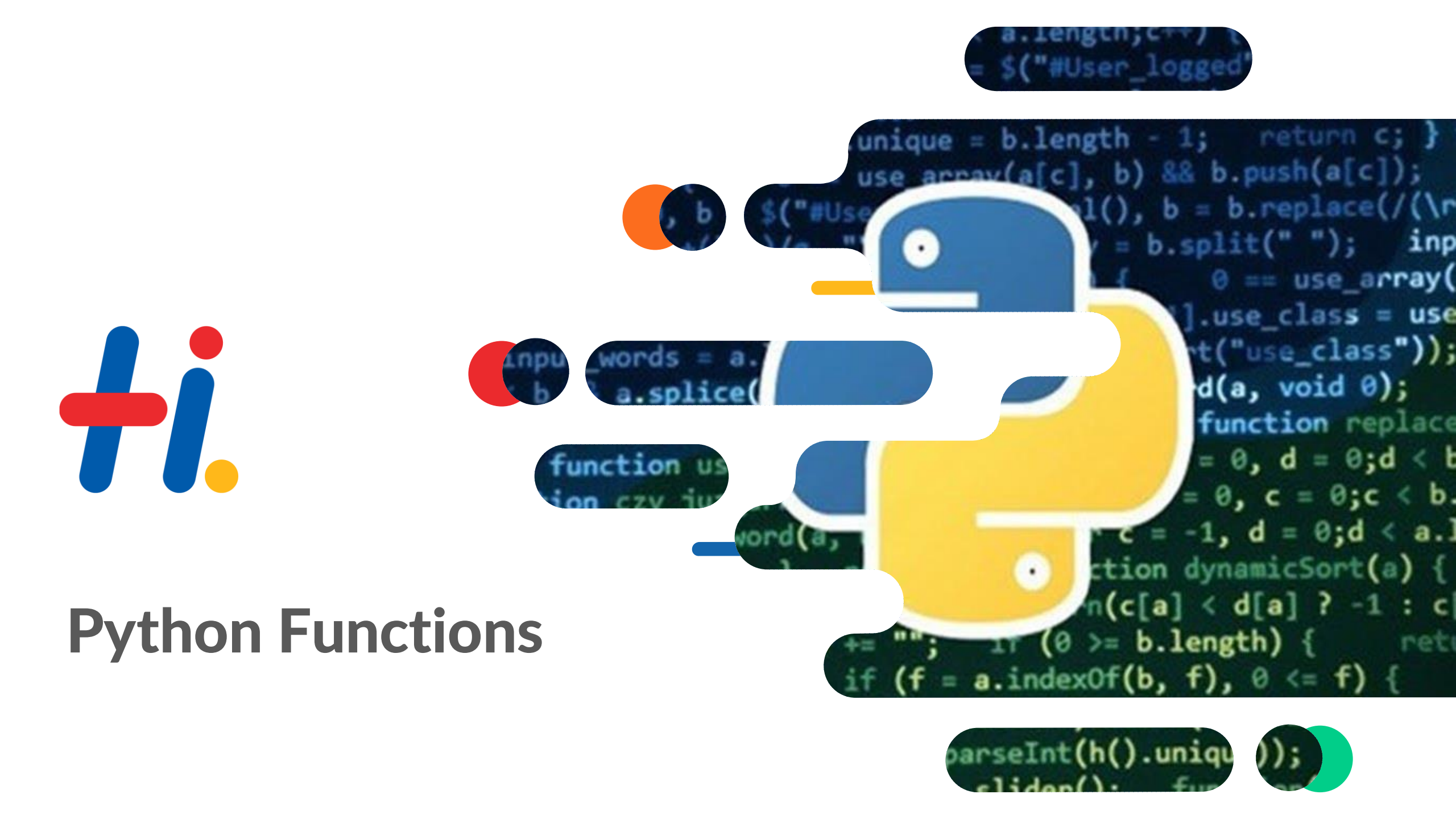
HEXAVARSITY
Enhancing Excellence

Python Programming

Functions, Modules & OOPS

Session Objective

- Python Functions
- Python Modules
- OOPS
  - Class and Object
    - Constructor
    - Access Specifiers
  - Inheritance
  - Polymorphism
  - Class and Static Methods
  - Variable Types & Scope



Python Functions

Python Functions

Functions

- A function is a group of related statements that performs a specific task.
- Functions help to break the program into smaller and modular chunks.
- As the program grows larger, functions make it more organized and manageable.

```
def function_name(parameters):
    """docstring"""
    statement(s)
```

Python Functions

Contd...

Function Definition:

- ✓ Keyword **def** that marks the start of the function header.
- ✓ A function name to uniquely identify the function.
- ✓ A **colon (:)** to mark the end of the function header.
- ✓ One or more valid python statements that make up the function body.
- ✓ Statements must have the same indentation level of 4 spaces.

Optional in Function :

- ✓ The return statement to return a value from the function.
- ✓ Parameters through which values are passed to a function.
- ✓ Documentation string (docstring) to describe what the function does.

Python Functions

Contd…

Example for Function

```
def greet(name):
    """This function greets to the person passed in as a parameter """
    print("Hello, " + name + ". Welcome to Python Programming!")
```

Note:

- ✓ The string after the function header is called the docstring. It explains about the function.
- ✓ Documentation is a good programming practice.

Python Functions Contd…

Function Call

Once the function is defined, it can be called from another function, program or even from the Python prompt.

```
>>> greet('Tharun')

Output;
Hello, Tharun. Welcome to Python Programming!
```

Python Functions

Contd...

Function with Argument

Keyword argument:

- Keyword arguments are related to the function calls.
- The caller identifies the arguments by the parameter name.
- This allows to skip the arguments or place them in order.
- Python interpreter can use the keywords provided to match the values with parameters.

Python Functions

Contd...

Keyword argument Example

```
def employeeDetails(eid, ename ):
  # "This prints a passed info into this function"
   print("Employee id: ", eid)
   print ("Employee name ", ename)
   return;

# Call function
employeeDetails(eid=31410,ename='Vimala')
```

Python Functions

Contd...

Variable-Length Arguments in Python with *args and **kwargs

*args	**kwargs
Arguments that can take an unspecified amount of input	Python can accept multiple keyword arguments, better known as **kwargs.
It stores data in list format	It stores data in dictonary format
Function definition: **def** fun_args(*args): print(args)	**Function definition:** **def** fun_args(**kargs): print(kargs)
Function call: fun_args(22,33,44,55)	**Function call:** fun_args(eid=31410,ename="**vimala**")
Output: (22, 33, 44, 55)	**Output:** {'eid': 21, 'ename': 'vimala'}

Python Modules

Python Modules

- Modules refer to a file containing Python statements and definitions.
- A file containing Python code, for example: Calculation.py, is called a module
- Modules are used to break down large programs into small manageable and organized files.
- Modules provide reusability of code.
- The module can be imported, instead of copying their definitions into different programs.

