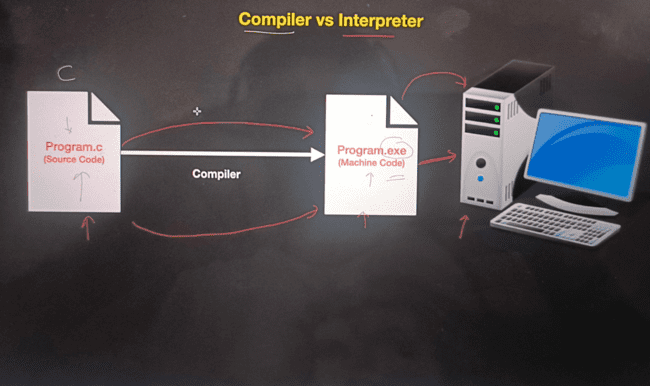
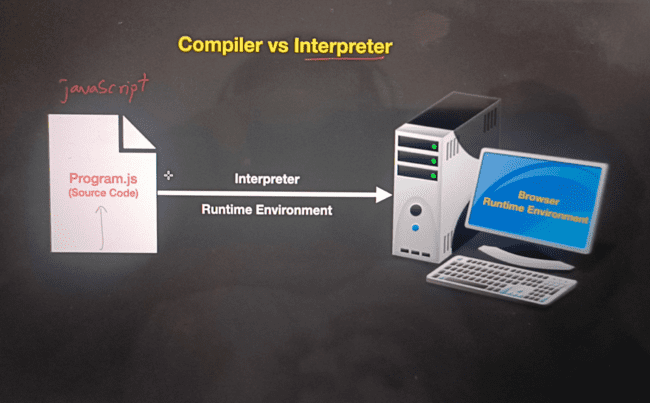
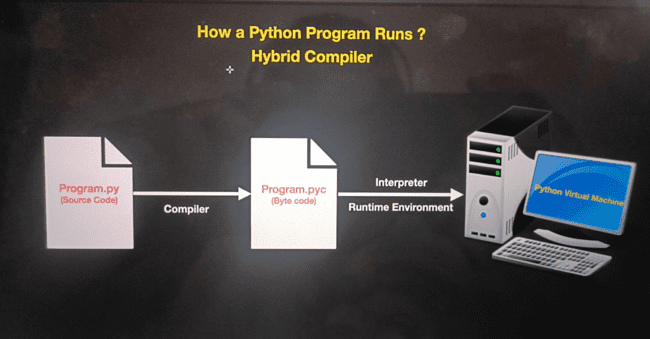
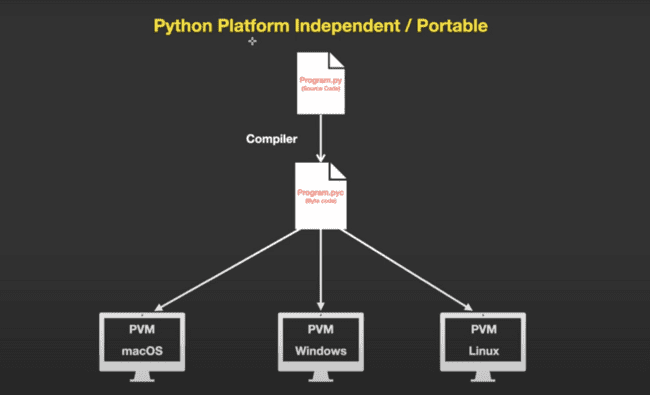
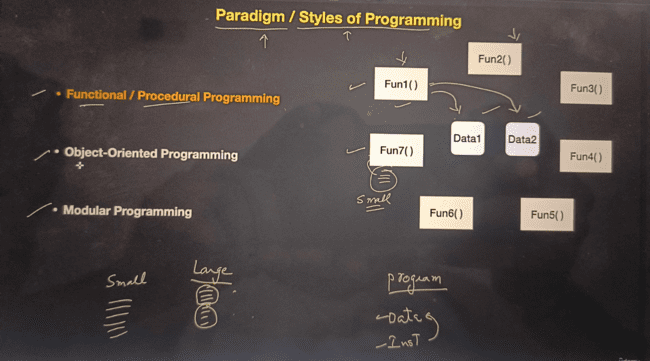
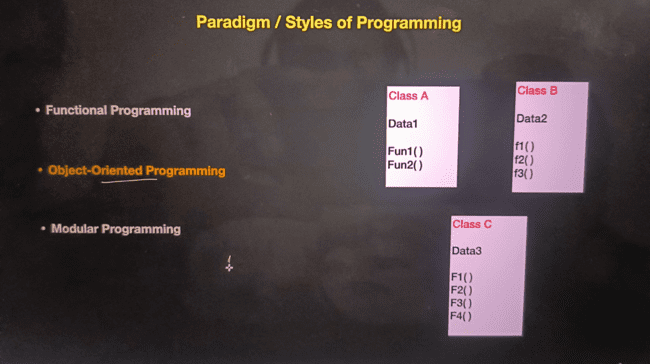
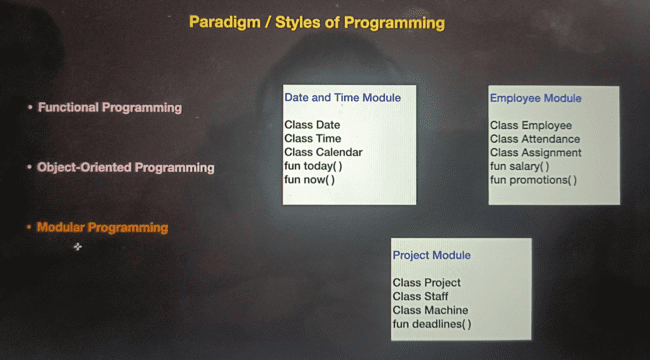
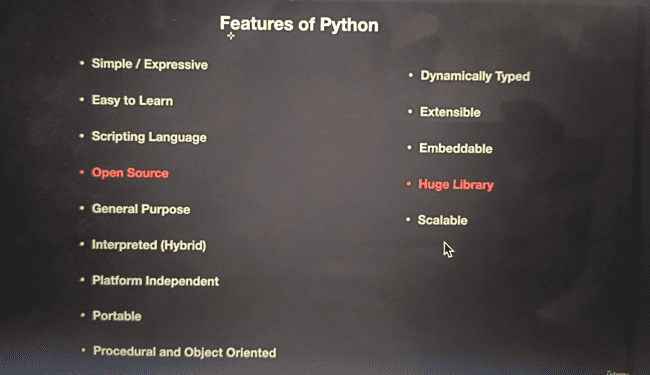
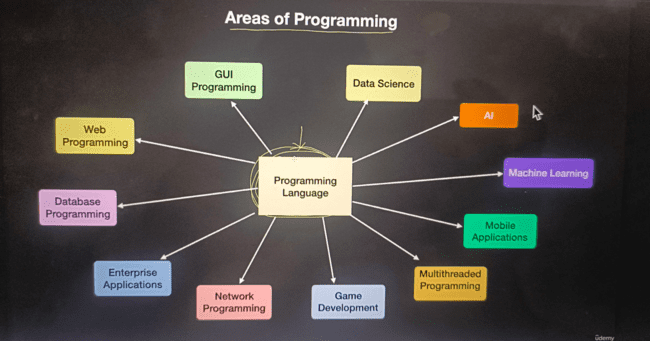
## Introduction to python

1. **Types of programing languages**a**)** Low level language : machine level and assembly   
   b) High level language: compiler based, interpreter based, hybrid   
   Note: python will come under Hybrid
2. **Compiler VS interpreter**a**)** Compiler is used to translate to exe file which machine will understand   
   b) Interpreter will directly execute its runtime environment   
   Note: (for ex in Interpreter when we run code it will take single piece of code or all code and directly execute it in machine code but in compiler will convert first to exe and we have to execute when we want)  
   c. Example javascript run inside browser so here browser is interpreter  
     
   **Compiler  
   **  
     
     
     
     
     
     
     
     
   **Interpreter**   
   
3. **Hybrid** **Compiler**  
   a. When we run python code it will convert source code to intermediate code called bytecode and then byte code will convert to machine code through interpreter (PVM)  
   b. Byte code will create as hidden file in memory in the file extension program.pyc   
   Ex: java, python   
   
4. **Python platform independent**Python will run on different different platform for same code  ****
5. **Paradigm / style of programming**   
   a) When we write large number of code its difficult to manage hence we have to adopt bellow methods to organize our code its called as Paradigm  
   b) Python support bellow all Paradigm  
     
   **Paradigm have various methods**  
   **a. Functional / procedural programing :** a.1)large program broken down into functions, function is piece of code which perform specific task   
    a.2) it is function centric more focus on function   
     
     
   **b. Object oriented programing**  
   b.1) We have a class called A the class contain Data1 and function the function used to call data   
     
     
     
     
     
     
     
      
   **c. modular programing:**   
   a big program break down into modules and module may contain class and functions together



1. **Features of python**   
   
2. **Areas of programing**

## Python installation and Setup

1. **Installing python**
2. **Installing pycharm**
3. **Installing VS code**
4. **Installing and using jupyter Notebook**

