Umamadhav Eedara
@umamadhav Eedara

## WHAT IS PYTHON?

- Python is a popular **programming language**.
- It was created by **Guido Van Rossum** in **1991**.

## PYTHON IS USED FOR?

- **Web Development**(server-side).
- **Database**.
- **Artificial intelligence**.
- **Machine Learning**.

**Umamadhav Eedara**
@umamadhav Eedara

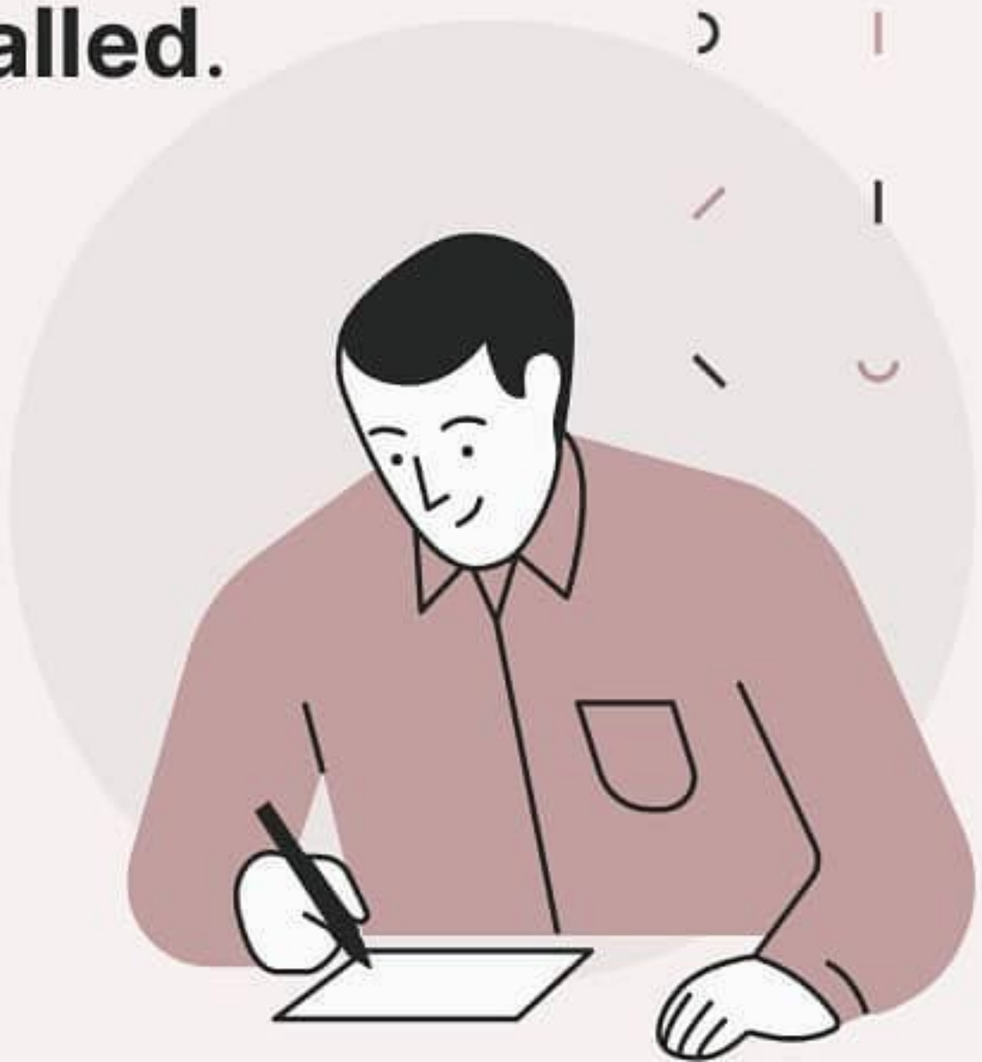
# WHY PYTHON?

- Python works on all platforms(Windows, Linux, and Mac).
- **Simple Syntax** similar to Engish.
- Python is an **Interpreter System**, Meaning that code can be executed as soon as it is written.

**Umamadhav Eedara**
@umamadhav Eedara

# HOW TO DOWNLOAD PYTHON?

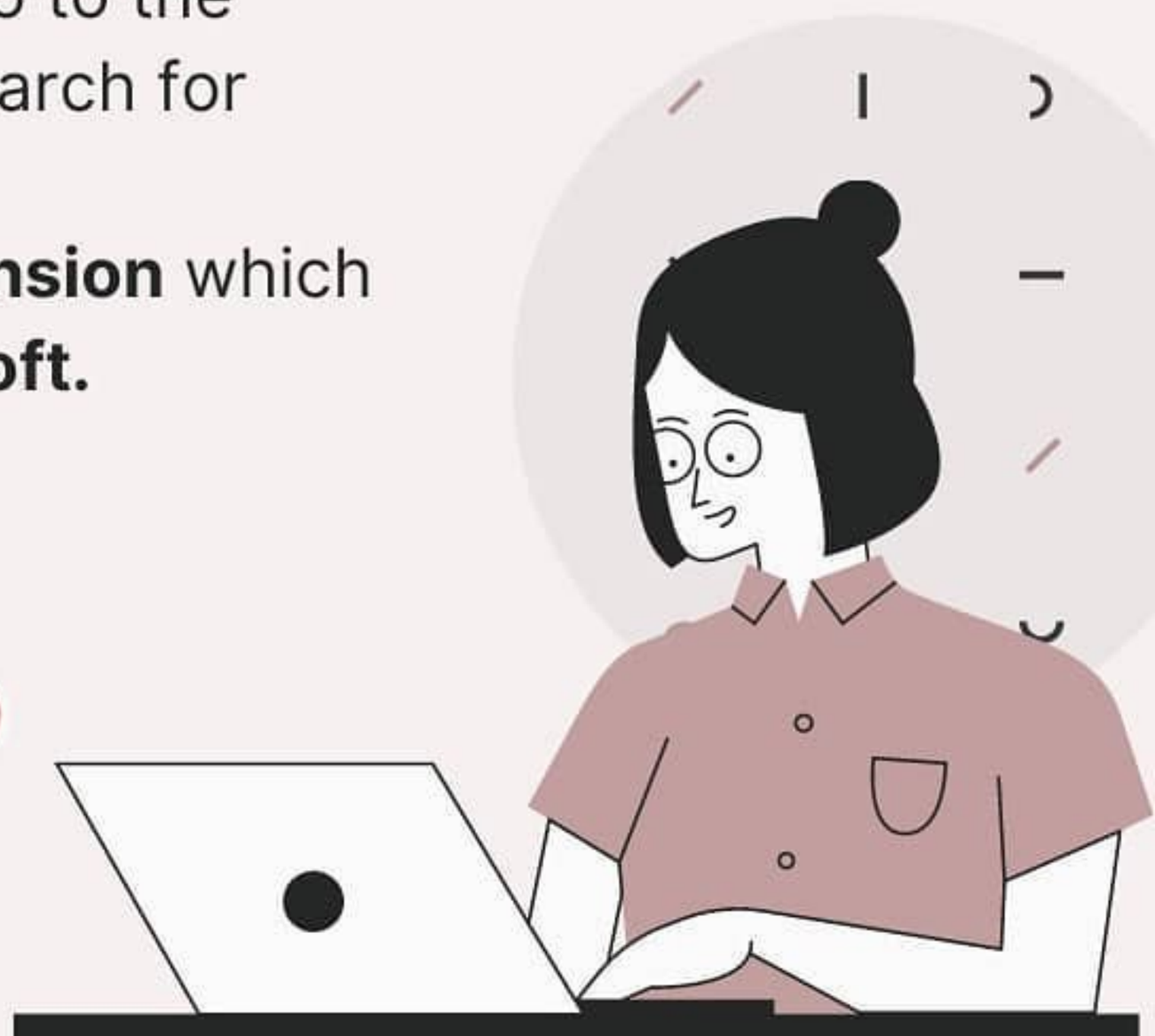
- Go to https://www.python.org/
- Click on the **Download** Button.
- Run the Downloaded File and **click on the checkout box Add python to python**.
- Now, click **Install**.
- Wait for few minutes and you get a message that **setup up installed**.
- Click Okay!