Python Modules

Contd…

Module Creation:

STEP 1:

Create a file Calculation.py

STEP 2:

Write the following code to calculate the average of 2 numbers

```
def avgof2Numbers(num1,num2):
    """This program calculate the average of
    two numbers and return the result"""
    sum = num1+num2
    result = sum/2
    return result
```

Python Modules

Contd…

STEP 3:

Import modules in Python

>>>import Calculation

STEP 4:

Access the function using the dot . Operator

>>>print("The average of 2 numbers is:", Calculate.avgof2Numbers(5,5))

STEP 5:

Output

The average of 2 numbers is: 5

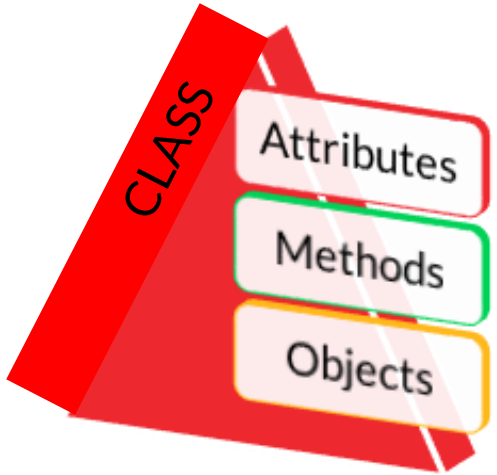
Note: A new Python file can be created to access the function from the module

OOPS - Class & Object

Class and Object

CLASS

- Python is an object-oriented programming language
- A class is a user-defined blueprint or prototype from which objects are created
- An object is simply a collection of data (variables) and methods (functions) that act on those data.
- Classes provide bundling of data and functionality together.



Class and Object

Contd...

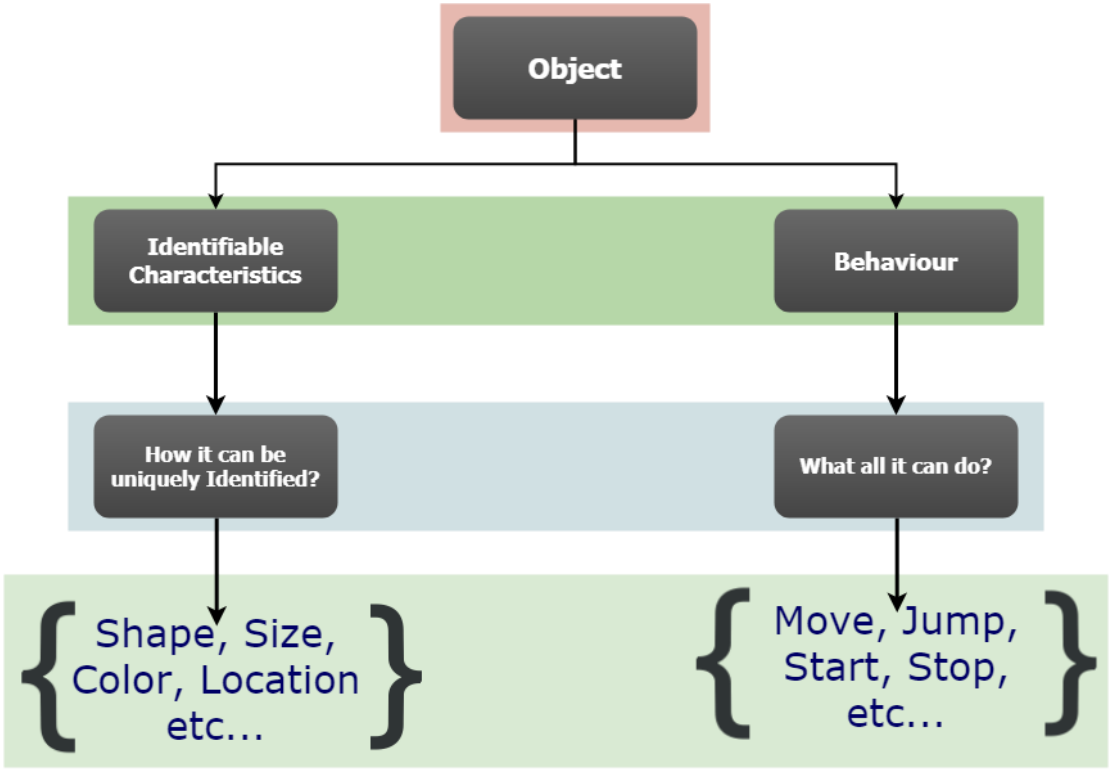
OBJECT

An object consists of ,

•State : It is represented by attributes of an object.

•Behavior : It is represented by methods of an object.

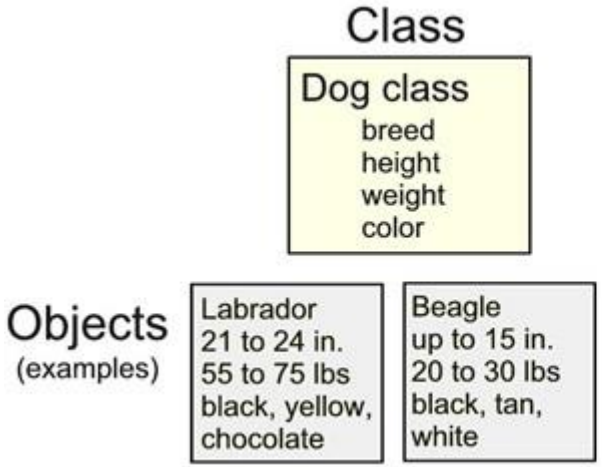
•Identity : It gives a unique name to an object and enables one object to interact with other objects.

