

Python

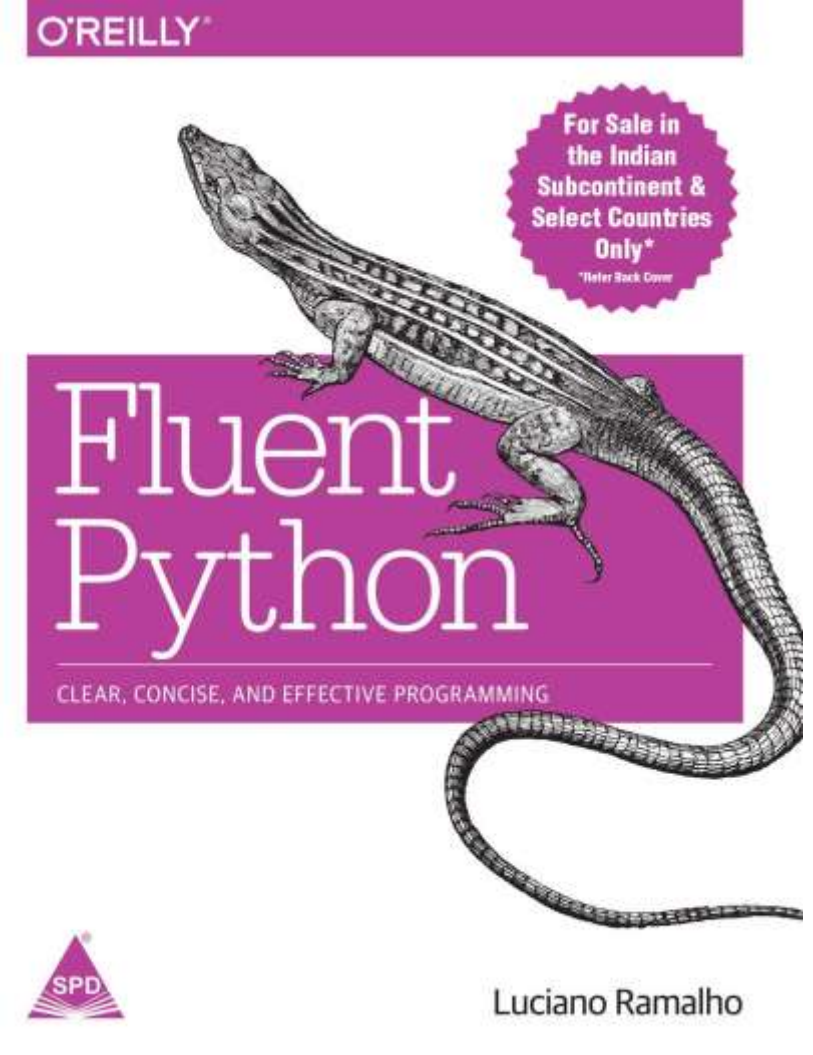
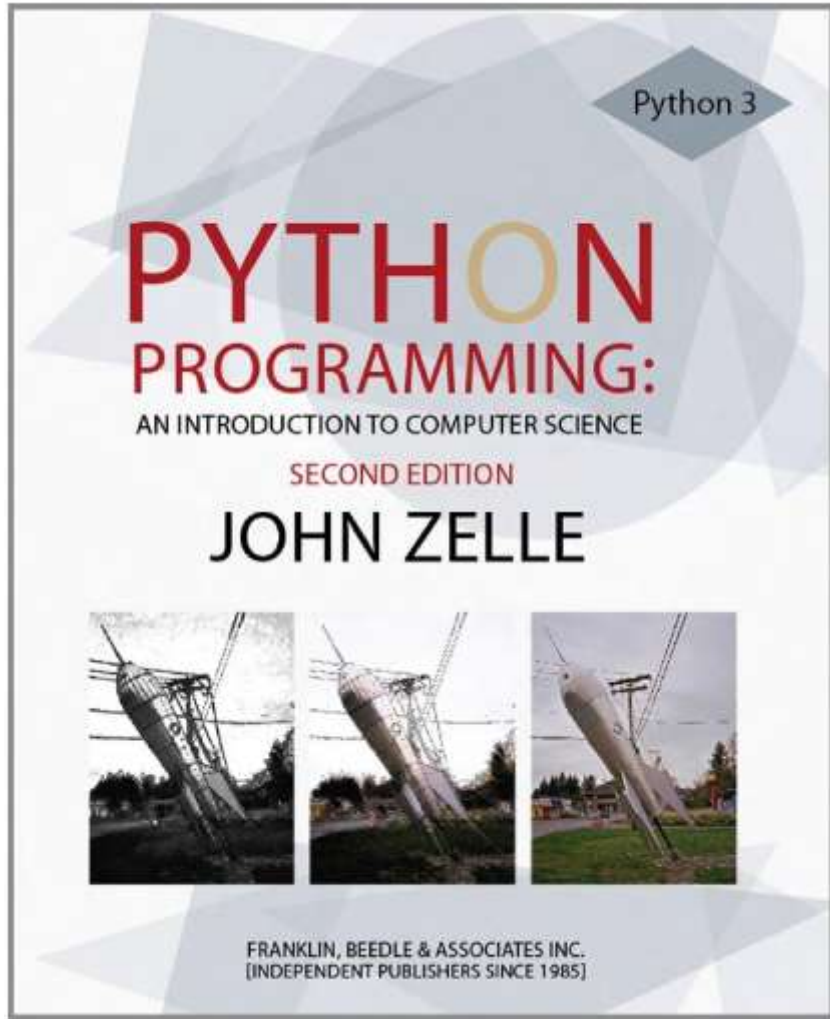