**Umamadhav Eedara**
@umamadhav Eedara

## INSTALLING VS CODE AND SETUP FOR PYTHON

- Go to https://code.visualstudio.com/
- Click on the Download Button.
- Run the Downloaded File and click agree on the check box.
- Now click **Install Button**.
- Wait for few minutes and you get a message that VS Code installed.
- Click Finish!
- Now, open vs code and go to the **Extension section** and search for **python**.
- Click on the **Python Extension** which was published by **Microsoft.**



**Drop A Love** ❤️

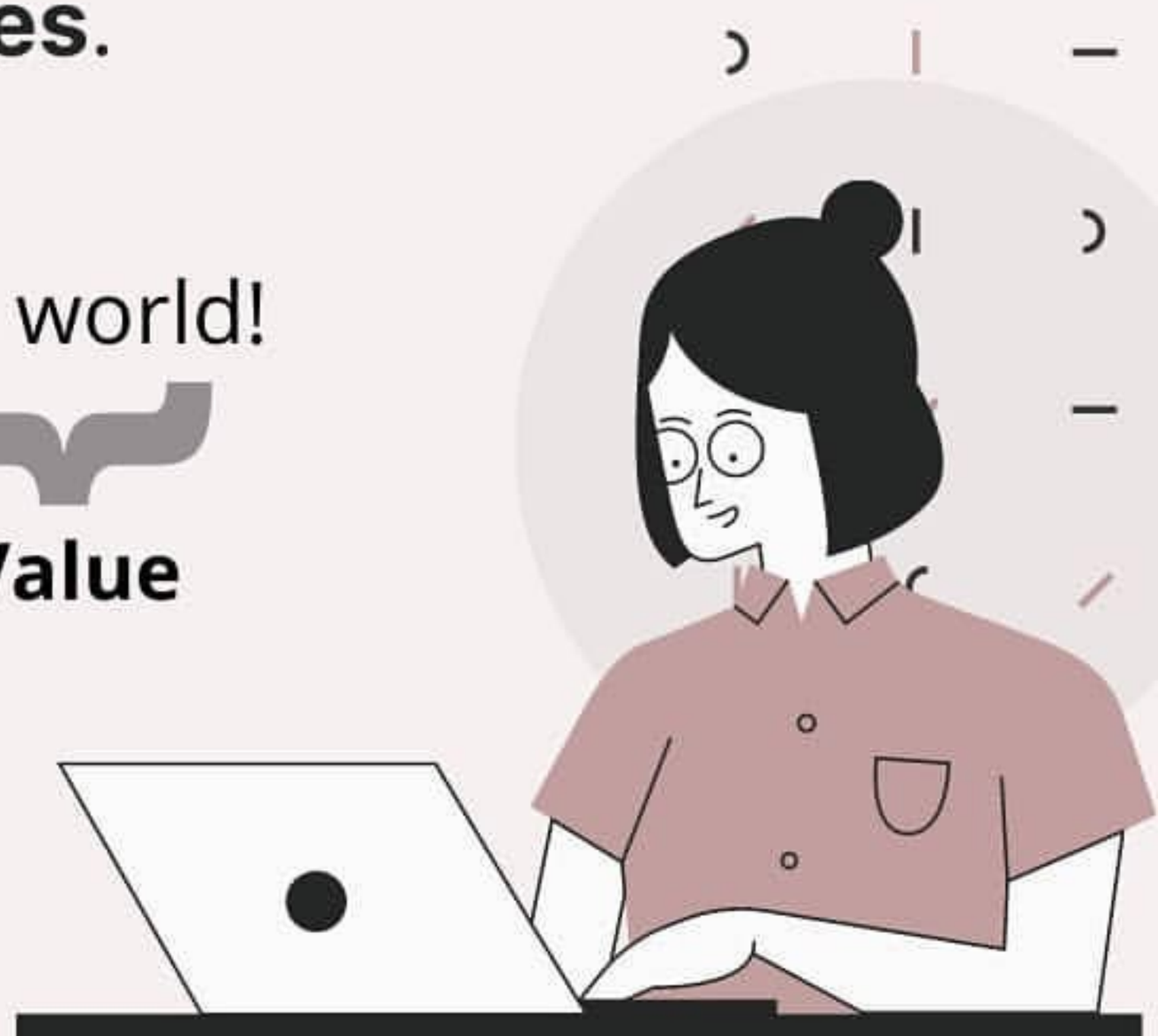
**Umamadhav Eedara**
@umamadhav Eedara

# VARIABLES IN PYTHON:

- Every **variable** is connected to a **value**.
- Variable are used to **store information**,and later we **can manipulate** it in computer program.
- Variables are like **containers** that are used to **store values**.