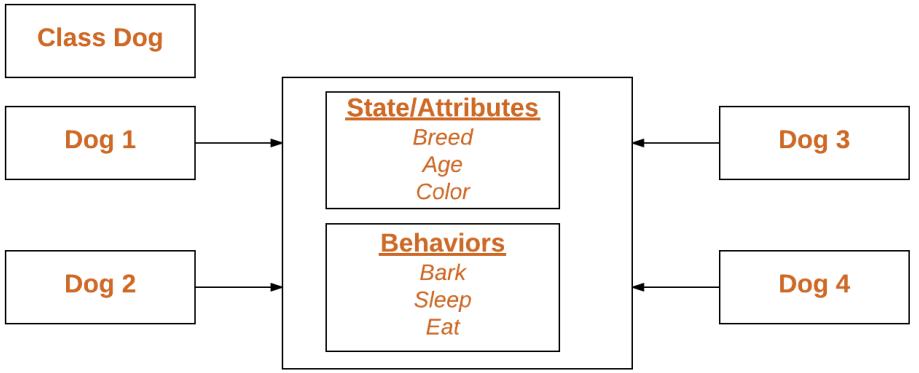


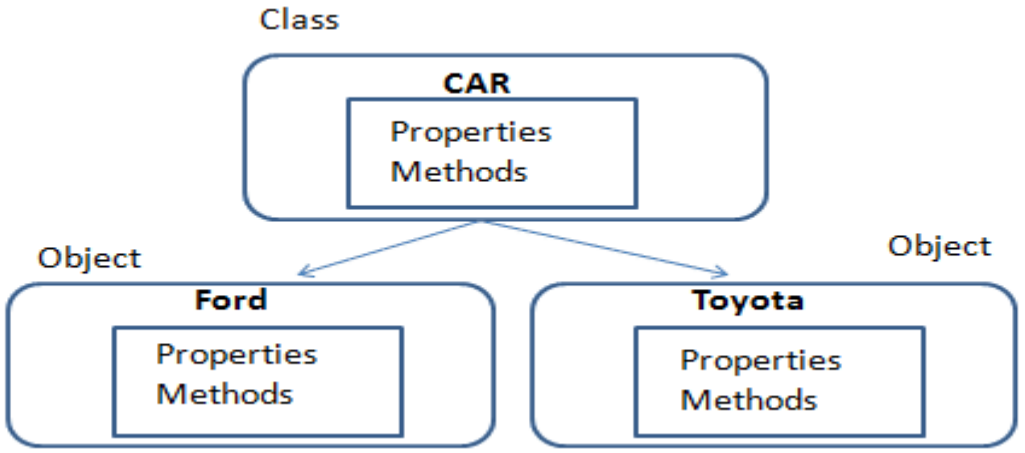
Class and Object

Contd…

Class and Object Example





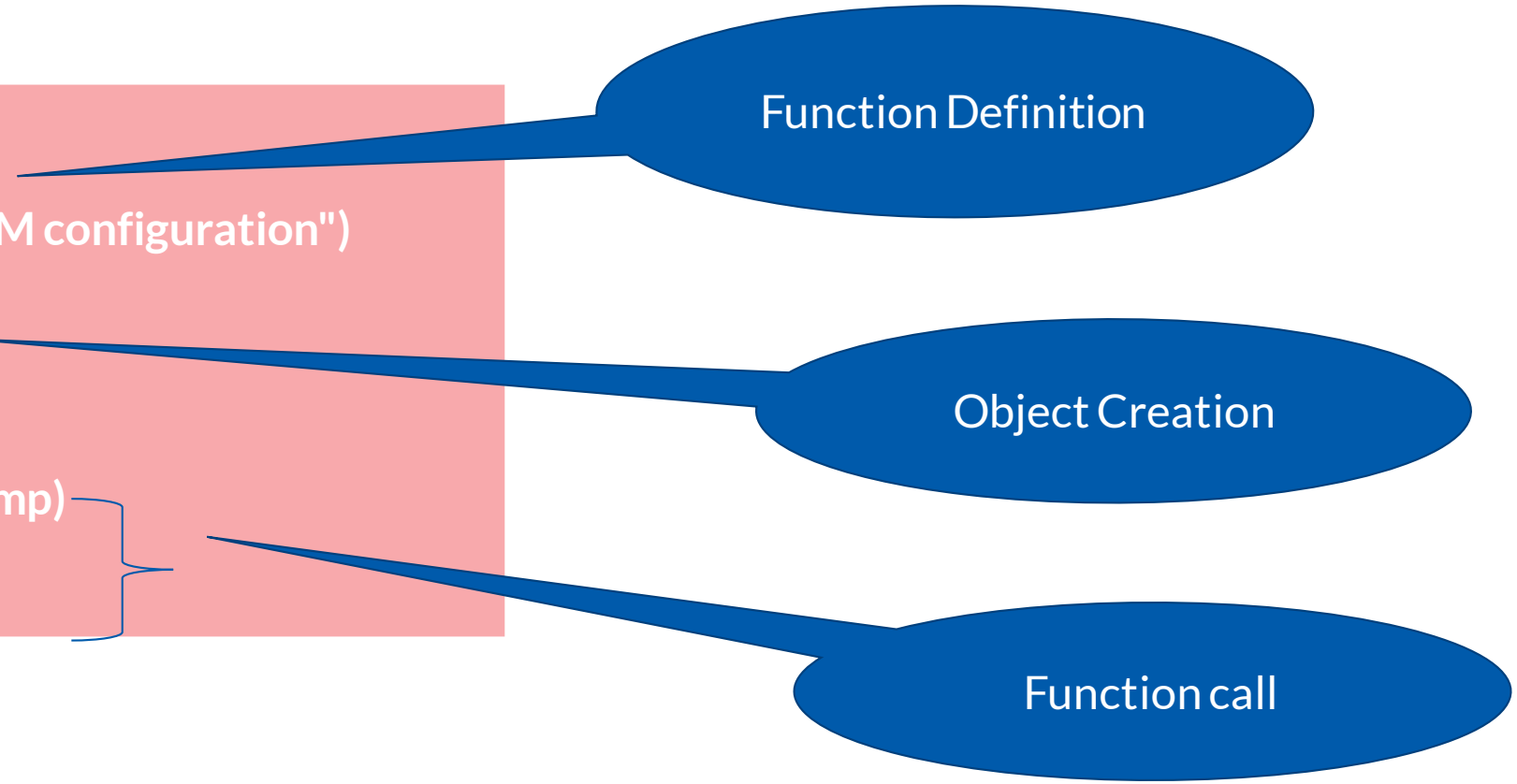


Class and Object

Contd…

Class Definition

```
class computer:
   def config(self):
      print("16 GB RAM configuration")

comp=computer()


computer.config(comp)

comp.config()
```



Class and Object

Contd…

Self

- Self is used to represent the instance of the class
- With this keyword attributes and methods of the class in python can be accessed
- It binds the attributes with the given arguments
- When a method is called as myobject.method(arg1, arg2), this is automatically converted by Python into MyClass.method(myobject, arg1, arg2)

Class and Object

Contd…

Class and Object Example

```
class computer:
   def __init__(self,ram,hardisk):
      self.ram=ram
      self.hardisk=hardisk

   def config(self):
      print("Computer configuration")
      print(self.ram, self.hardisk)

comp=computer("16GB","80GB")
comp1=computer("8GB","30GB")

comp.config()
comp1.config()
```


Constructor in Python

Constructors in Python

Class functions that begin with **double underscore** __ are called special functions as they have special meaning.

__init__ method:

- It is known as a **constructor** in object-oriented concepts.
- This method called when an object is created from the class
- It allow the class to **initialize** the attributes of a class.
- Constructor contains instructions that are executed at the time of Object creation.

