giving the introduction
  def intro(self):
    print('Hey Everyone! I am', self.name)
```

```
s = Student('Sahil')
s.intro()
```

```
s2 = Student('Prakhar')
s2.roll_number = 231
s2.marks = 100

print(s.name)
print(s2.roll_number)

print(type(s))
print(type(s2))
```

```
Hey Everyone! I am Sahil
Sahil
231
<class '__main__.Student'>
<class '__main__.Student'>
```

```
class Test:
  def __init__():
    pass


t = Test()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-69-c77c31b3bb1b> in <module>()
      4
      5
----> 6 t = Test()

TypeError: __init__() takes 0 positional arguments but 1 was given
```

SEARCH STACK OVERFLOW

✓ 0s completed at 23:19 ● ×