Message = Hello world!

**Variable** **Value**

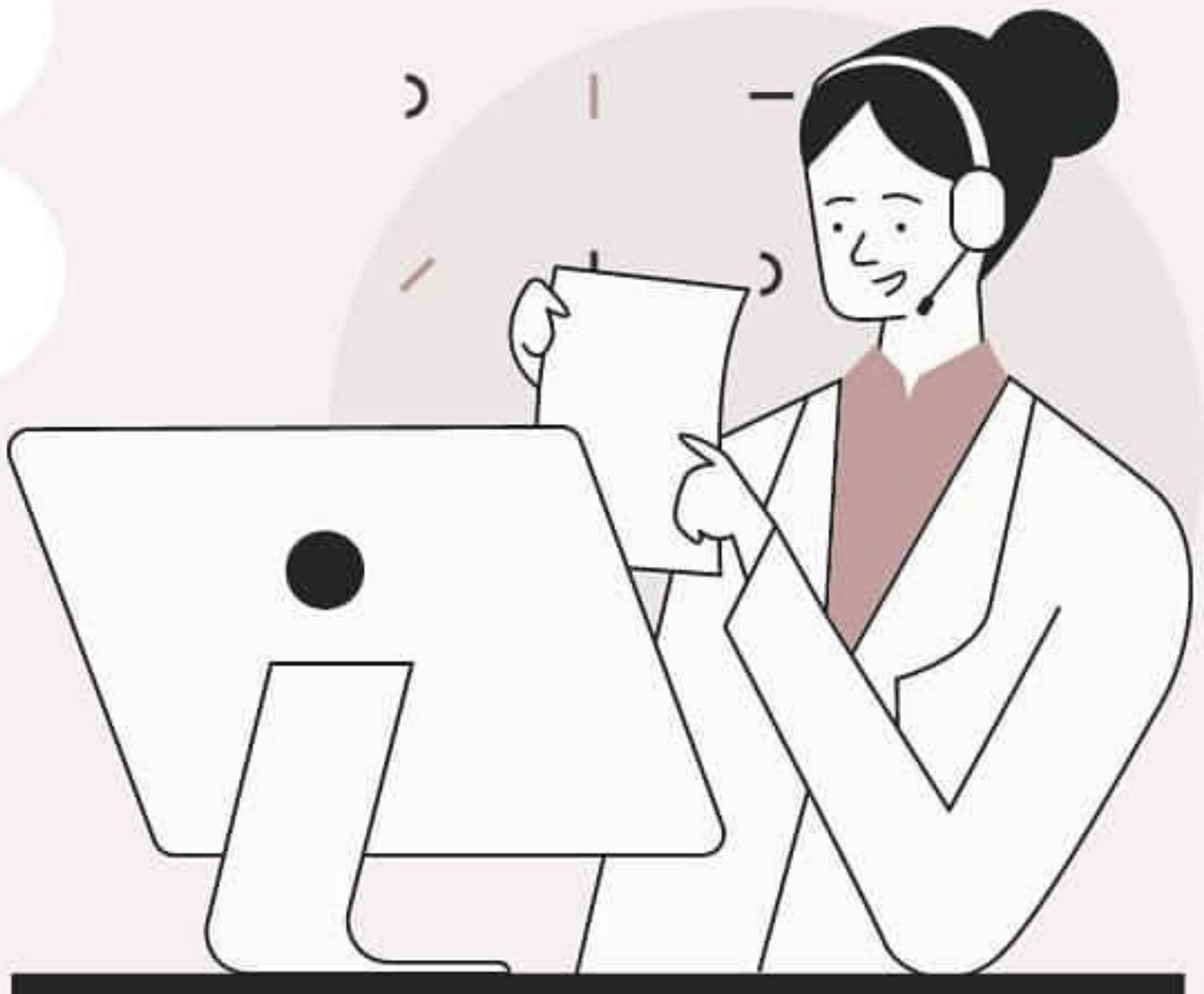


**Umamadhav Eedara**
@umamadhav Eedara

- Variable can create with **Name**,**Letter**,**Number**, and **Underscore**.
- But **not** with a **Number**.
- **Spaces** are **not allowed** in variable.
- But, **underscores** can be **used to** seperate the variable.

✔ **Message_1**

✘ **1_Message**



Umamadhav Eedara
@umamadhav Eedara

# SIMPLE DATA TYPES IN PYTHON:

- **Strings,**
- **Integers**, and
- **Float.**

Umamadhav Eedara
@umamadhav Eedara

# STRINGS:

- Anything inside the **Quotes** is considered as a **string**.
- We can use **Singe**, **Double**, and **Triple** Quotes.

Example:

'This is a string'

"This is also a string"

Umamadhav Eedara
@umamadhav Eedara

# INTEGERS:

- Integer, is a **whole number**, **positive** or **negative**, **without decimals**, of **unlimited length**.

Example:

x=1
y=56472317
z=-4567

} (Integers)

Umamadhav Eedara
@umamadhav Eedara

# FLOAT

- Float, is a **number, positive** or **negative**, containing **one** or **more decimals**.

Example:

x=1.0
y=564.72317
z=-4.567

} (Floats)

Drop A Love

**Umamadhav Eedara**
@umamadhav Eedara

# PYTHON LISTS:

- A list is a **collection** of **iteams** in a **particular order**.
- In a list we can include **Alphabets**, **Numbers**.
- In python **[] Square Brackes** indicates lists and **individual element** in the list is seperated by **commas**.