Syntax of constructor declaration :

```
def __init__(self):
    # body of the constructor
```

Constructor in Python Contd…

Types of constructors :

Default Constructor :

- ✓ The default constructor is simple constructor which doesn't accept any argument.
- ✓ It's definition has only one argument which is a reference to the instance being constructed.

Parameterized Constructor :

- ✓ Constructor with parameters is known as parameterized constructor.
- ✓ The parameterized constructor take its first argument as a reference to the instance being constructed known as self and the rest of the arguments are provided by the programmer.

```
# parameterized constructor
  def __init__(self, f, s):
    self.first = f
    self.second = s
```

Garbage Collection

Destroying Objects (Garbage Collection)

- Python deletes unused objects (built-in types or class instances) automatically to free the memory space
- The process by which Python periodically reclaims blocks of memory that no longer are in use is termed Garbage Collection
- Python's garbage collector runs during program execution
- It is triggered when an object's reference count reaches zero
- An object's reference count changes as the number changes

Access Specifier in Python

Python - public, private and protected

Private:

- ✓ Private members of a class have denied access from outside the class.
- ✓ They can be handled only from within the class.

Public:

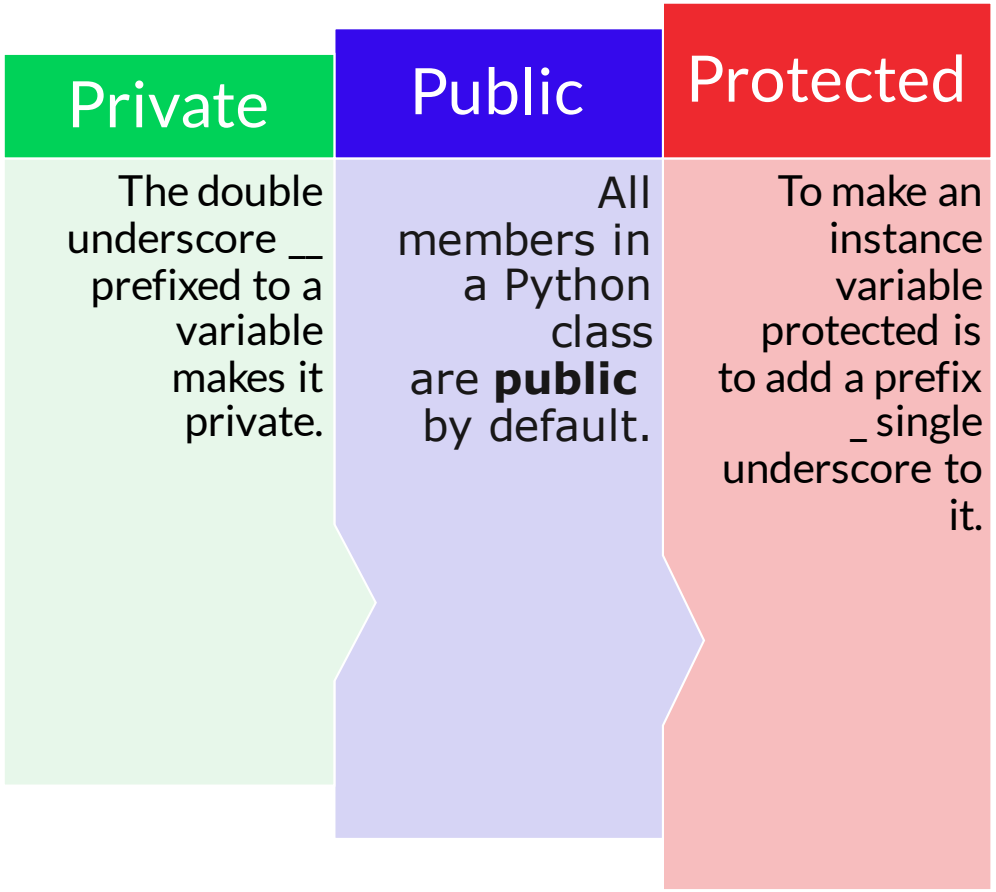
- ✓ Public members are accessible from outside the class.
- ✓ The object of the same class is required to invoke a public method.

Protected:

- ✓ Protected members of a class are accessible from within the class and are also available to its sub-classes.
- ✓ This enables specific resources of the parent class to be inherited by the child class.

Access Specifier in Python

Contd…



Access Specifier in Python Contd…

Public

```
class employee:
def __init__(self, name, sal):
self.name=name
self.salary=sal
```

Protected

```
class employee:
def __init__(self, name, sal):
self._name=name # protected attribute
self._salary=sal # protected attribute
```

Private

```
class employee:
def __init__(self, name, sal):
self.__name=name # private attribute
self.__salary=sal # private attribute
```

OOPS - Inheritance

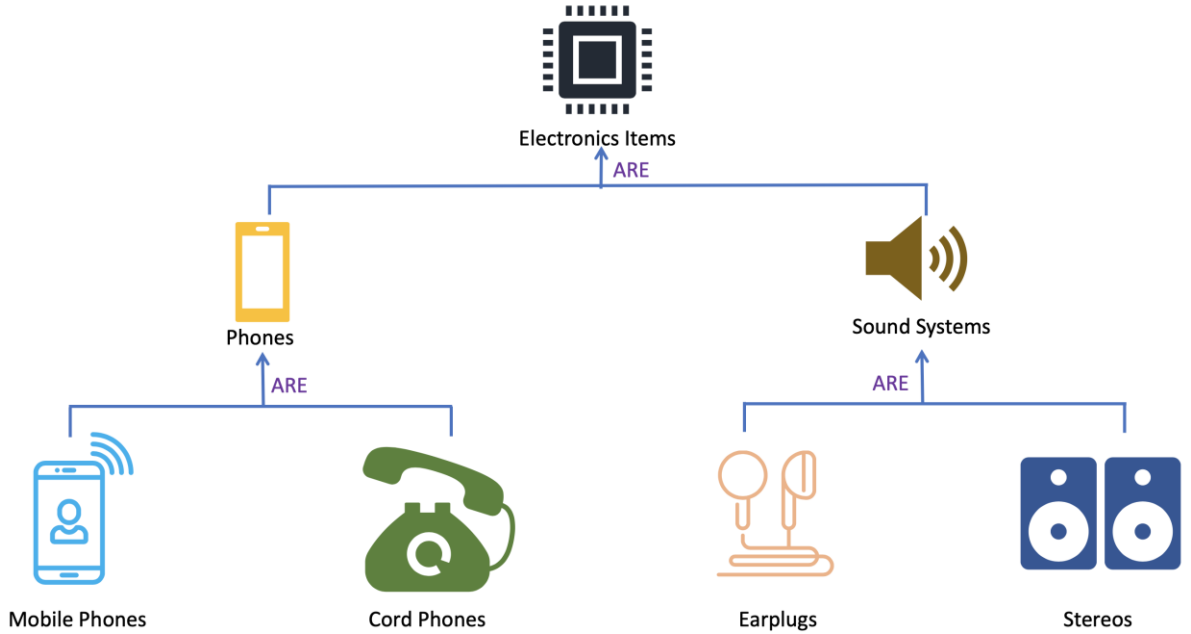
Inheritance

Inheritance

- In inheritance an object is based on another object.
- Inheritance is the capability of one class to derive or inherit the properties from another class.
- The methods and attributes that were defined in the base class will also be present in the inherited class.