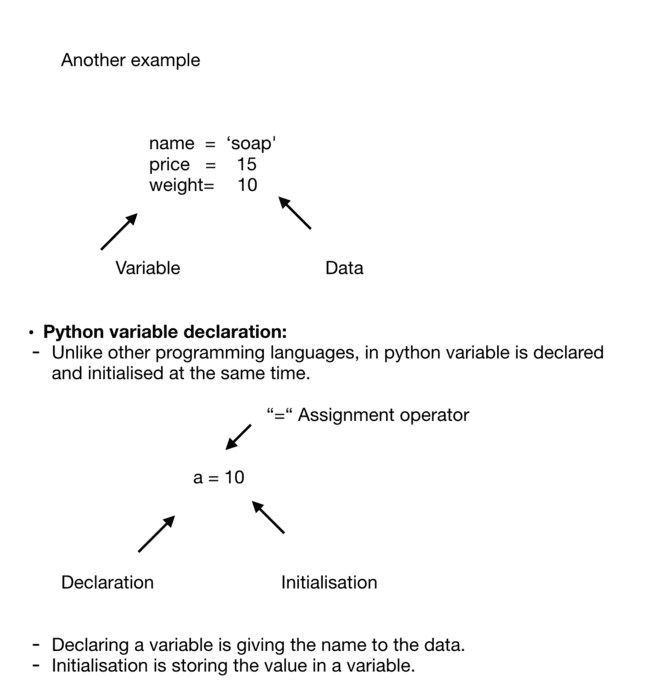
## Python DataTypes

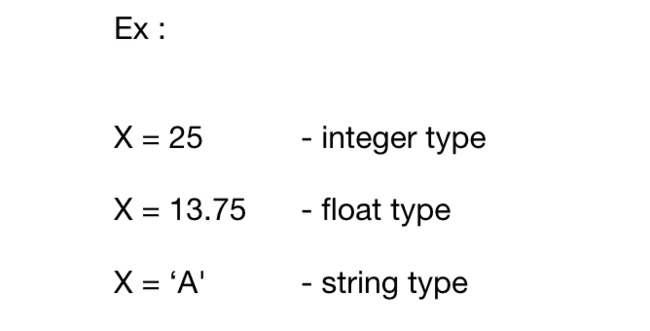
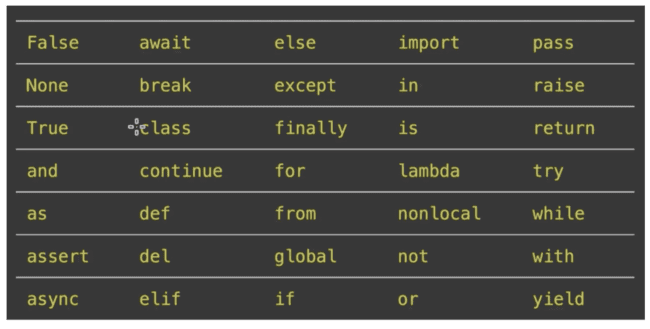
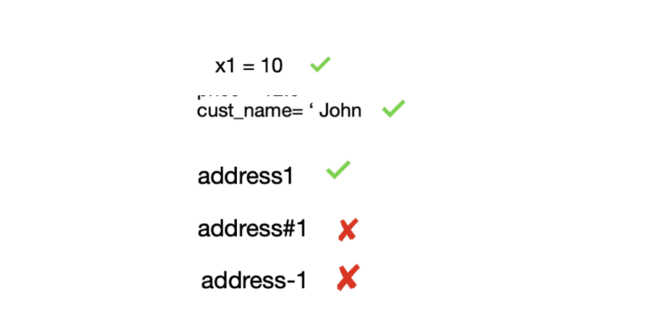
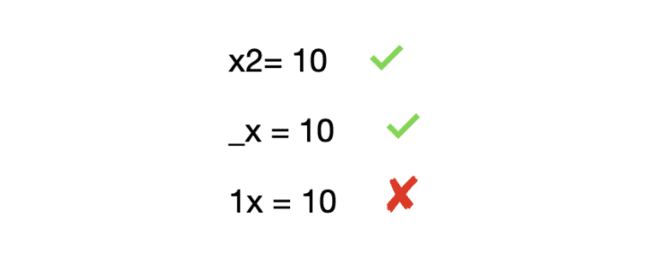
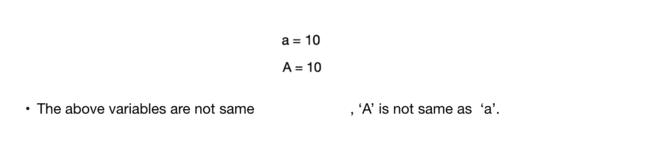
1. **Section introduction   
   1.a) Numeric Types  
   2.b) variables   
   3.c) Literals  
   4.d) conversions**
2. **What are variables**

a.) Variables are names given to data or referral to data

b.) Variables are the names given to the data stored in a memory location.  
 Ex a=10, name=’soap’

Note: in above example ‘a’ called as declaration and assigning value to a is called as initialization

**Declaration of multiple variables**a, b, c=15, 10, 12 or X=Y=Z=1  
Name, price, weight=’cv’, 50, 100  
  


1. **Dynamically typed language:** Python is called a dynamically typed language because you don't have to declare the data type of a variable when you create it. Instead, the type of a variable is determined at runtime  
     
   Variables do not have a specific data type , it’s type depends on the value assigned to it  
     
   
2. **Rules for Declaring a variable  
   a)** we cannot use keywords to declare a variable.These words cannot be used while declaring the variable.  
    a.1) Keywords : the words which are predefined in the language are called keywords or reserved words  
     
     
   b) The first rule says that we can mix alphabet and numbers while declaring the variables, we can even use an underscore.  
     
   c) However we cannot use any special symbol like $, &,@,#,- etc…  
     
     
   d) The second rule says that the variable must start with a letter or underscore character only.  
     
     
     
   e) The last rule states that variables are case sensitive,   
   
3. **Python dataTypes**   
   **a) Numeric:** it can support one value   
    **a.1) int:** storing integer value called integer, we can store how much lengthy we   
    want (no sizeLimit in python) and also we can store negative value as well   
    EX: a=1234567890, b=10, c = -10  
    **a.2) float:** storing decimal value called float   
    EX: a=12.59, b= -12.59, c=12345.67890   
    **a.3) bool:** it is logical data like True and False   
    Note: True value is 1 and False value is 0  
    **a.4) complex:**   
    EX: x=3+4j, c=complex(12,9)

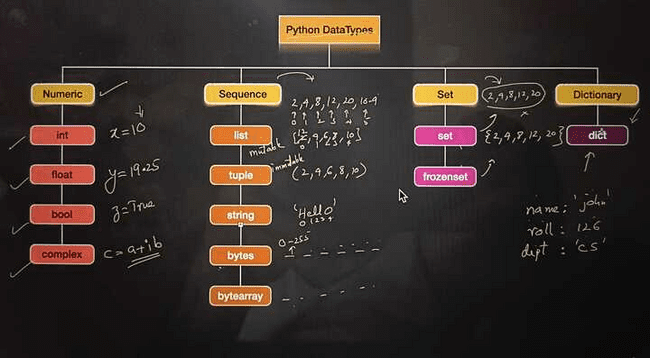
**b) Sequence :** collection of values, value could be integer, float, or char, but in sequence every values have index

**b.1) List:** list is collection of values here value can be modified (mutable)  
 **b.2) tuple:** tuple is also sequence of collection of value but in tuple value can not  
 be modified (immutable)  
 **b.3) Strings:** collection of numbers symbols, words or char   
 **b.4) Bytes:** it contain one byte of data and each value should be in between

0 to 255 only it is immutable

**b.5) Bytearray:** it contain one byte of data and each value should be in  
 between 0 to 255 only it is mutable

**c) set :** set is also collection of values but set values are not index based   
 **c.1) Set:** set is collection of values but value are un-ordered and it is mutable   
 **c.2) frozenset:** it is set of value but it is freezed value (cannot be modified)   
 (im-mutable)

**d) Dictionary:**   
 **d.1) Disct:** it contain key and value, it is very useful for storing data and  
 retrieving Data   
 Example: name : ‘cv’ (name is key, value is CV )  
  
**Note:**   
**mutable** means value can be changed   
**im-mutable** means previous value can not be modified   
**For Ex:** first variable “A” value assigned 10, later we changed “A” value to 20, in python previous value of “A” will not changed it will store in memory but only change is “A” will start pointing to new value 20 its called **immutable**  
   


1. **Literals and constant**   
     
   **a) Literals:** Literals are simply the raw data values that are used to represent data types. They are fixed values that are directly written in the code.

**EX: Examples of literals**

number\_literal = 42 # Numeric literal

string\_literal = "hello" # String literal

boolean\_literal = True # Boolean literal  
  
In these examples, 42, "hello", and True are literals representing an integer, a string, and a boolean value, respectively. They are directly written in the code and represent specific values.

**b) Constants:** Constants are identifiers (variable names) that are used to represent fixed values in the code. They are usually used to make the code more readable and maintainable by giving meaningful names to fixed values.  
  
**Example of constants (by convention)**

TAX\_RATE = 0.2 # Constant representing tax rate

MAX\_SPEED = 100 # Constant representing maximum speed

In this example, TAX\_RATE and MAX\_SPEED are constants representing the tax rate and the maximum speed, respectively. By convention, we use uppercase names with underscores to indicate that these variables should be treated as constants

1. **Base conversions**   
   Base conversion in Python involves converting a number from one numerical base system to another  
     
   bin() converts a decimal number to its binary representation.

int() with base 2 converts a binary string to its decimal equivalent.

hex() converts a decimal number to its hexadecimal representation.

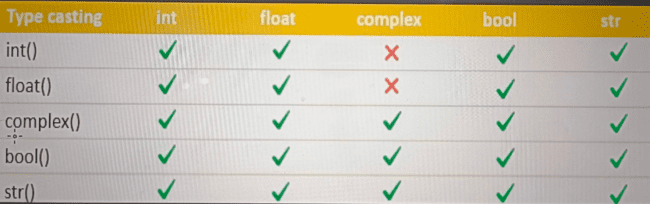
int() with base 16 converts a hexadecimal string to its decimal equivalent.

1. **Type conversions**Converting one data type to another data type using below mentioned functions is called as Type conversions  
     
   **1) Implicit**

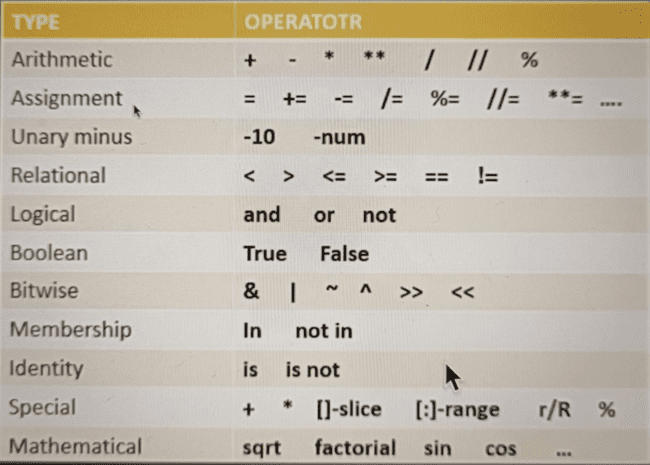
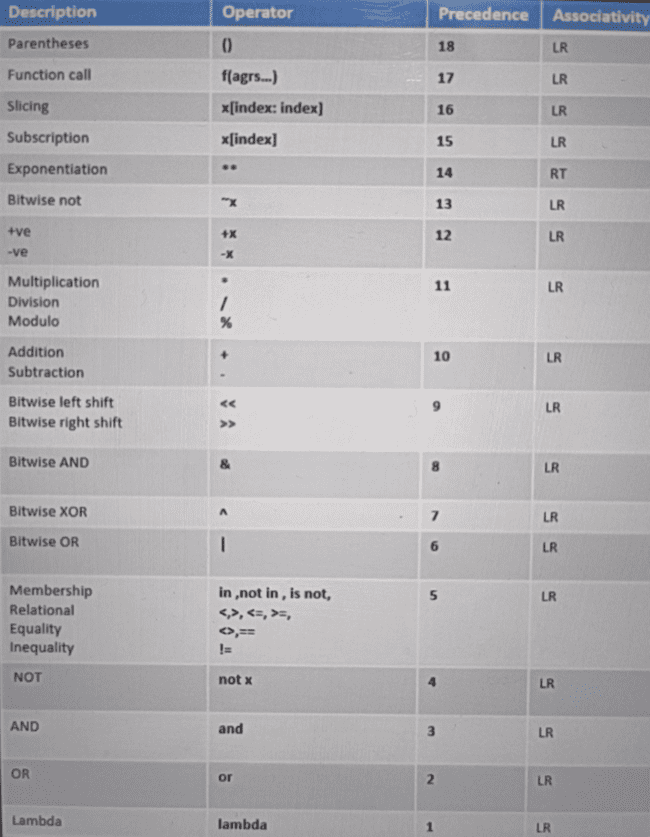
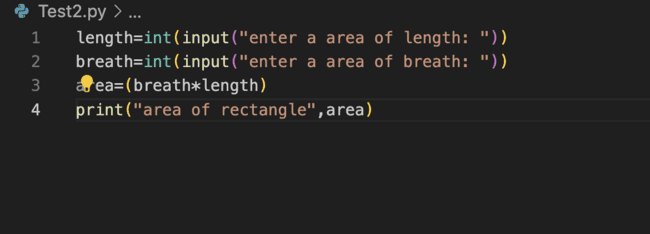
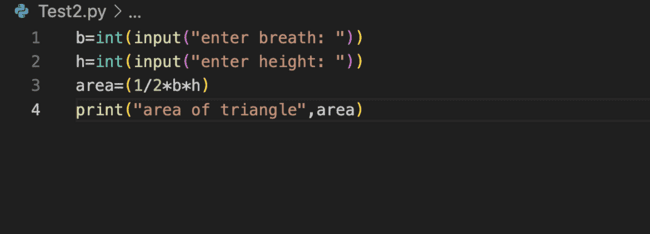
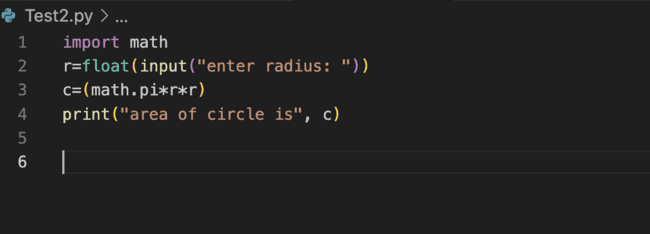
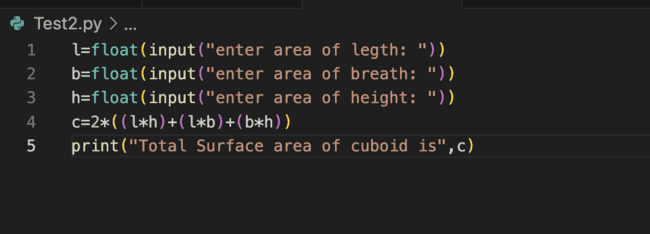
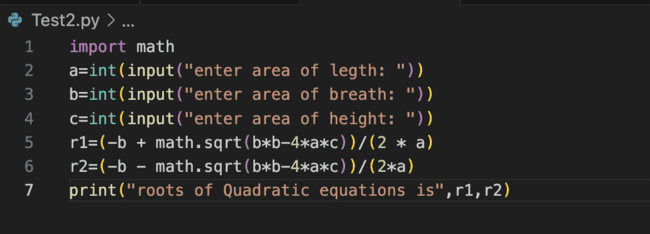
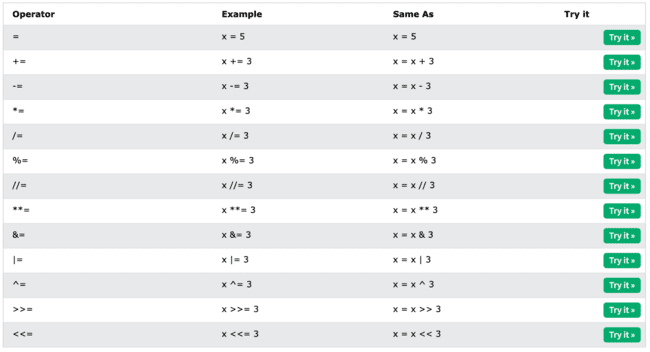
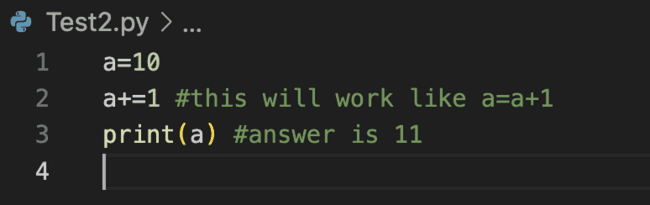
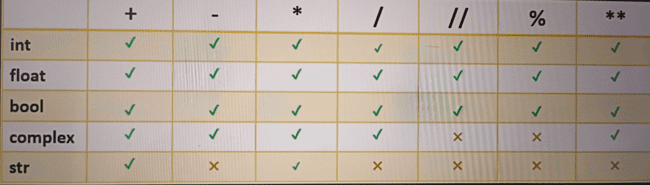
**2) Explicit**