Example:

```
cars=["audi","benz","tata","bently"]
```

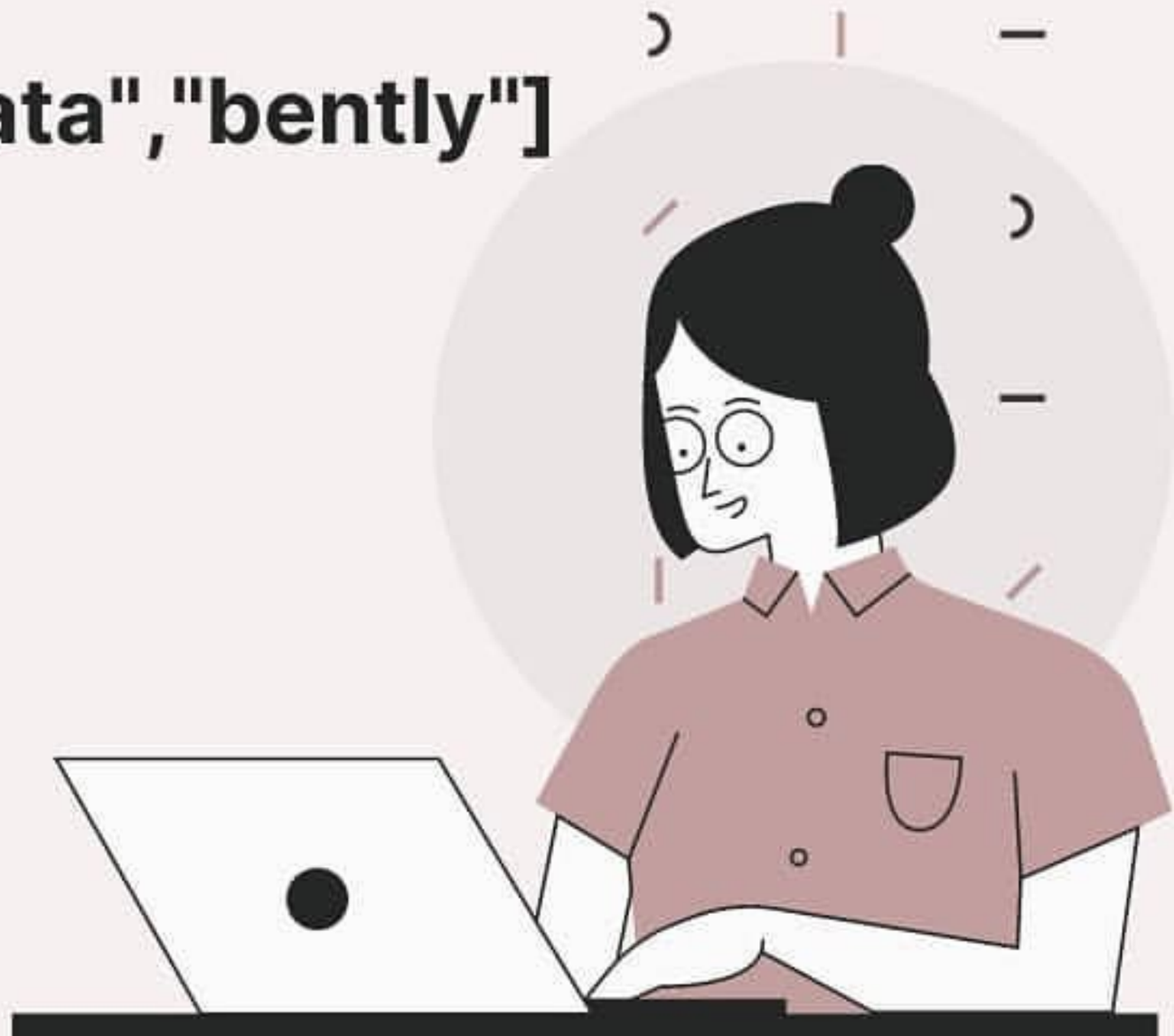
Umamadhav Eedara
@umamadhav Eedara

## SYNTAX:

```
>>>cars=["audi","benz","tata","bently"]
>>>print(cars)
```

## OUTPUT:

```
>>>["audi","benz","tata","bently"]
```



**Umamadhav Eedara**
@umamadhav Eedara

# INDEX POSITION IN LISTS:

- python consider **first iteam** in tht list to be position at '**0**' **not** at '**1**'.
- We can do **reversing indexing** but, **last element** is consider as '**-1' not '0**'.

**-4** **-3** **-2** **-1**

cars=["audi","benz","tata","bently"]

**0** **1** **2** **3**

Umamadhav Eedara
@umamadhav Eedara

## SYNTAX:

```
>>>cars=["audi","benz","tata","bently"]
>>>print(cars[0])
>>>print(cars[1])
>>>print(cars[-1])
>>>print(cars[-2])
```

## OUTPUT:

```
>>>audi
>>>benz
>>>bently
>>>tata
```

Drop A Love

Umamadhav Eedara
@umamadhav Eedara

# PYTHON TUPLES:

- Tuples are the same as a list but in a list, we can change the items whereas in tuples we can't change the items.
- Tuples items can be sorted in parentheses().

Example:

```
cars=("audi","benz","tata","bently")
```

Umamadhav Eedara
@umamadhav Eedara

## SYNTAX:

```
>>>cars=("audi","benz","tata","bently")
>>>print(cars)
```

## OUTPUT:

```
>>>("audi","benz","tata","bently")
```

**Umamadhav Eedara**
@umamadhav Eedara

# DICTIONARIES IN PYTHON:

- A Dictionary in python is a **collection** of **key-value** pairs.
- We can use a key to **access the value associated** with **that key**.
- A key's value can be a **number**, **string**, **list**, or even another **dictionary**.
- In python, a dictionary is wrapped in **Braces{}**.