The benefits of inheritance are:
1.It represents real-world relationships well.
2.It provides reusability of a code.

```
class DerivedClassName(BaseClassName):
    pass
```



Inheritance

Contd…

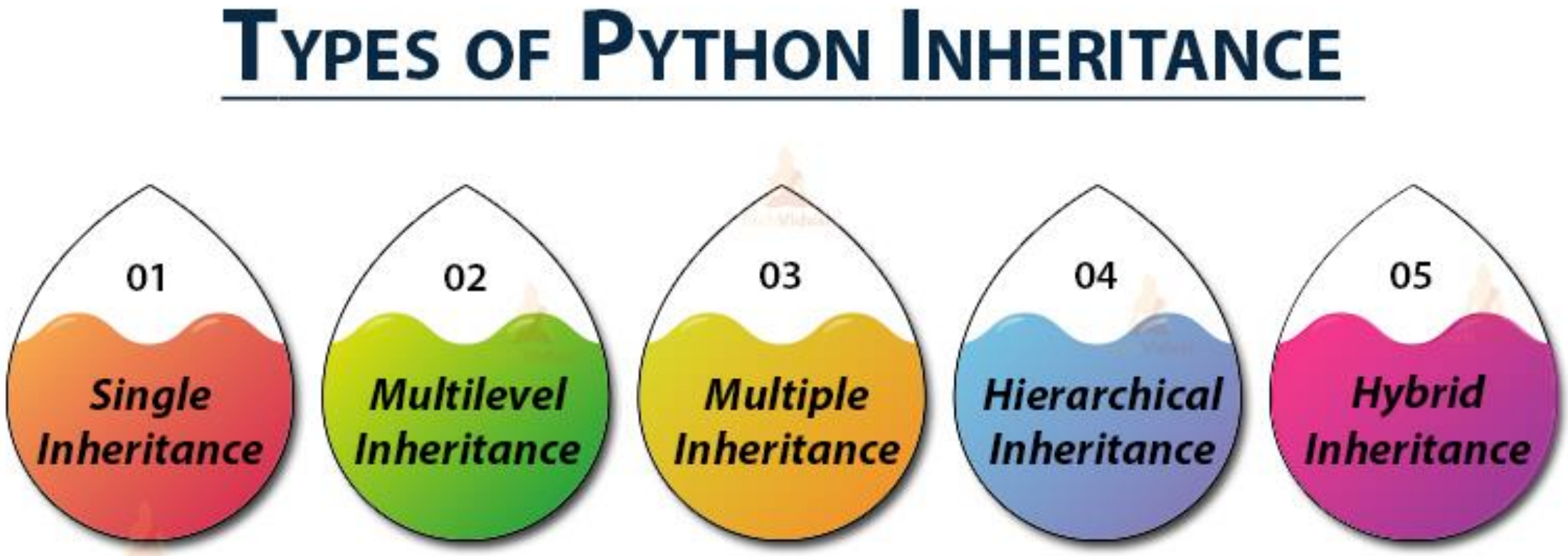
Inheritance Example

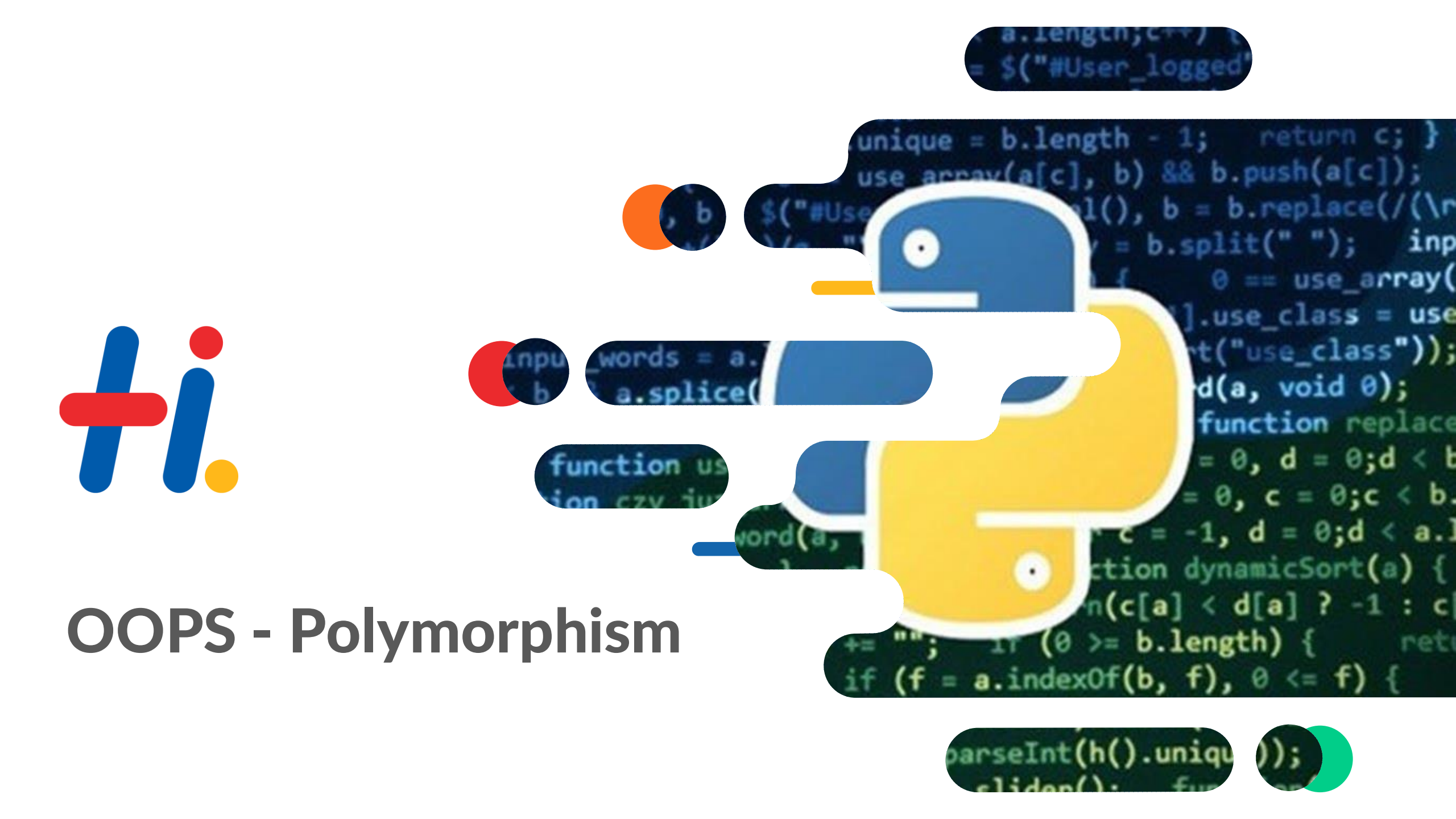
```
class Hexa_Project:

   def Project1(self):
      print("BFS")
   def Project2(self):
      print("ATM")

class Hexa_Admin(Hexa_Project):
   def Project3(self):
      print("Hexavarsity")
   def Project4(self):
      print("HR")
```

```
hex=Hexa_Project()
hex1=Hexa_Admin()

hex.Project1()
hex.Project2()
hex1.Project3()
hex1.Project4()
```