**implicit :** In Implicit type conversion of data types , the Python interpreter automatically converts one data type to another without any user involvement.

**Explicit :** The programmer has to convert and programmer have to mention   
  
**Functions:** int(), float(), Bool(), complex(), str()

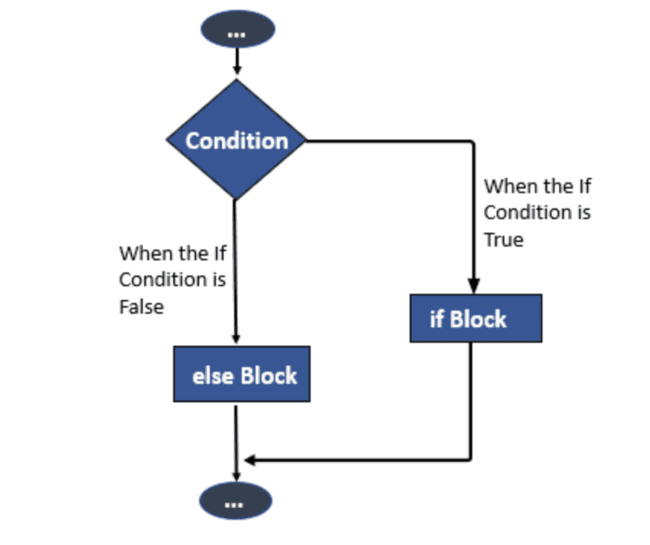
**Note:**   
a) complex number cannot be converted to integer, float   
b) String also cannot be converted into integer, float, complex if its alphabets but if the string variable is in digits then its possible to convert. (for ex a=’1234’)  
c) all data types can be converted to Bool() and str()  
  


## Operators and Expression

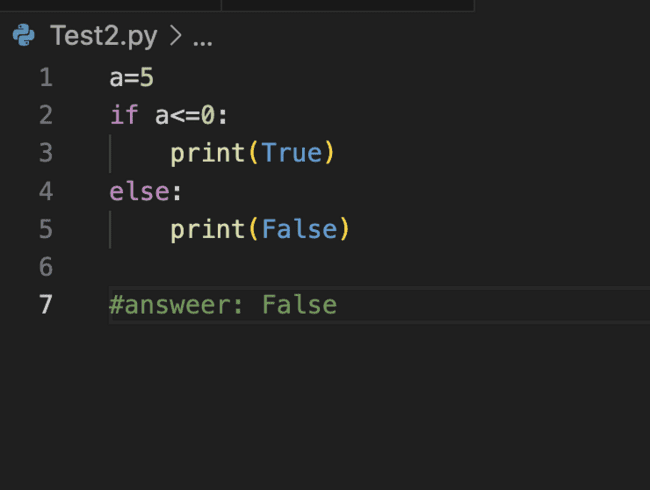
1. **Operators:   
   **
2. **Expression:**  
   an expression is a combination of values, variables, and operators that evaluates to a single value. Expression in python can be used in a variety of ways, such as in assignment statements, function arguments, and conditional statements  
     
   In Python, expressions are executed based on precedence, which determines the order in which operators are evaluated in an expression. Operators with higher precedence are evaluated before operators with lower precedence. If operators have the same precedence, the expression is evaluated from left to right.  
   
3. **Programs using Expressions**   
   Write program on area of rectangle   
   Formula for rectangle: (area=length \* breath)  
     
   
4. **Challenges:**  
   **a) find Area of triangle**   
     
     
   **b) converting KM to Miles**  
     
     
   c) write program on find Area of circle   
     
     
   d) Total surface area of cuboid  
     
     
     
     
     
   e) finding roots of Quadratic equations   
   
5. **Assignment Operators**  
   In Assignment Operators if we use “a+=1” it will first calculate a+1 and then store the result value into the “a”. like this it will work for all the below mentioned operators   
     
     
   **EX:**  
   
6. **Arithmetic with all Data Types**  
   

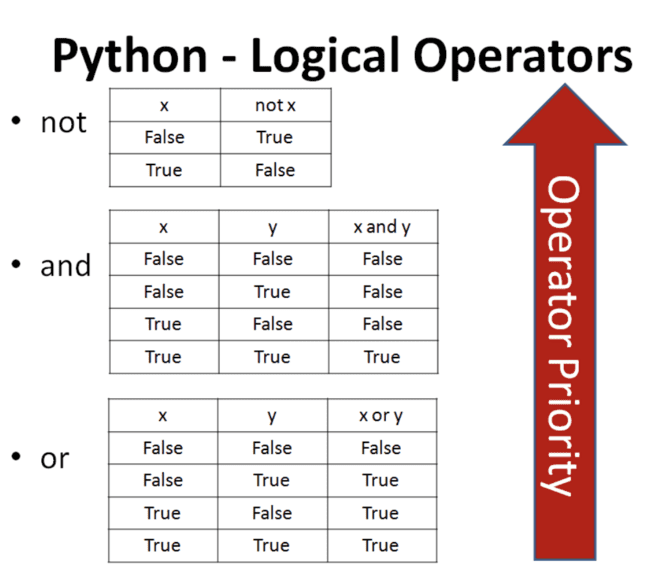
## Conditional Statement

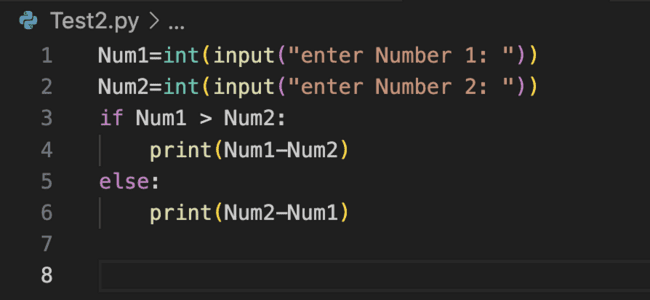
1. **If else Conditional**

****

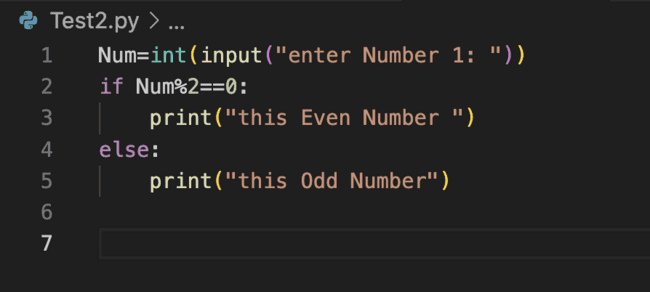
**EX: for if else condition**

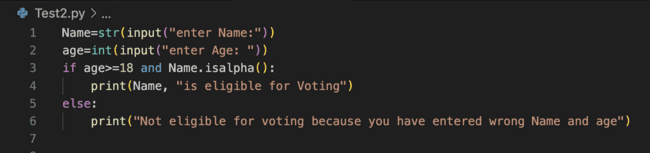
****

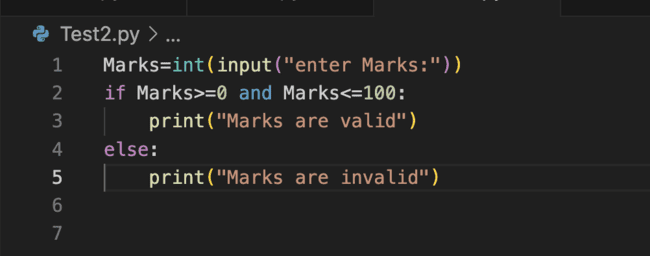
1. **Logical operator in if else Condition   
   **
2. **Challenges on if else condition**

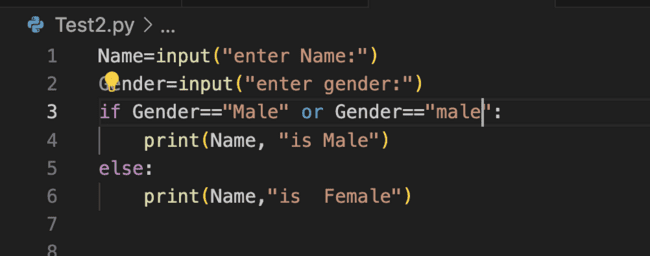
**a) Find difference between two number and the result value should be in positive value   
**

**b) Check if number is odd or Even**

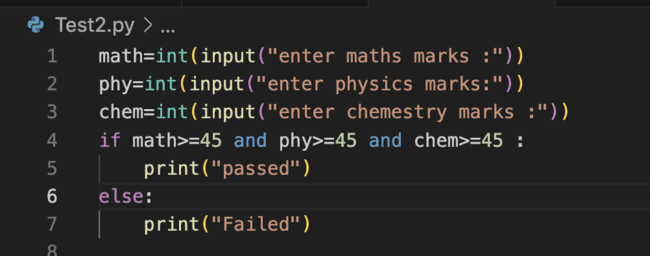
****

**c) Check for Age Eligibility for casting Vote  
**

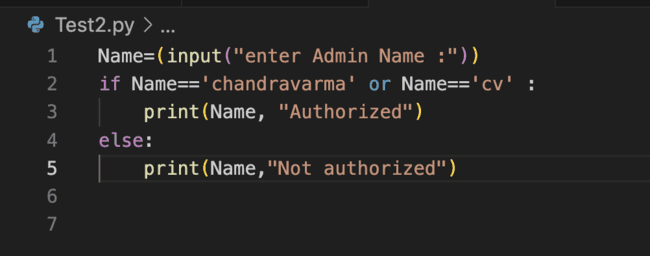
**d) check if marks of subject are within range 0-100  
**