Example:

```
Dog={"colour" : "Black", "Age" : 20}
```

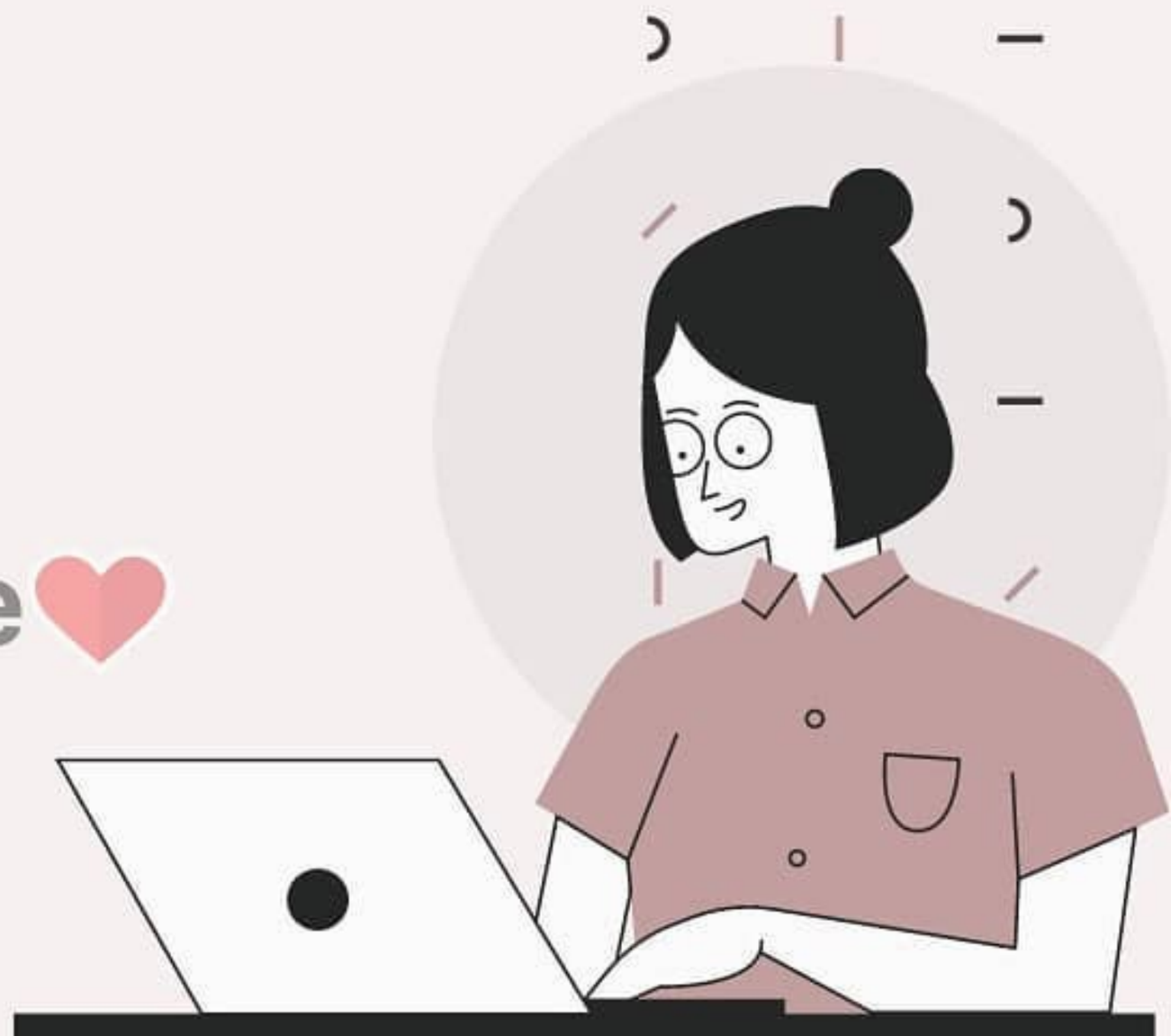
**key** **value**

**SYNTAX:**

```
>>>Dog={"colour" : "Black", "Age" : 20}
>>>print(Dog["Colour"])
>>>print(Dog["Age"])
```

**OUTPUT:**

```
>>>Black
>>>20
```



Drop A Love❤️

Umamadhav Eedara
@umamadhav Eedara

## Python Arithmetic Operators:

| Name | Operator |
|---|---|
| Addition | + |
| Subtraction | - |
| Multiplication | * |
| Division | / |
| Modulus | % |
| Exponentiation | ** |
| Floor division | // |

Umamadhav Eedara
@umamadhav Eedara

Drop A Love

**Umamadhav Eedara**
@umamadhav Eedara

## FOR LOOP IN PYTHON:

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- With the for loop, we can execute a set of statements, once for each item in a list, tuple, set etc..
- If you want the same action with every time in a Sequence or set types you can use "for loop"

**Umamadhav Eedara**
@umamadhav Eedara

## SYNTAX:

```
>>>fruits = ["apple", "banana", "cherry"]
>>>for x in fruits:
>>>    print(x)
```

## OUTPUT:

```
apple
banana
cherry
```

**Drop A Love**

## WHILE LOOPS IN PYTHON:

- While Loop **runs** as long as a certain **condition is true**.
- The while loop **requires** a **relevant variable** to be ready.

**Umamadhav Eedara**
@umamadhav Eedara

Umamadhav Eedara
@umamadhav Eedara

**SYNTAX:**

```
>>>def greeting ():
>>>    print("Hello!")
>>>gretting()
```

**OUTPUT:**

Hello!

Drop A Love

## OBJECT- ORIENTED PROGRAMMING [OOP] IN PYTHON:

- Object-Oriented programming is one of the most effective approaches to writing software.
- Python is a multi-paradigm programming language it supports different programming approaches.
- The concept of oop in python focuses on creating reusable code. This concept is also known as DRY [Don't repeat Yourself].

**Umamadhav Eedara**

@umamadhav Eedara

Umamadhav Eedara
@umamadhav Eedara

# IN PYTHON, THE CONCEPT OF OOP FOLLOWS SOME BASIC PRINCIPLES:

- **Class,**
- **Objects,**
- **Inheritance,**
- **Encapsulation, and**
- **Polymorphism**