Inheritance

Contd…





OOPS - Polymorphism

Polymorphism

Polymorphism

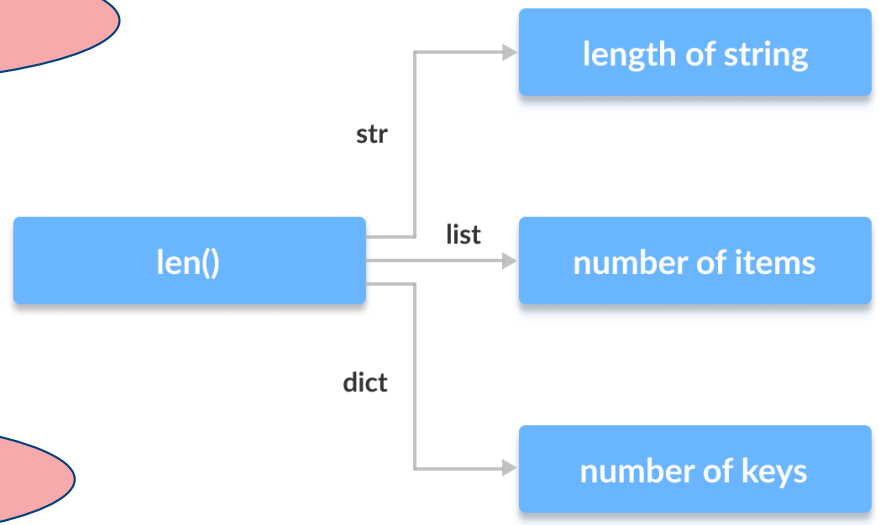
The word polymorphism means having many forms. In programming, polymorphism means same function name but different signatures being uses for different types.

Example:
Python program to demonstrate in-built poly-morphic functions

```
# len()
print(len("geeks"))

# len()
print(len([10, 20, 30]))
```

used for a string

used for a list



Polymorphism

Contd…

Polymorphism with “+” operator

- The operator “+” is used extensively in Python programs.
- For integer data types, + operator is used to perform arithmetic addition operation.

```
num1 = 1
num2 = 2
print(num1+num2)
```

- For string data types, + operator is used to perform concatenation.

```
str1 = "Python"
str2 = "Programming"
print(str1+" "+str2)
```

Polymorphism Contd…

Class Polymorphism in Python

- Python allows different classes to have methods with the same name.
- These methods can be called with their respective objects of the class.
- Python doesn't support method overloading based on different number of parameters in functions.

```
class PythonTraining:
  def __init__(self,trgid,trgname):
    self.trgid=trgid
    self.trgname=trgname
  def trgDetails(self):
    print("Python Training id:  ",self.trgid,"  training name:",self.trgname)
```

Polymorphism

Contd…

```
class JavaTraining:
   def __init__(self,trgid,trgname):
      self.trgid=trgid
      self.trgname=trgname
   def trgDetails(self):
      print("Java Training id:
",self.trgid,"  training
name:",self.trgname)
```

```
python=PythonTraining(22,"Python
game")
java=JavaTraining(11,"Spring")
python.trgDetails()
java.trgDetails()
```

Output:

```
Python Training id:   22   training name: Python game
Java Training id:   11   training name: Spring
```

Polymorphism Contd…

Polymorphism and Inheritance

- The child classes in Python inherit methods and attributes from the parent class.
- Methods and attributes can be redefined specifically to fit the child class, is Method Overriding.
- The overridden methods and attributes can be accessed.

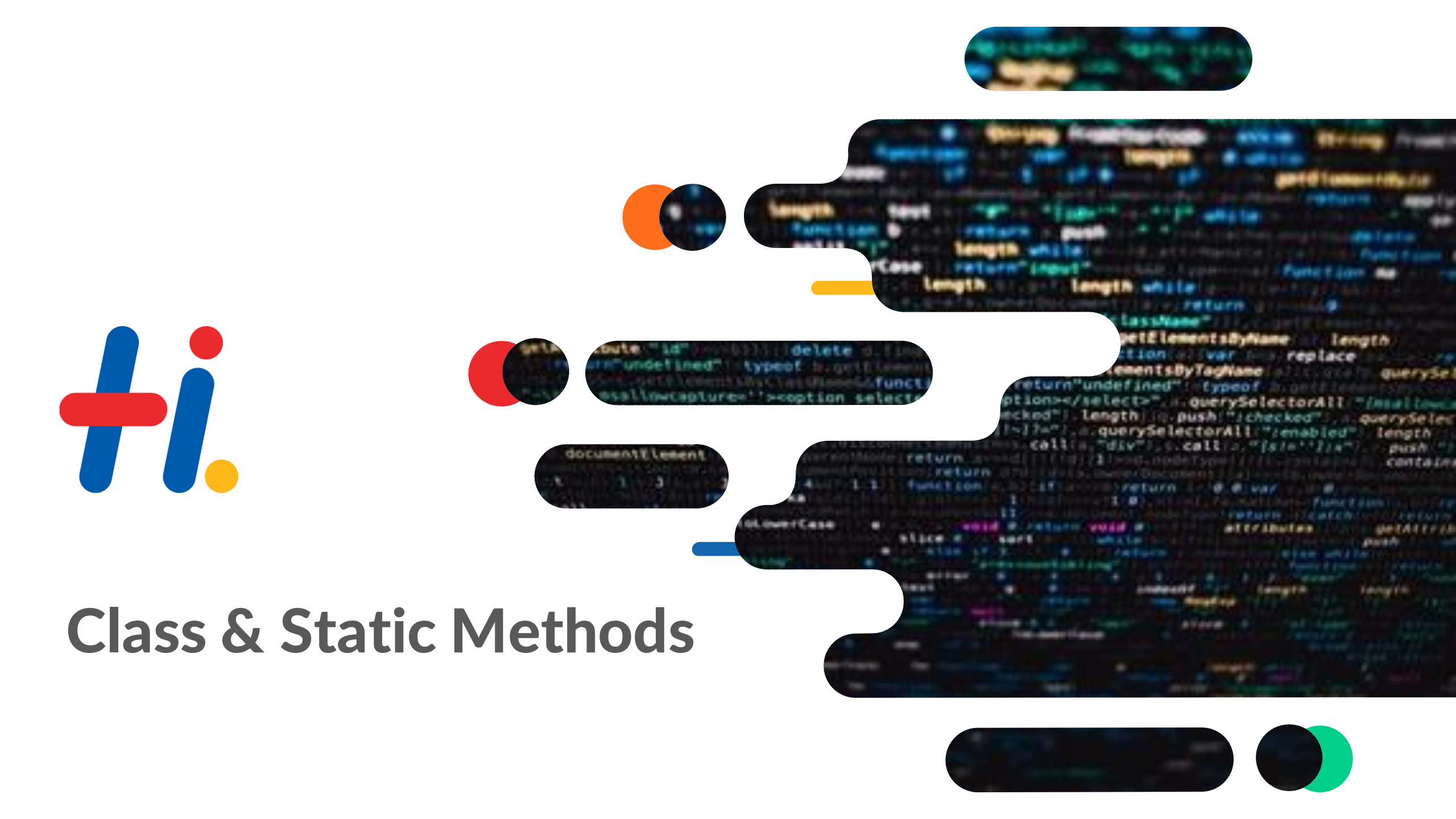
```
class Customer:
   def Details(self):
      print("Customer information.....")

class Order(Customer):
   def Details(self):
      print("Order details!!!!!!")

cust=Customer()
ord=Order()

ord.Details()
```