**e) check if person is Male or Female   
**

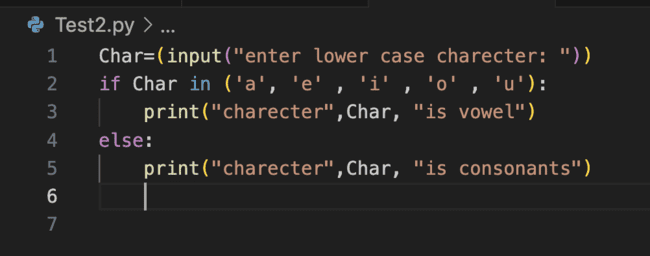
**f) check student has passed or failed, by taking marks in 3 subjects**

****

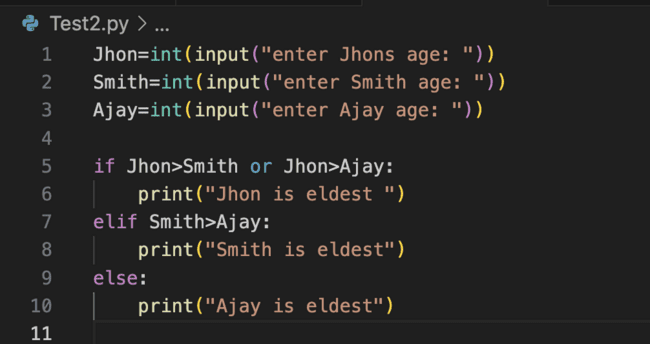
**g)check if person is authorized for admin access**

****

**h)check if given lower case character is vowel or consonants**

****

1. **Nested if and elif Statement**

**a) check who is eldest in Jhon, Smith, Ajay using Nested if and elif  
**

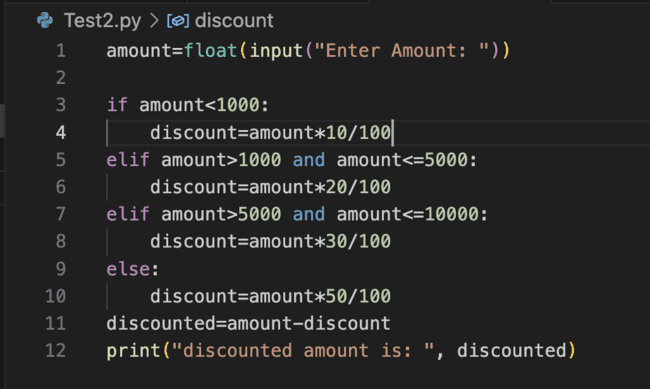
**b)Calculate Discounted amount for given Discount Details using elif condition**

amount <1000 discount is 10%

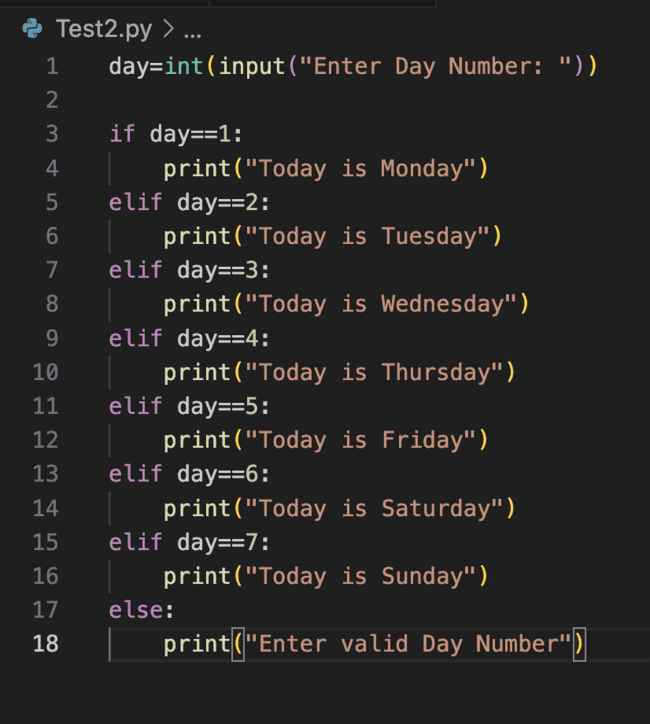
amount >1000 and <=5000 discount is 20%

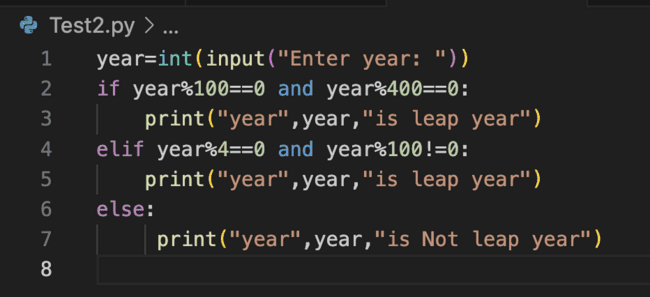
amount >5000 and <=10000 Discount is 30%

amount >10000 Discount is 50%

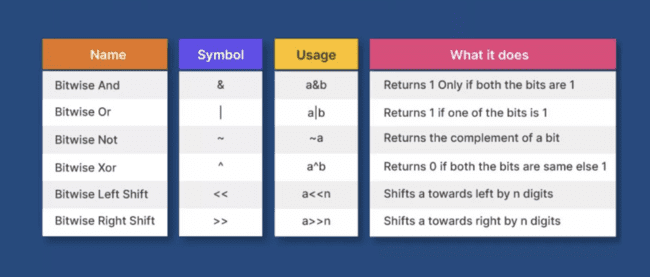
****

**c)Take a day number 1 to 7 and display day name using else if**

****

**d)check whether the year is leap year or not **

1. **Relational operators**Relational operator with Data Types****
2. **Bitwise Operator**

****

**a)Bitwise And operator in binary (&)**  
1 and 1 =1

1 and 0 =0

0 and 1 =0

0 and 0 =0  
  
**b)Bitwise OR operator in binary ( | )**

1 OR 1 =1

1 OR 0 =1

0 OR 1 =1

0 OR 0 =0

**c)Bitwise XOR operator in binary (^)**

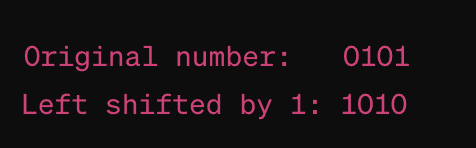
1 XOR 1 =0

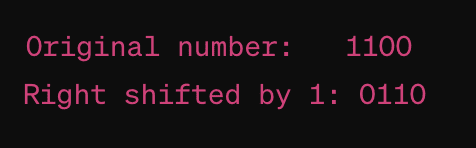
1 XOR 0 =1

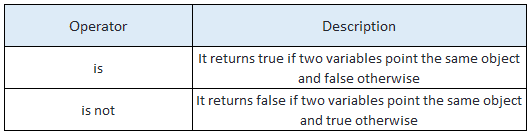
0 XOR 1 =1

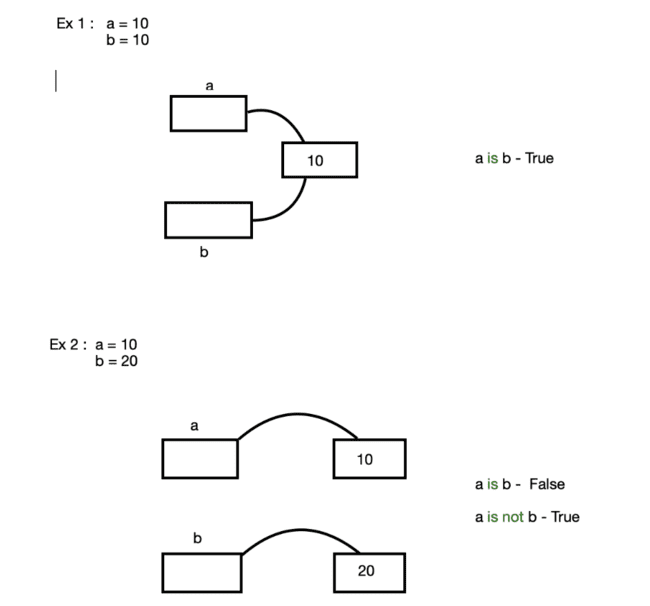
0 XOR 0 =0

**d)Bitwise Left Shift operator in binary (<<)**

Consider the binary number 0101 (which represents the decimal number 5 in binary). If we perform a left shift by 1 bit (0101 << 1), it would look like this:  


**e)Bitwise Right Shift operator in binary (>>)**Consider the binary number 1100 (which represents the decimal number 12). If we perform a right shift by 1 bit (1100 >> 1), it would result in: ****

1. **Identity Operator**  
   **Note:** it will work when you directly assign value to a variable or using “Int” data type to accept input but when you accept input values from the user without data type declaration python creates new memory even when the given values are the same.

**EX**:  


## Loops / Control flow

**1) Control Flow**

a) While loop

b) For loop

c) Continue, break, pass

d) Infinite loop

e) Match case

**2) Introduction to Loop’s**

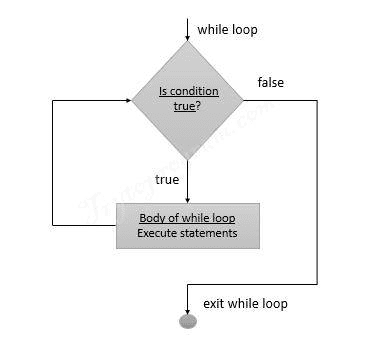
a) Loops are also called repeating statements.

b) If you want a set of statements to repeat again and again in the program then we use

loops The statements can repeat either ‘for number of times’ or ‘As long as the condition

is true’

**While Loop**

  
**EX**: print hello 10 time  
count=0

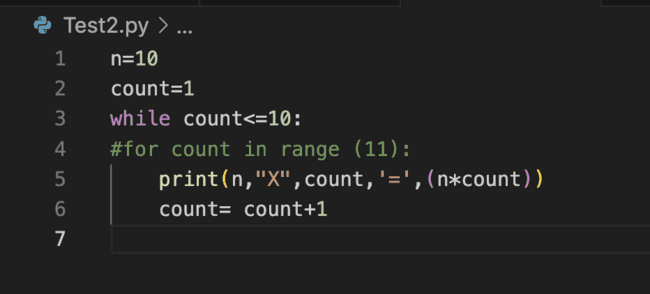
while count<10:

print(“Hello”)