Drop A Love

# PYTHON OBJECT- ORIENTED PROGRAMMING [OOP] CONCEPTS:

- **Classes**, and
- **Objects.**

**Umamadhav Eedara**

@umamadhav Eedara

Umamadhav Eedara
@umamadhav Eedara

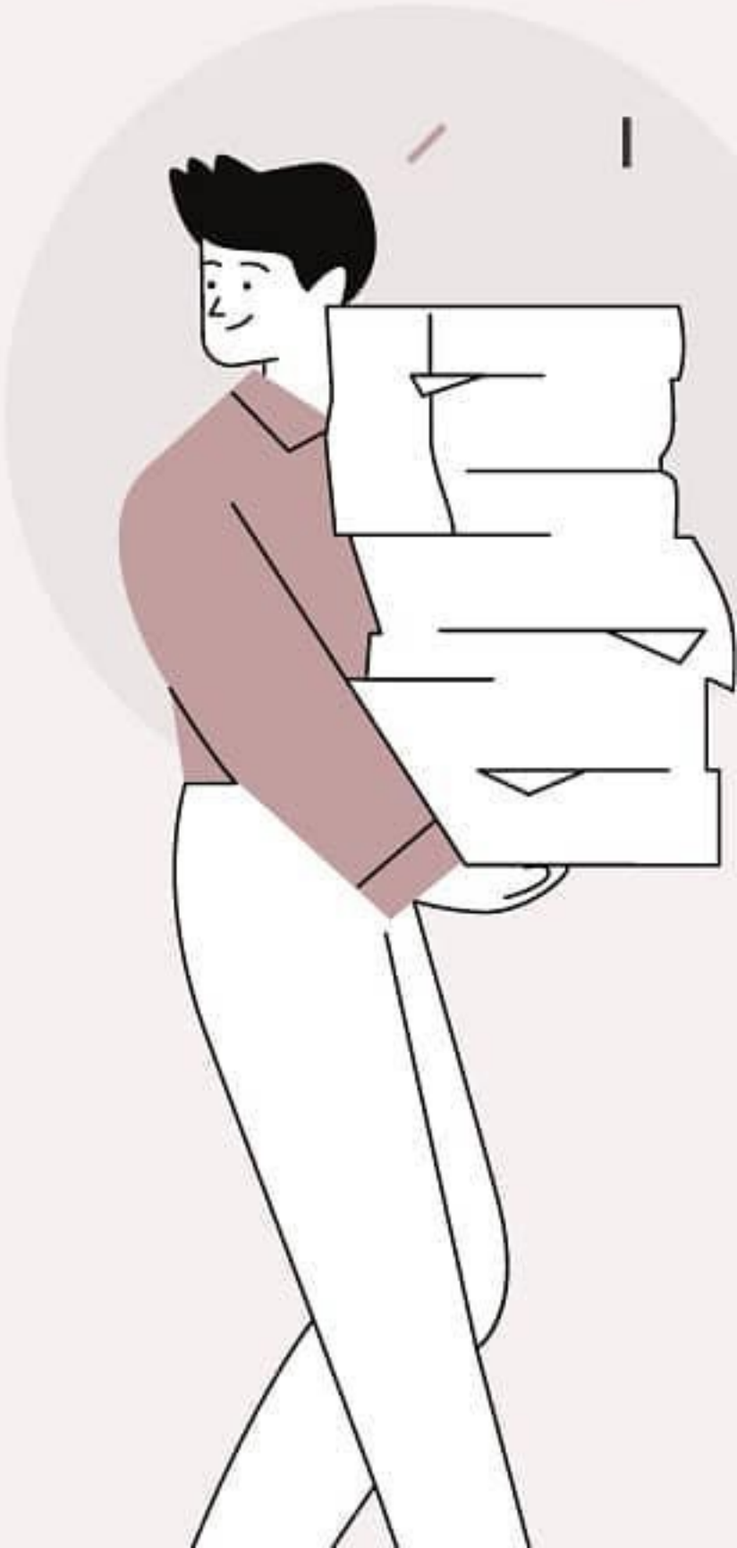
# CLASS:

- A Class is like an object constructor or a "blueprint" for creating objects.
- In oop's, you can write classes representing real-world things and situations and create objects based on these classes.
- To create a class, use the keyword class.
- When you write a class, you define the general attributes and methods that a whole category of objects can have.

Umamadhav Eedara
@umamadhav Eedara

# OBJECT:

- An object (Instance) is an Instantiation of a class.
- When class is defined, only the description for the object is defined.
- We can create as many objects as possible.



Look at the syntax and output >>>

**Umamadhav Eedara**
@umamadhav Eedara

**SYNTAX:**

```
>>>class Dogs:
>>>    def__init__(self,name,age):
>>>        self.name = name
>>>        self.age = age
>>>    def sit(self):
>>>        print(f"{self.name} is sitting")
>>>Dog = Dogs("Charlie", 10)
>>>print(Dog.name)
>>>print(Dog.age)
>>>Dog.sit()
```

**OUTPUT:**

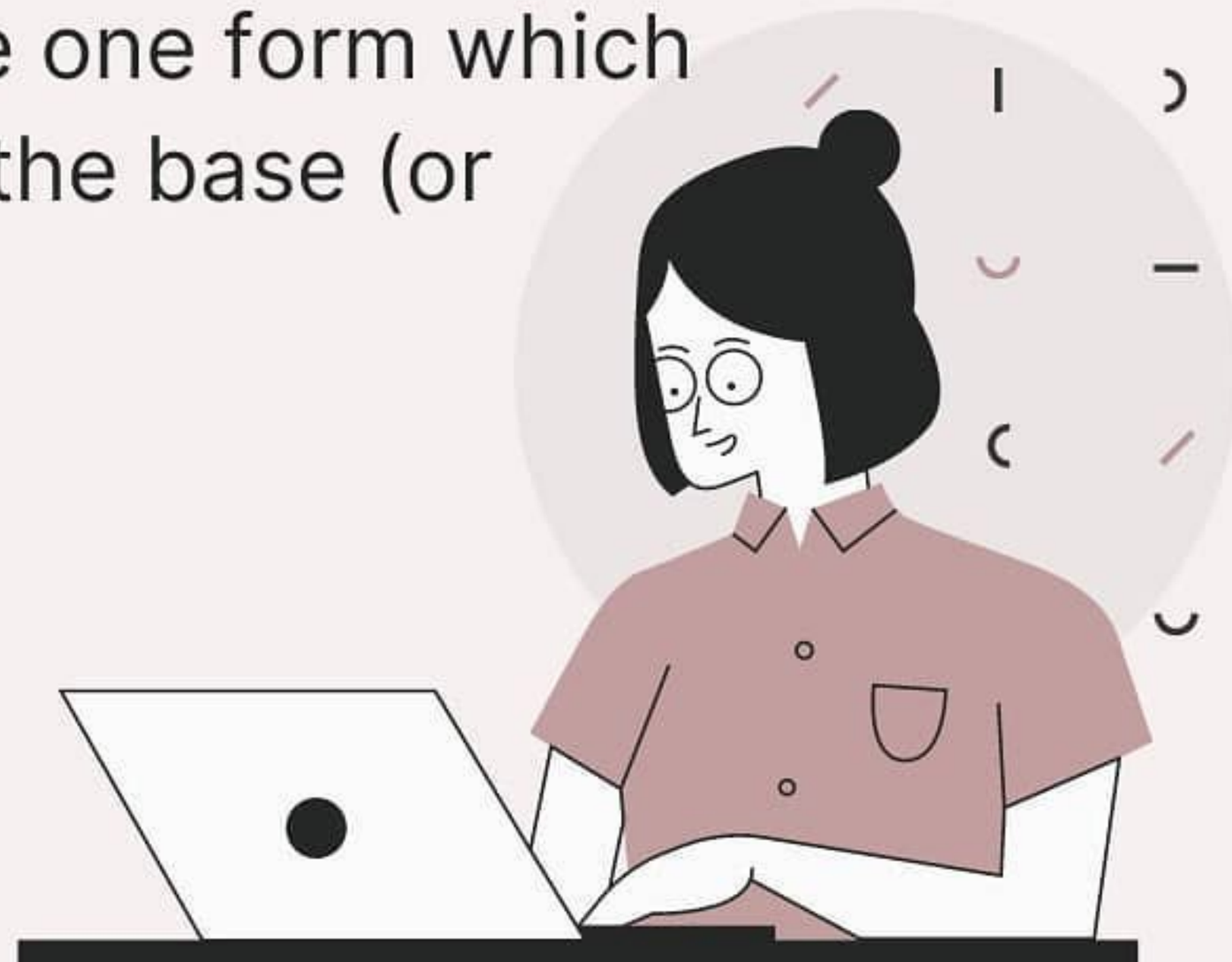
```
>>>Charlie
>>>10
>>>Charlie is sitting
```