Polymorphism

Contd...

Example for Inheritance with Polymorphism

```
class Customer:
   def __init__(self,name):
      self.name=name
   def Details(self):
      print("Customer information.....")

class Order(Customer):
   def __init__(self,ordername):
      super().__init__("Vimala")
      self.ordername=ordername
   def Details(self):
      print("Order details!!!!!!")
      print(self.name,self.ordername)

ord=Order("Watch")
ord.Details()
```

Class & Static Methods

Class method vs Static method in Python

The @Classmethod decorator

- A class method is a method which is bound to the class and not the object of the class.
- They have the access to the state of the class
- It can modify a class state that would apply across all the instances of the class.
- For example it can modify a class variable that will be applicable to all the instances.

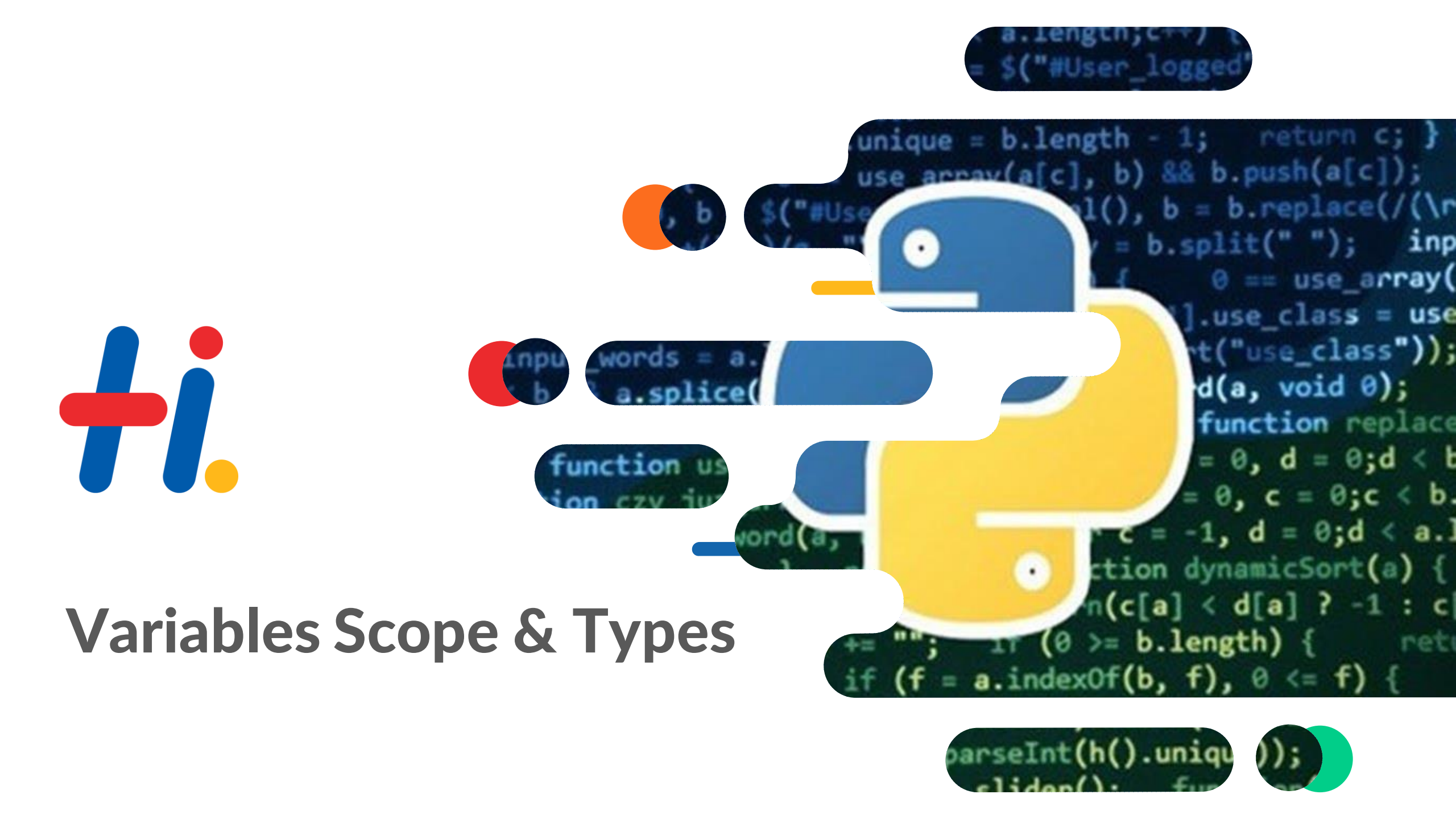
```
class C(object):
   @classmethod
   def fun(cls, arg1, arg2, ...):
      ....
fun: function that needs to be converted into a class method
returns: a class method for function.
```

Class method vs Static method in Python Contd…

The @Static Method

- A static method does not receive an implicit first argument.
- A static method is also a method which is bound to the class and not the object of the class.
- A static method can't access or modify class state.

```
class C(object):
   @staticmethod
   def fun(arg1, arg2, ...):
       ...
returns: a static method for function fun.
```



Variables Scope & Types

Variables Scope

Variables

- Variables are reserved memory locations to store values.
- Based on the data type of a variable, the interpreter reserve the memory space.
- Python variables do not need explicit declaration to reserve memory space.
- The declaration happens automatically when the value is assigned.
- The equal sign (=) is used to assign values to variables.