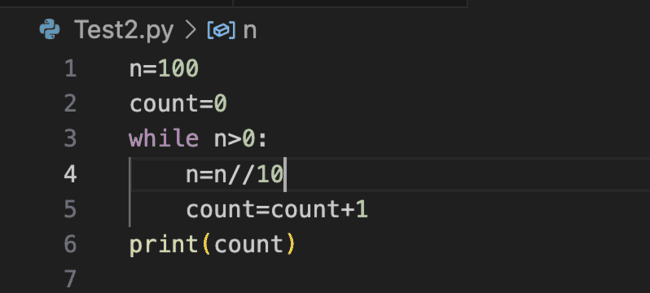
count= count+1

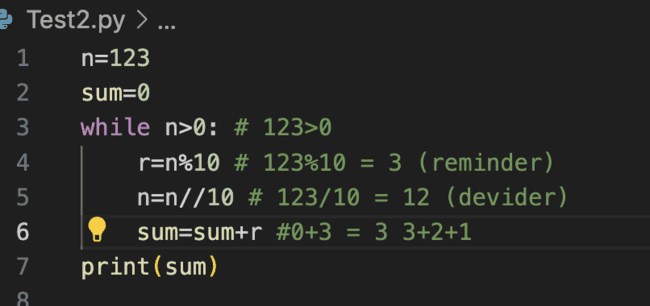
**3) While loop Challenges**

3.1) Display Multiplication table for given number

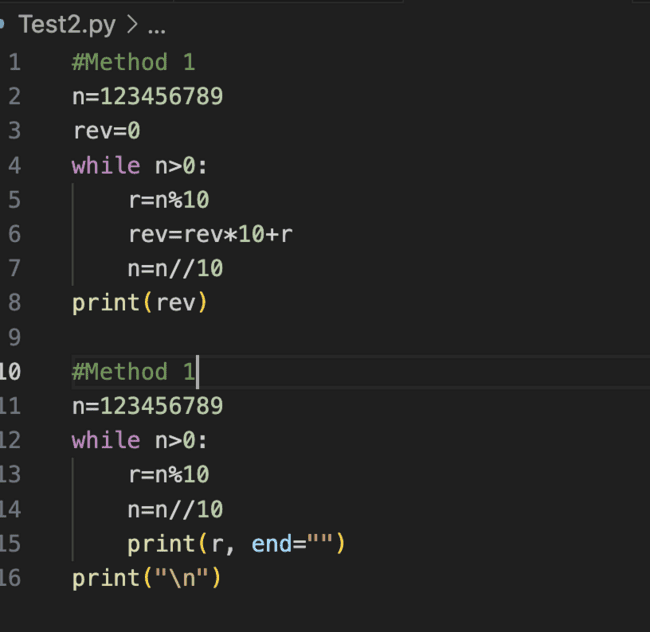


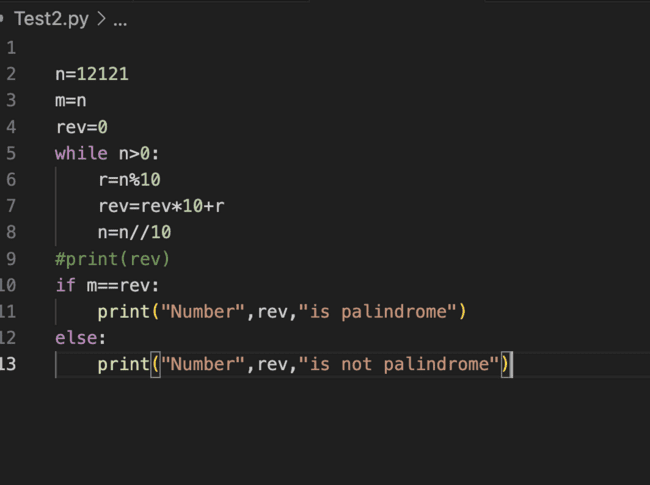
3.2) counting number of digits in number



3.3) find sum of digit in number   


3.4) Reversing number

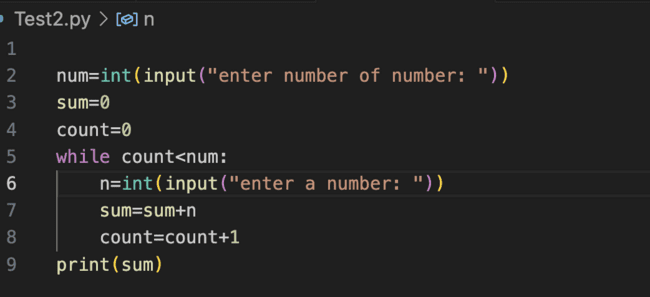


3.5) check given number is palindrome or not   


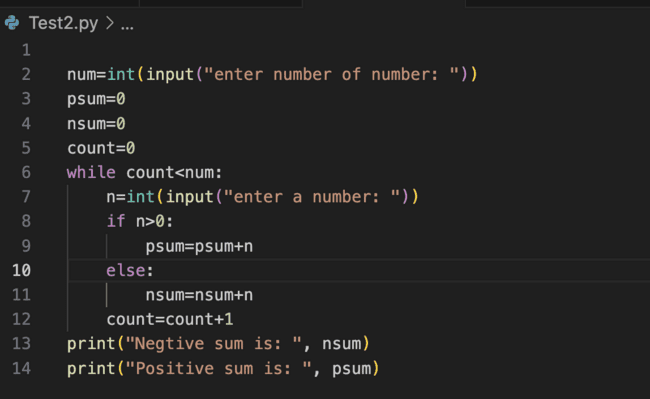
3.6) find sum of given number as input   
**Note:**

First take number from user how many time he want to enter number

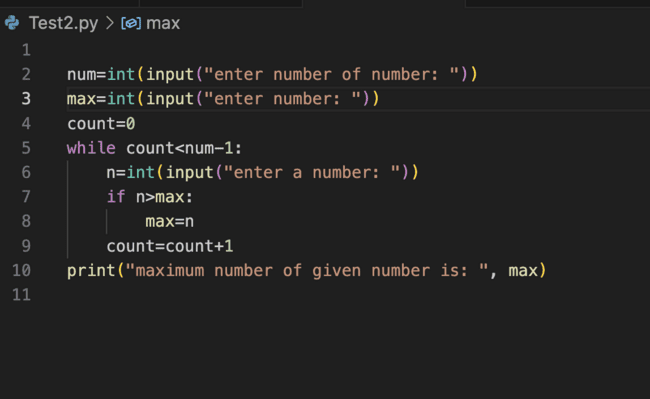
Then make sum of entered number



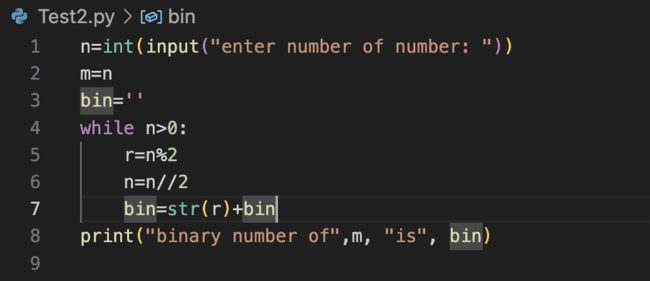
3.7)print sum of -ve value and +ve value



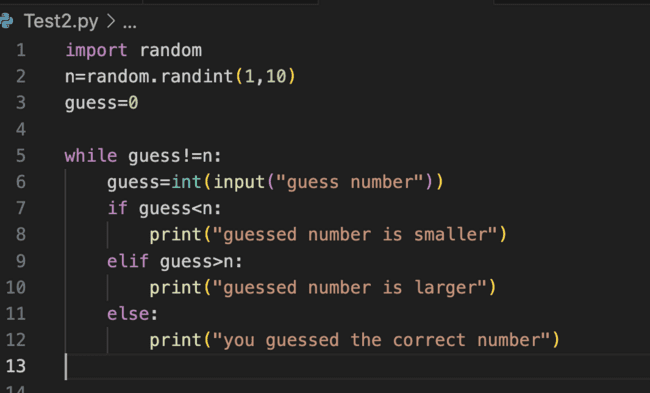
3.8) find Maximum number of given numbers



3.9) convert decimal number to binary



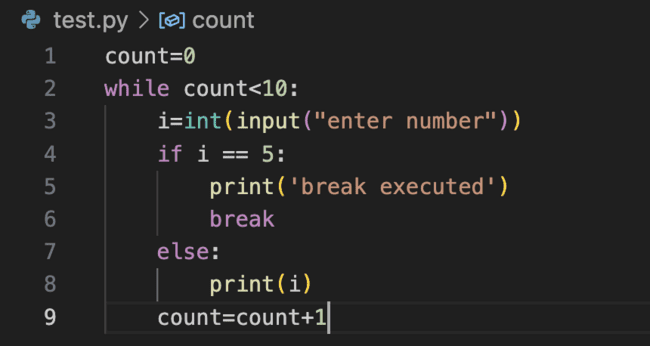
3.10) guess number between 1 to 10



**4) infinite loop - Break - Continue - Pass with while loop**

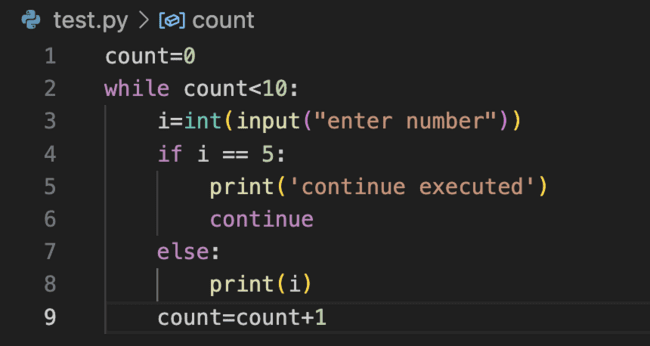
**4.1) Break:**

In the Break if condition is true then it will print “break executed” and then it will break statement and come out of loop, if its fail it will print(i) value until while loop will get fails

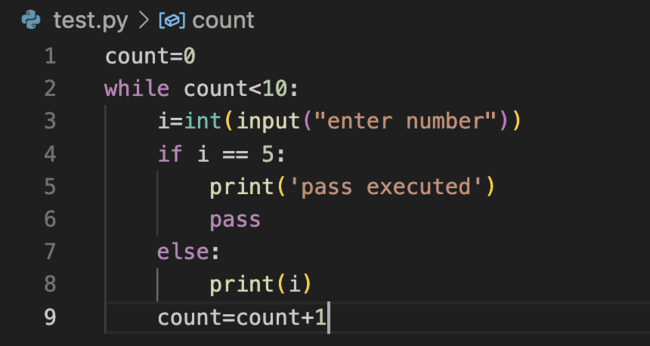


**4.2) Continue:**

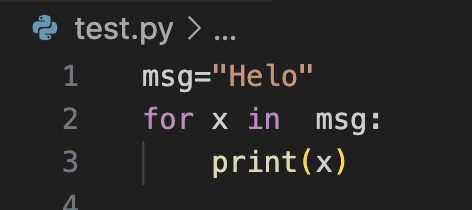
In continue, based on bellow example it will not increment count value bcz it will continue back to loop if condition is true, when if condition will fail then it will print else part and increment count value

****

**4.3) Pass**In pass, based on bellow example it will check if condition if it is true and then it will pass cursor to the count increment (but in continue cursor will not move to count increment statement if condition is true)

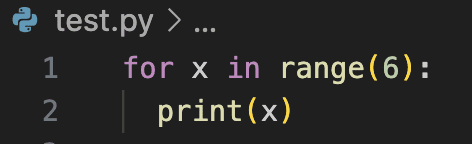
****

**5) For Loop:**

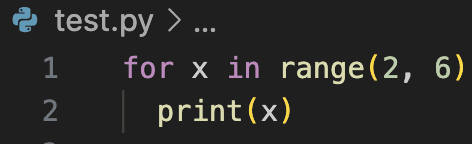
A Python for loop is a type of loop which is used to iterate a set of sequential statements multiple times (like a list, tuple, or string) Iterating over a sequence means going through each element one by one ****

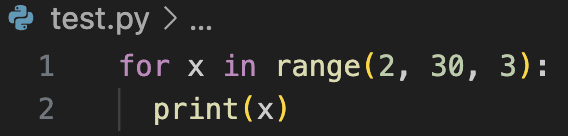
**5.1) For loop range() Function**

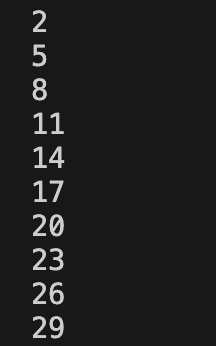
5.1.1) The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.  
Ex:

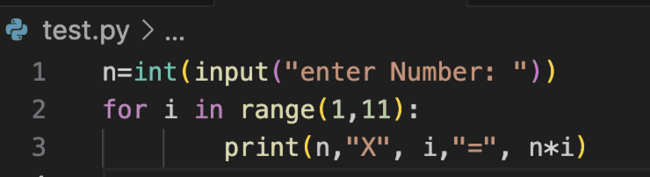


5.1.2) The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6)  
Ex:

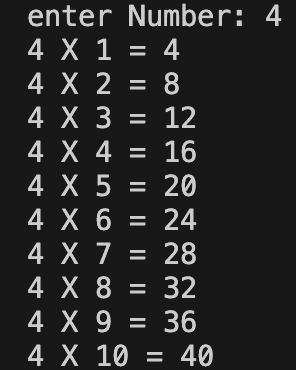
  
  
5.1.3)The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(2, 30, 3)  
Ex:

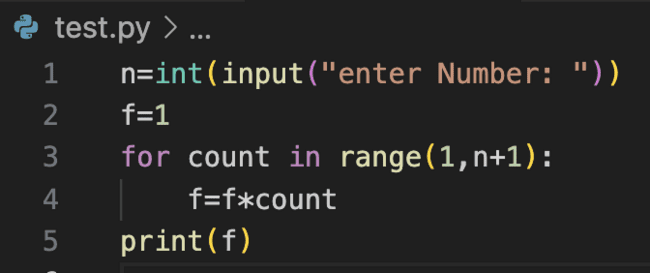


Output**:   
  
**

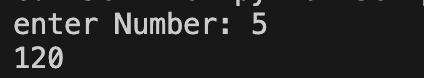
**6) For loop Challenges**6.1) Display Multiplication table for given number   


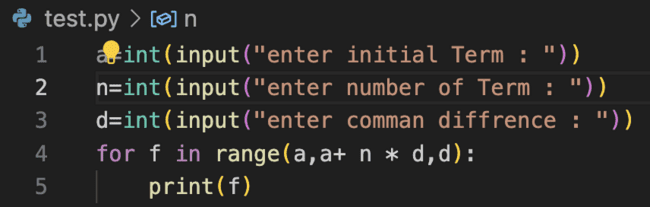
Output:



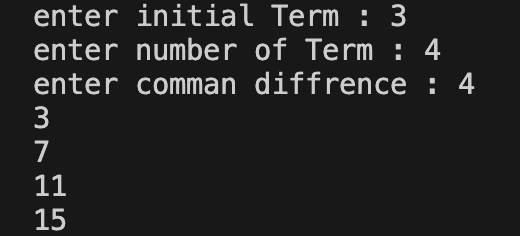
6.2) Find Factorial of given number   


Output:

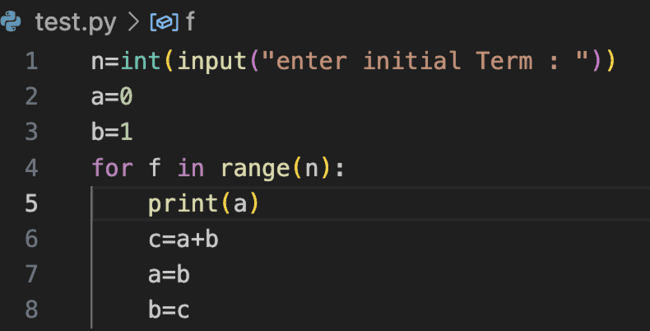


6.3) Print n term of AP (Arithmetic progression) series   


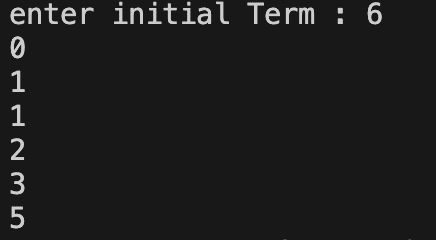
Output:

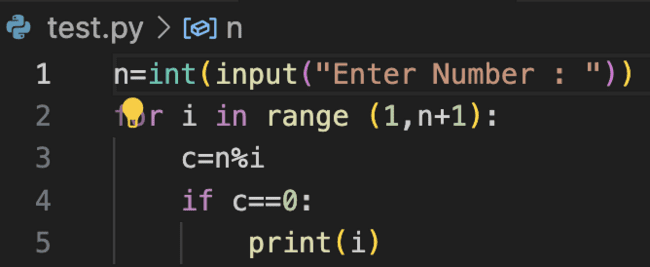


6.4) print n term of fibonacci series

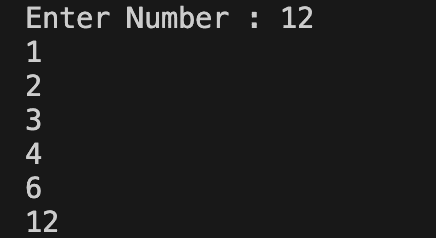


Output:

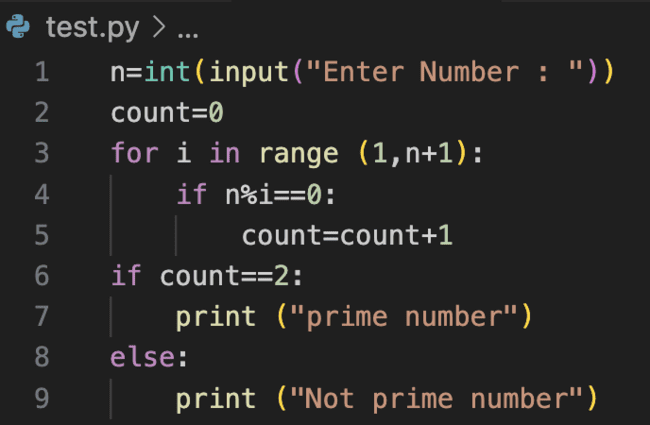


6.5) Find the factor of number   


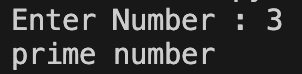
Output:



6.6) check if number is prime or not



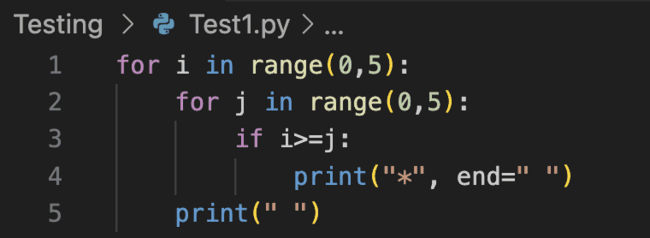
Output:



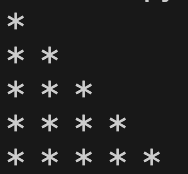
**7) Break , Continue, Pass with For loop**Break, continue, and pass work like same as above explain in while loop   
**Refer** (infinite loop - Break - Continue - Pass with while loop)

**8) Nested Loop:** A nested loop is a loop inside another loop. This means that one loop is contained within another loop. Each time the outer loop executes, the inner loop will execute completely.

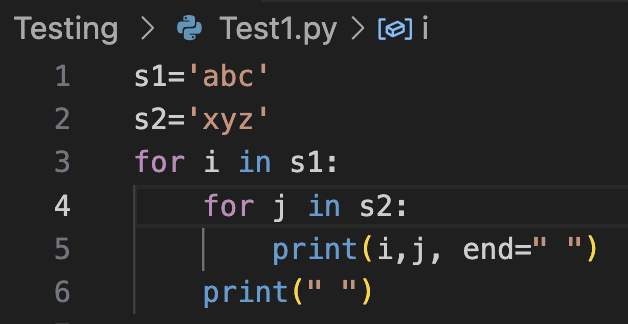
Example 1:



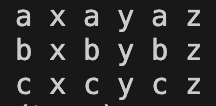
Output



Example 2:

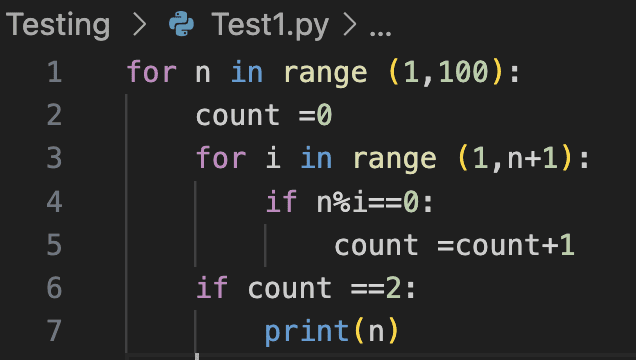


Output:

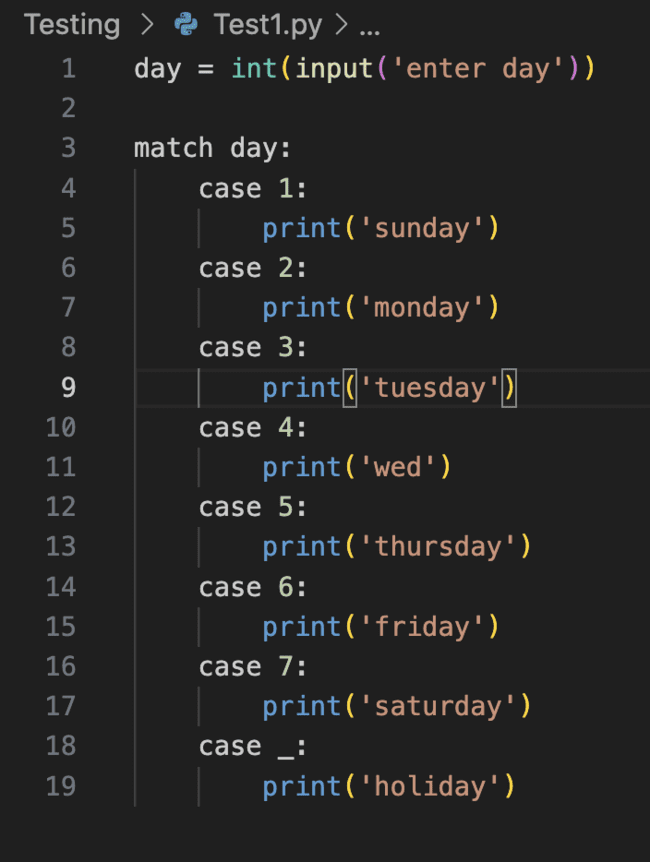


**8.1**) **challenges on Nested loop**

8.1.1) Print prime number from 1 to 100 (prime number is number is only divided by 1 and 2)



**9) Match Case**

****

## Strings and its Methods

**1) introduction to string**

a string is a sequence of characters enclosed within either single quotes (') or double

quotes (").

EX: a=’cv’ b=”Jhon’s”

Note: if Apostrophe ‘ is used in string we have to use double quotes

**2) Operator on string**

**2.1) concatenation**

It means combining two or more strings/lists into one longer string. You can do this using

the + operator:

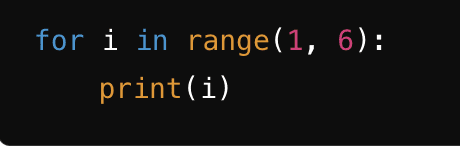
**Ex**: a= ‘how’+’are’+’you’ (how are you)

**2.2) Repetition**

Repetition in Python is commonly handled using loops, which allow you to execute a

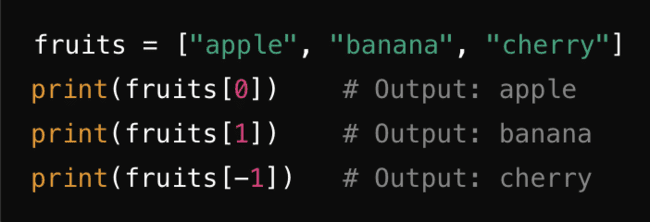
block of code multiple times

Ex:

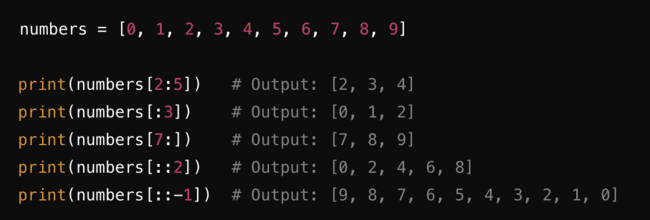


**2.3) indexing:**Indexing in Python refers to accessing elements of a sequence using their

positions. Python sequences that support indexing include lists, tuples, strings, and more   
 EX:



**2.4) slicing:** Slicing in Python allows you to access a subset of elements from sequences like lists, tuples, and strings. Slicing is done using the colon : operator within square brackets [].   
EX:

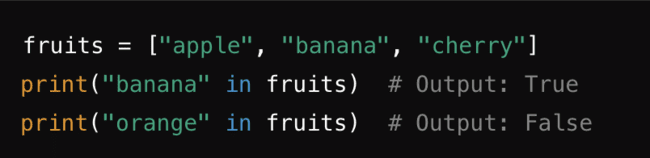


**2.5) in and not in :** used to check if a value is present in a sequence (like a list, tuple,

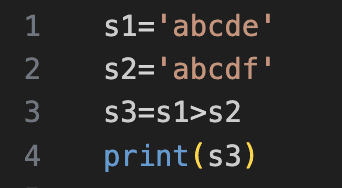
string, or set). These operators return True or False based on whether the specified

value is found in the sequence.