**Drop A Love**

# PYTHON OBJECT- ORIENTED PROGRAMMING [OOP] INHERITANCE CONCEPT:

- Inheritance is a way of creating a new class for using all the methods and properties of an existing class without modifying it.
- The new class is called derived (or child) class and the one form which it inherits is called the base (or parental) class.

**Umamadhav Eedara**
@umamadhav Eedara



**Umamadhav Eedara**
@umamadhav Eedara

# THERE ARE 3 TYPES OF INHERITANCE:

- Single-level Inheritance.
- Multi-level Inheritance.
- Multiple-level Inheritance.

Look at the syntax and output of each one >>>

# Single-level Inheritance:

Umamadhav Eedara
@umamadhav Eedara

**SYNTAX:**

```
>>>Class A:
>>>    def function_1 (self):
>>>        print("Function_1 is working")
>>>Class B(A):
>>>    def function_2 (self):
>>>        print("Function_2 is working")
>>>a = A()
>>>b = B()
>>>b.function_1
>>>b.function_2
```

**OUTPUT:**

```
>>>Function_1 is working
>>>Function_2 is working
```

# Multi-level Inheritance:

**Umamadhav Eedara**
@umamadhav Eedara

**SYNTAX:**

```
>>>Class A:
>>>    def function_1 (self):
>>>        print("Function_1 is working")
>>>Class B(A):
>>>    def function_2 (self):
>>>        print("Function_2 is working")
>>>Class C(B):
>>>    def function_3 (self):
>>>        print("Function _3 is working")
>>>a = A()
>>>b = B()
>>>c = C()
>>>c.function_1
>>>c.function_2
>>>c.function_3
```

**OUTPUT:**

```
>>>Function_1 is working
>>>Function_2 is working
>>>Function_3 is working
>>>>uncupn_
>>>Function_3 i
```

# Multiple-level Inheritance:

**SYNTAX:**

```
>>>Class A:
>>>    def function_1 (self):
>>>        print("Function_1 is working")
>>>Class B:
>>>    def function_2 (self):
>>>        print("Function_2 is working")
>>>Class C(A,B):
>>>    def function_3 (self):
>>>        print("Function _3 is working")
>>>a = A()
>>>b = B()
>>>c = C()
>>>c.function_1
>>>c.function_2
>>>c.function_3
```

**OUTPUT:**

```
>>>Function_1 is working
>>>Function_2 is working
>>>Function_3 is working
```

**Umamadhav Eedara**
@umamadhav Eedara

Umamadhav Eedara
@umamadhav Eedara

# PYTHON OBJECT- ORIENTED PROGRAMMING [OOP] ENCAPSULATION CONCEPT:

- In Object-oriented Python programming, you can restrict access to methods and variables.
- This can prevent the data from being modified by accident and is known as Encapsulation.
- Encapsulation can be achieved using private variables and private methods.
- In python, we denote private attributes using single or double underscore as the prefix.

Look at the syntax and output >>>

**SYNTAX:**

```
#Private variable can be accessed only within the class
>>>class number:
>>>    __a = 10
>>>    def num (self):
>>>        print(self.__a)
>>>n = number
>>>n.num
```

**OUTPUT:**

```
>>>10
```

**Umamadhav Eedara**

@umamadhav Eedara

Umamadhav Eedara
@umamadhav Eedara

**SYNTAX:**

```
#Private methods can be accessed only within the class
>>>class number:
>>>    def __num:
>>>        print("10")
>>>    def num:
>>>        self.__num()
>>>n = number
>>>n.num
```

**OUTPUT:**

```
>>>10
```

## PYTHON OBJECT-ORIENTED PROGRAMMING [OOP] POLYMORPHISM CONCEPT:

- In Object-oriented Python programming, Polymorphism is the ability to use a common interface for multiple forms.
- suppose, we needed to colour a shape, there are multiple shapes like[Rectangle, Triangle, and Circle ]. However, we could use the same method to colour all the shapes. This is called Polymorphism.

**Umamadhav Eedara**

@umamadhav Eedara

Umamadhav Eedara
@umamadhav Eedara

# THERE ARE FOUR WAYS OF IMPLEMENTING POLYMORPHISM:

- **Duck typing,**
- **Operator overloading,**
- **Method overloading,** and
- **Method overriding.**

Look at the meaning, syntax, and output of each one >>>

## DUCK TYPING:

- Duck typing is a concept related to dynamic typing.Where, the type ot the class of an object is less important than the methods it defines.You do not check types at all. Insted, you can check for the presence of a given method or attribute.

**SYNTAX:**

```
>>>Class Duck:
>>>    def sound(self):
>>>        print("Quack Quack!')
>>>Class Cat:
>>>     def sound(self):
>>>         print("Meow Meow!!')
>>>def all_sounds(obj):
>>>    obj.sound()
>>>x = Duck()
>>>all_sounds(x)
```

**OUTPUT:**

```
>>>Quack Quack!
```

**Umamadhav Eedara**
@umamadhav Eedara

## METHOD OVERLOADING:

- Method overloading is the ability of a function or an operator to behave in different ways based on the parameters that are passed to a function.