Variables Scope

Contd…

Variables Types

The part of a program where a variable is accessible is called its scope.

Local Scope

- ✓ Whenever a variable is defined within a function, its scope lies only within the function.
- ✓ It is accessible from the point at which it is defined until the end of the function and exists for as long as the function is executing the source.

```
def print_number():
   first_num = 1
   # Print statement 1
   print("The first number defined is: ", first_num)

print_number()
# Print statement 2
print("The first number defined is: ", first_num)
```

NameError: name 'first_num' is not defined

Variables Scope

Contd…

Global Scope

A variable defined outside any function becomes a global variable, and its scope is anywhere within the program.

```
greeting = "Hello"

def greeting_world():
    world = "World"
    print(greeting, world)

def greeting_name(name):
    print(greeting, name)

greeting_world()
greeting_name("Samuel")
```

Variables Type

Contd…

Class and Instance Variable

Class Variable	Instance Variable
Instance variables are for data unique to each instance	Class variables are for attributes and methods shared by all instances of the class.
Instance variables are variables whose value is assigned inside a constructor or method with self	Class variables are variables whose value is assigned in the class.

Variables Type

Contd…

Instance Variable	Class Variable

class Mobile:
 	
  def __init__(self):
  #instance variable
  self.name=**"Samsung"**
  self.color=**"blue"**
 	
m1=Mobile()
m2=Mobile()
 	
m1.name=**"iPhone"**
 	
print(m1.name,m1.color)
print(m2.name,m2.color)
	class Mobile:
	  #class variable
	  memory=**"60 GB"**
	  def __init__(self):
	  self.name=**"Samsung"**
	  self.color=**"blue"**
	
	m1=Mobile()
	m2=Mobile()
	
	m1.name=**"iPhone"**
	
	print(m1.name,m1.color,m1.memory)
	print(m2.name,m2.color,m2.memory)

Output: iPhone blue Samsung blue	Output: iPhone blue 60GB Samsung blue 60GB

Variables Type

Contd…

Class and Static Variable

- ✓ Class or static variables are shared by all objects.
- ✓ Instance or non-static variables are different for different objects
- ✓ Static keyword is not required to be mentioned.
- ✓ All variables which are assigned a value in class declaration are class variables.
- ✓ And variables which are assigned values inside methods are instance variables.

Variables Type

Contd…

Example for Class and Static Variable

```
class CSStudent:
  stream = 'cse'                # Class Variable
  def __init__(self,name,roll):
    self.name = name          # Instance Variable
    self.roll = roll          # Instance Variable


a = CSStudent('Geek', 1)
b = CSStudent('Nerd', 2)
```

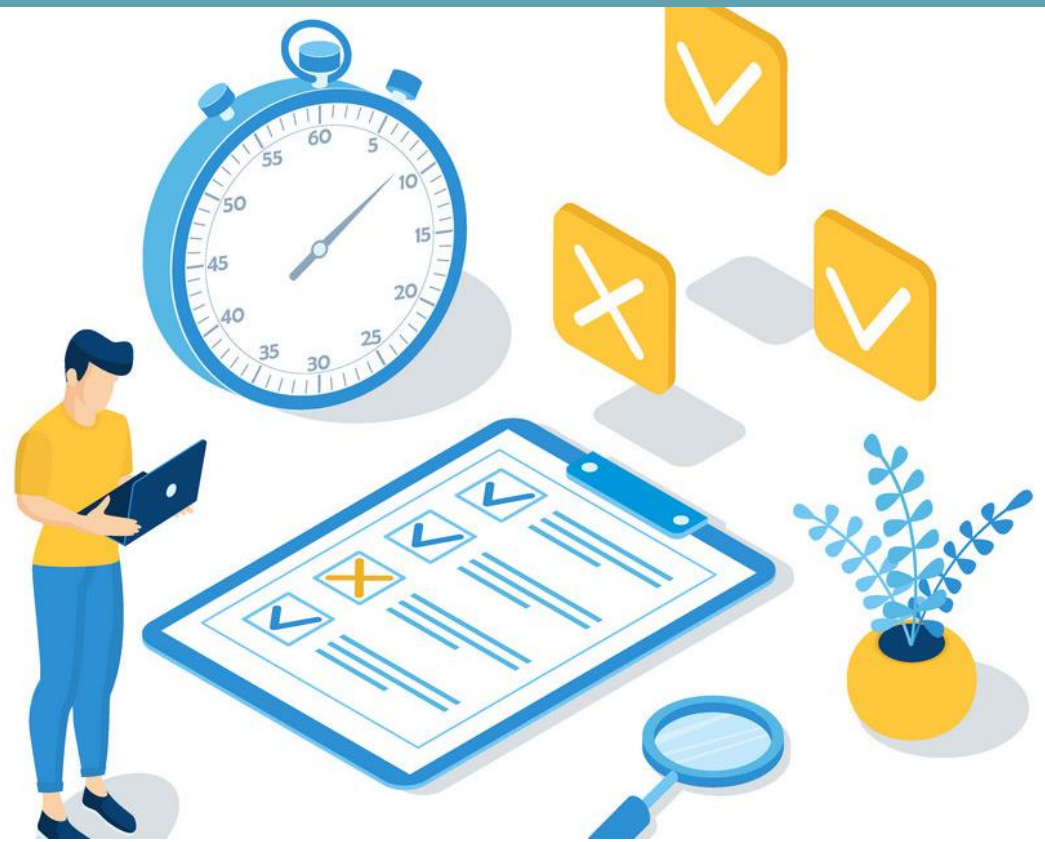
```
print(a.stream)  # prints "cse"
print(b.stream)  # prints "cse"
print(a.name)   # prints "Geek"
print(b.name)   # prints "Nerd"
print(a.roll)   # prints "1"
print(b.roll)   # prints "2"

# Class variables can be accessed using class name
print(CSStudent.stream) # prints "cse"
```

Think and Answer

1. Which keyword is used for function?
2. _____ is used to create an object.
3. The assignment of more than one function to a particular operator is _____
4. How to create a empty class in python?
5. What type of inheritance ?

```
class A():
   pass
class B():
   pass
class C(A,B):
   pass
```

Think and Answer



1. Def
2. Constructor
3. Operator Overloading
4. Class A:
   pass
5. Multiple Inheritance

Thank you

Innovative Services

Passionate Employees

Delighted Customers