Ex:



**3) Relational Operator on String**

****

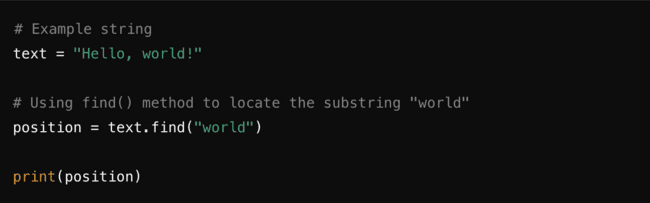
**4) String Methods**

| Mthods | Description |
| --- | --- |
| capitalize() | Converts the first character to uppercase |
| casefold() | Converts string into lower case |
| center() | Returns a centered string |
| count() | Returns the number of times a specified value occurs in a string |
| encode() | Returns an encoded version of the string |
| endswith() | Returns true if the string ends with the specified value |
| expandtabs() | Sets the tab size of the string |
| find() | Searches the string for a specified value and returns the position of where it was found |
| format() | Formats specified values in a string |
| format\_map() | Formats specified values in a string |
| index() | Searches the string for a specified value and returns the position of where it was found |
| isalnum() | Returns True if all characters in the string are alphanumeric |
| isalpha() | Returns True if all characters in the string are in the alphabet |
| isascii() | Returns True if all characters in the string are ascii characters |
| isdecimal() | Returns True if all characters in the string are decimals |
| isdigit() | Returns True if all characters in the string are digits |
| isidentifier() | Returns True if the string is an identifier |
| islower() | Returns True if all characters in the string are lower case |
| isnumeric() | Returns True if all characters in the string are numeric |
| isprintable() | Returns True if all characters in the string are printable |
| isspace() | Returns True if all characters in the string are whitespaces |
| istitle() | Returns True if the string follows the rules of a title |
| isupper() | Returns True if all characters in the string are upper case |
| join() | Converts the elements of an iterable into a string |
| ljust() | Returns a left justified version of the string |
| lower() | Converts a string into lower case |
| lstrip() | Returns a left trim version of the string |
| maketrans() | Returns a translation table to be used in translations |
| partition() | Returns a tuple where the string is parted into three parts |
| replace() | Returns a string where a specified value is replaced with a specified value |
| rfind() | Searches the string for a specified value and returns the last position of where it was found |
| rindex() | Searches the string for a specified value and returns the last position of where it was found |
| rjust() | Returns a right justified version of the string |
| rpartition() | Returns a tuple where the string is parted into three parts |
| rsplit() | Splits the string at the specified separator, and returns a list |
| rstrip() | Returns a right trim version of the string |
| split() | Splits the string at the specified separator, and returns a list |
| splitlines() | Splits the string at line breaks and returns a list |
| startswith() | Returns true if the string starts with the specified value |
| strip() | Returns a trimmed version of the string |
| swapcase() | Swaps cases, lower case becomes upper case and vice versa |
| title() | Converts the first character of each word to upper case |
| translate() | Returns a translated string |
| upper() | Converts a string into upper case |
| zfill() | Fills the string with a specified number of 0 values at the beginning |

**5) String method Examples**

**5.1 Find() :** this method used to find the substring in the string

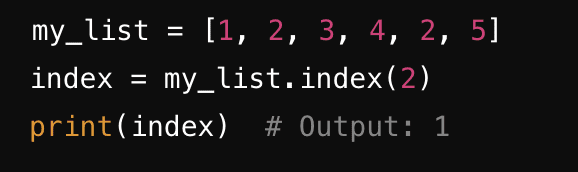
Ex:



**5.2 index():** the index() method is used to find the index of the  
 first occurrence of a specified value within a sequence such as a

list, tuple, or string

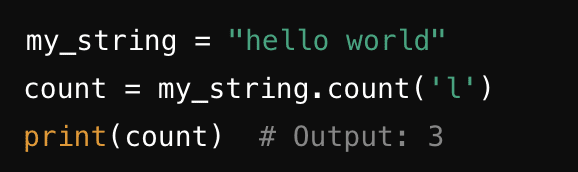
EX:



**5.3 count()**: The count() method in Python is used to count the number of

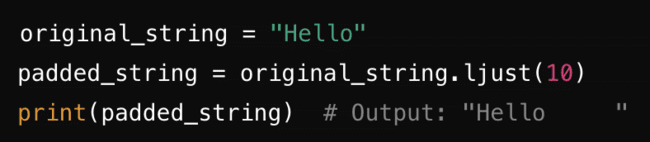
occurrences of a specified element in a sequence such as a list, tuple, or string

Ex:



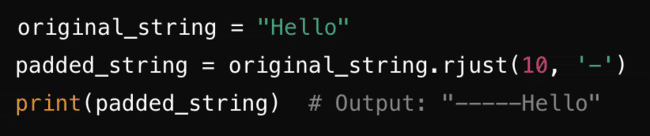
**5.4 ljust()**:

EX:



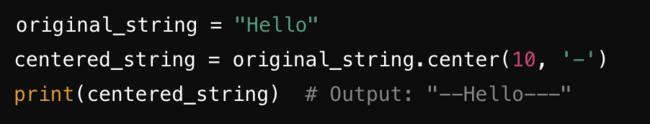
**5.5 rjust()**:

Ex:



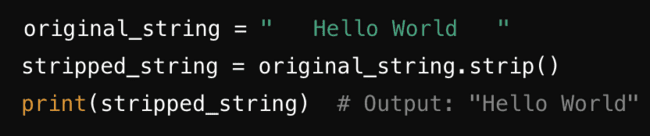
**5.6 Center():**

Ex:

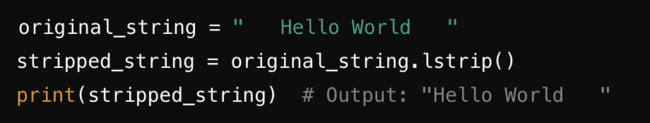


**5.7 strip():** remove space from both side

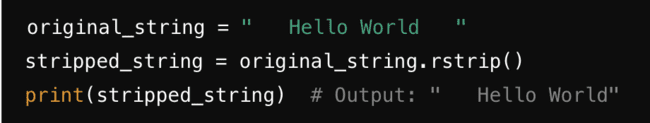
Ex:

****

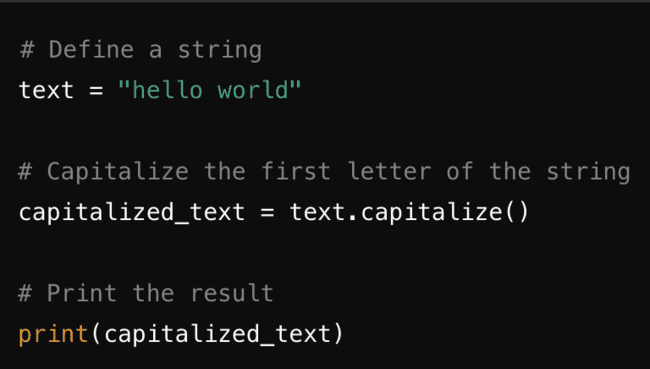
**5.8 lstrip():** remove space from left side

****

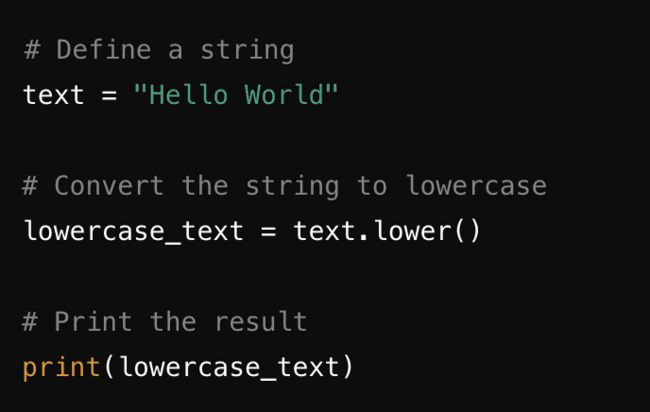
**5.9 rstrip():** remove space from right side