**SYNTAX:**

```
>>>class A:
>>>    def __init__(self):
>>>        pass
>>>    def sum (self, A = none, B = none, C = none):
>>>        s = 0
>>>        if (A != none and B != none and C != none):
>>>            s = A+B+C
>>>        elif (a != none and B != none):
>>>            s = A + B
>>>        else:
>>>            s = A
>>>        return s
>>>a = A()
>>>a.sum(2,6)
```

**OUTPUT:**

```
>>>6
```

**Umamadhav Eedara**

@umamadhav Eedara

## OPERATOR OVERLOADING:

- If any operator performs additional actions other than what it is meant for, it is called operator overloading.

**SYNTAX:**

```
>>>class student:
>>>    def__init__(self, M1 , M2):
>>>        self.M1 = M1
>>>        self.M2 = M2
>>>    def __add__(self, other):
>>>        m1 = self.M1 + other.M1
>>>        m2 = self.M2 + other.M2
>>>        s3 = student(m1 , m2)
>>>        return s3
>>>s1 = student(10, 20)
>>>s2 = student(30, 40)
>>>s3 = s1 + s2
>>>print(s3.m1)
```

**OUTPUT:**

```
>>>40
```

**Umamadhav Eedara**
@umamadhav Eedara

## METHOD OVERRIDING:

- Method overriding is a concept of oop that allows us to change the implementation of a function in the child class that is defined in the parent class. It is the ability of a child class to change the implementation of any method which is already provided by one of its parent class(ancestors).

### SYNTAX:

```
>>>class A:
>>>    def show (self):
>>>        print( "A Show")
>>>class B(A):
>>>    def show (self):
>>>        print("B show")
>>>b = B()
>>>b.show()
```

### OUTPUT:

```
>>>B show
```

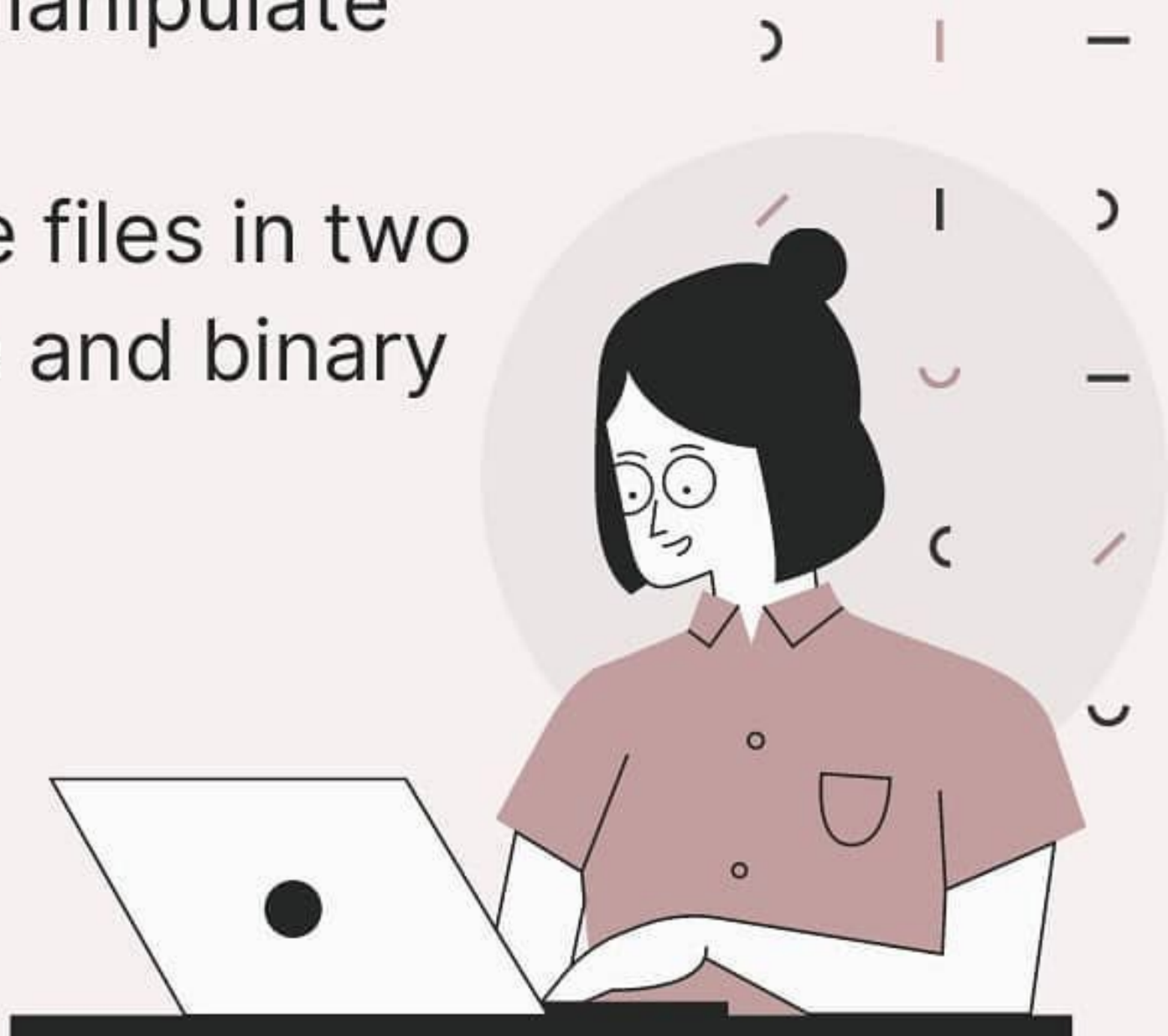
**Umamadhav Eedara**

@umamadhav Eedara

Umamadhav Eedara
@umamadhav Eedara

# WHAT IS FILE HANDLING IN PYTHON:

- A file is some information or data which stays in the computer storage devices. Python gives you easy ways to manipulate these files.
- Generally we divide files in two categories, text file and binary file.



Umamadhav Eedara
@umamadhav Eedara

## TEXT FILES:

- Text files don't have any specific encoding and they can be opened in a normal text editor itself.

**Examples:**

- Web standards: html, XML, CSS, JSON etc.
- Source code: c, app, js, py, java, etc.
- Documents: txt, tex, RTF, etc.

## BINARY FILES:

- Most of the files that we see in our computer system are called binary files.

**Examples:**

- Document files: .pdf, .doc, .xls etc.
- Image files: .png, .jpg, .gif, .bmp etc.
- Video files: .mp4, .3gp, .mkv, .avi etc.
- Audio files: .mp3, .wav, .mka, .aac etc.

Drop A Love