****

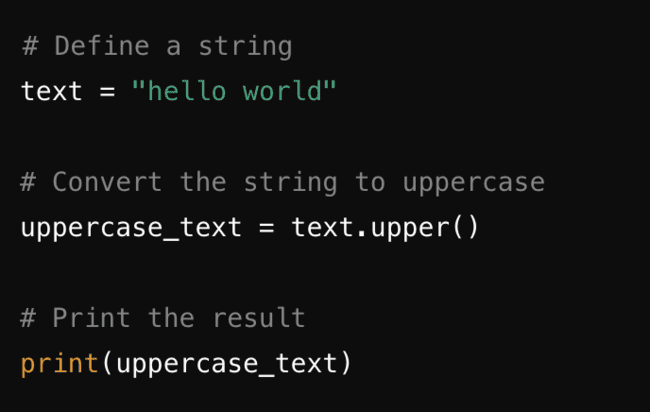
**5.10 Capitalize():**

****

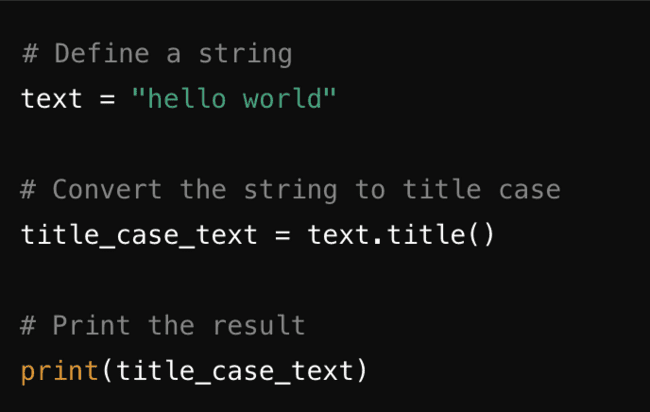
**5.11 Lower():**

****

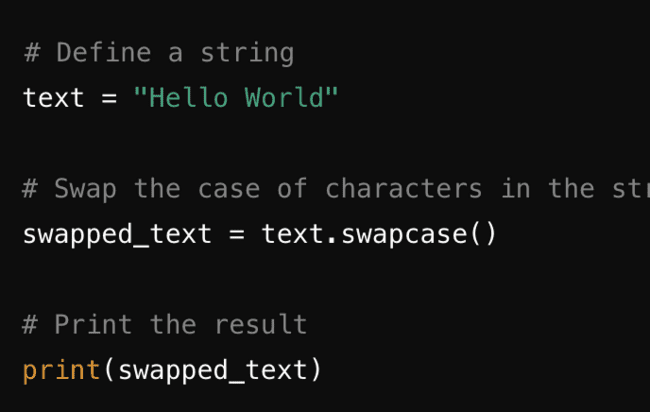
**5.12 Upper():**

****

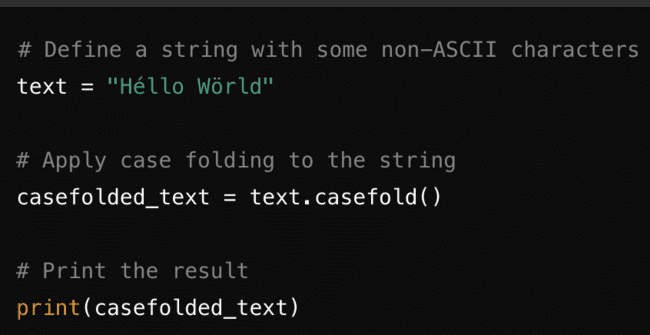
**5.13 title():**

****

**5.14 swapcase():**

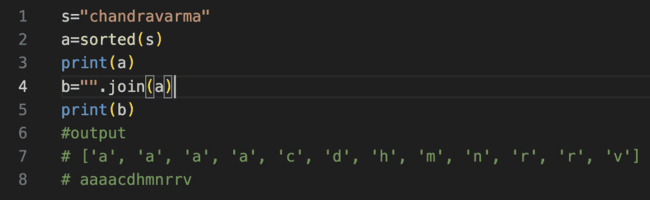
****

**5.15 casefold()**

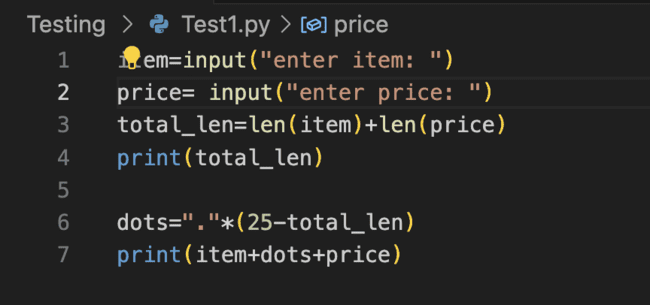
****

**6) Challenges**

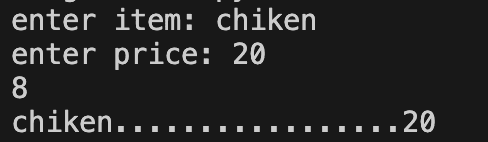
**6.1) Sorting letter of string**

****

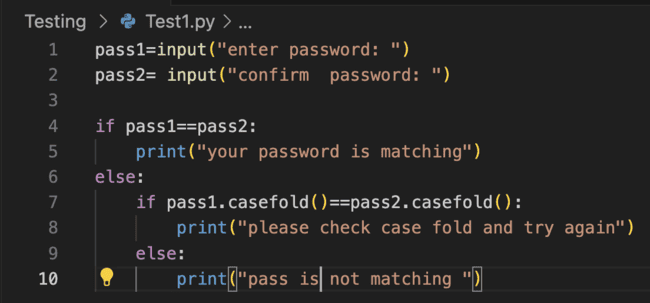
**6.2) Displaying Data in give format**

****

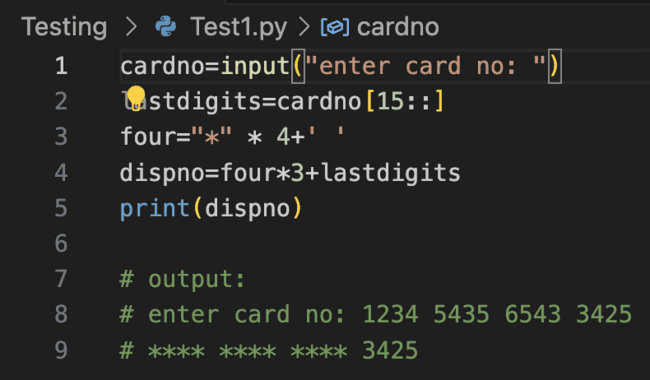
**Output:**

****

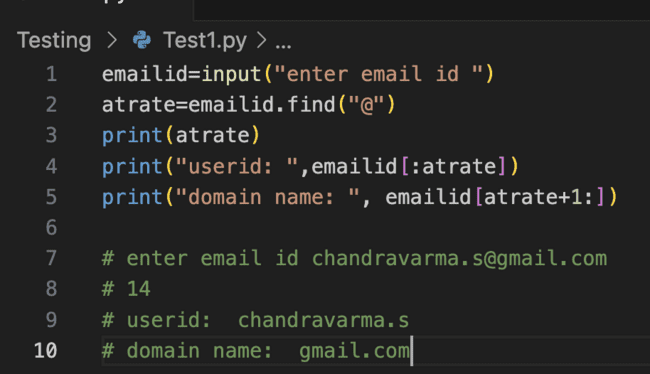
**6.3) Check if the password and confirm password are same**

****

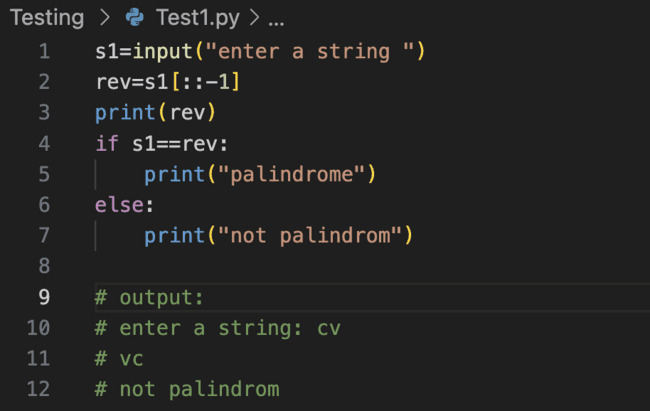
**6.4) Display credit card number**

****

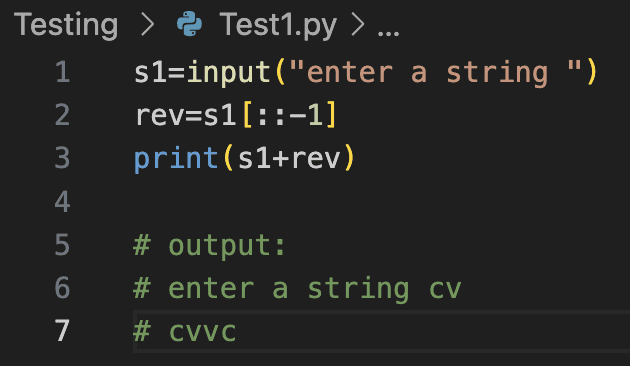
**6.4) find user id and domain name from email id**

****

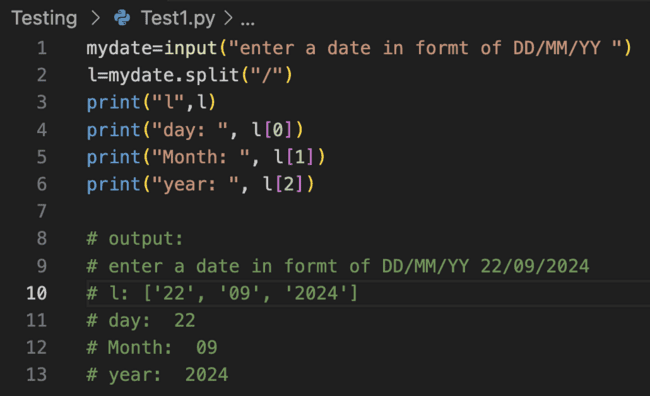
**6.4) checking string is palindrome or not**

****

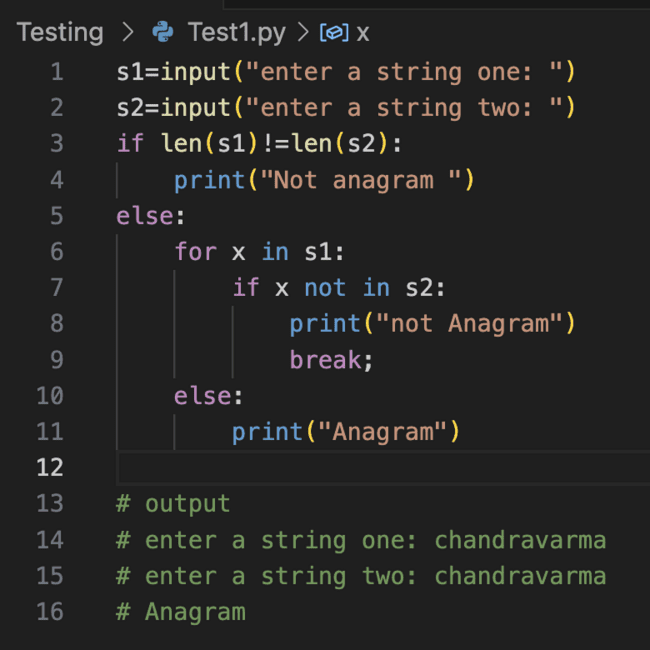
**6.4) convert string to palindrome**

****

**6.4) find date month year from given date**

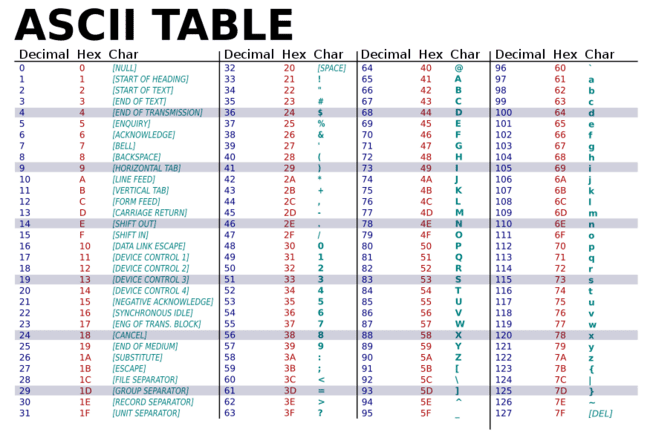
****

**6.4) find string is anagram or not**

****

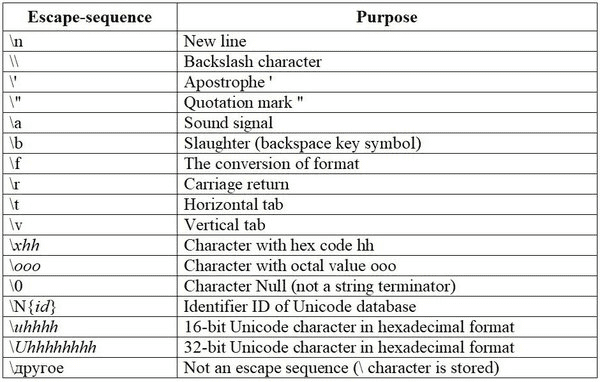
## Formatted Printing

**1) ASCII Code vs Unicode**

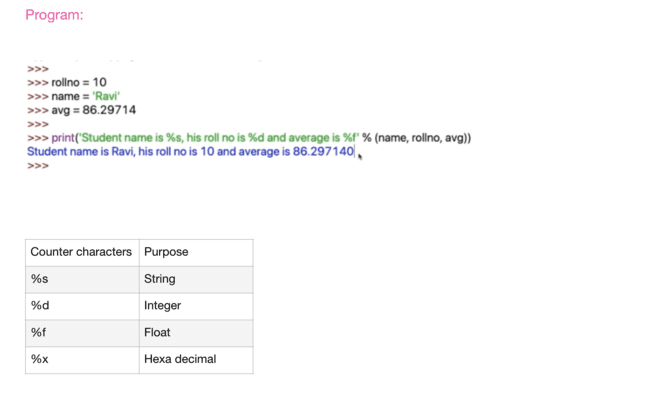
****

**2) print Functions:** use below sequence in print function

**2.1 Escape Sequence**

****

**3) C-Style Printing**

****