by ...

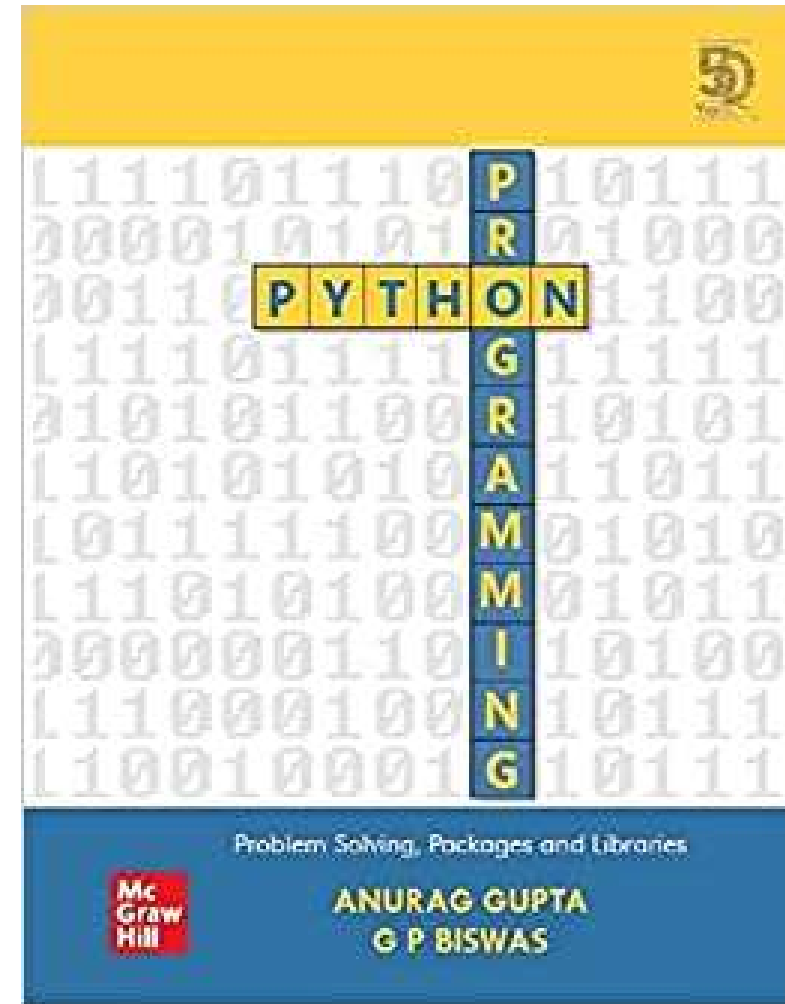
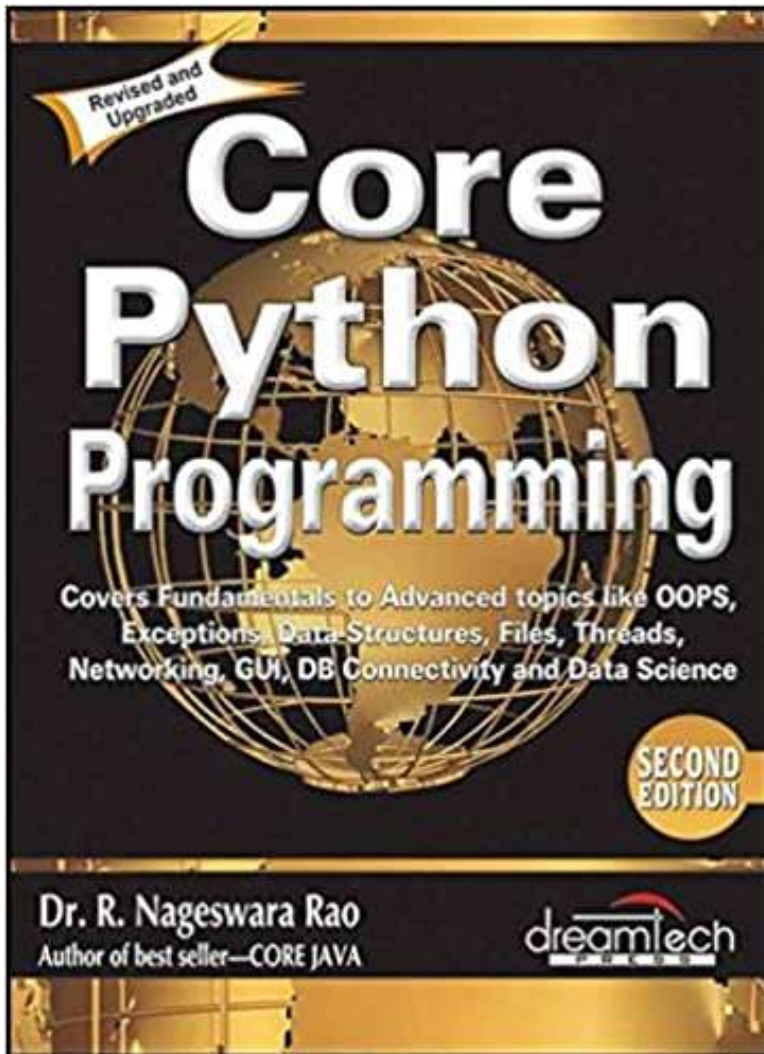
Dr. Pradyumna Kumar Tripathy

*Associate Professor & Head, Dept. of CSE,
Silicon Institute of Technology, Bhubaneswar*

Reference Books



I Recommend:



Introduction

- High Level Programming Language.
- Developed by Guido Van Rossum in Feb 1989 while working at Centrum Wiskunde & Informatics (CWI). [released as **Open Source** in February 1991]
- Based on/ Influenced by two programming Languages:
 - **ABC Language**: A teaching Language created as a replacement of BASIC
 - **Modula-3**
- Easy-to-learn, powerful, **object oriented** programming Language



something people can modify and share because its design is publicly accessible

Python is named : Idea is from a British Comedy “**Monty Python’s flying circus**”, not the snake.



Companies Using Python

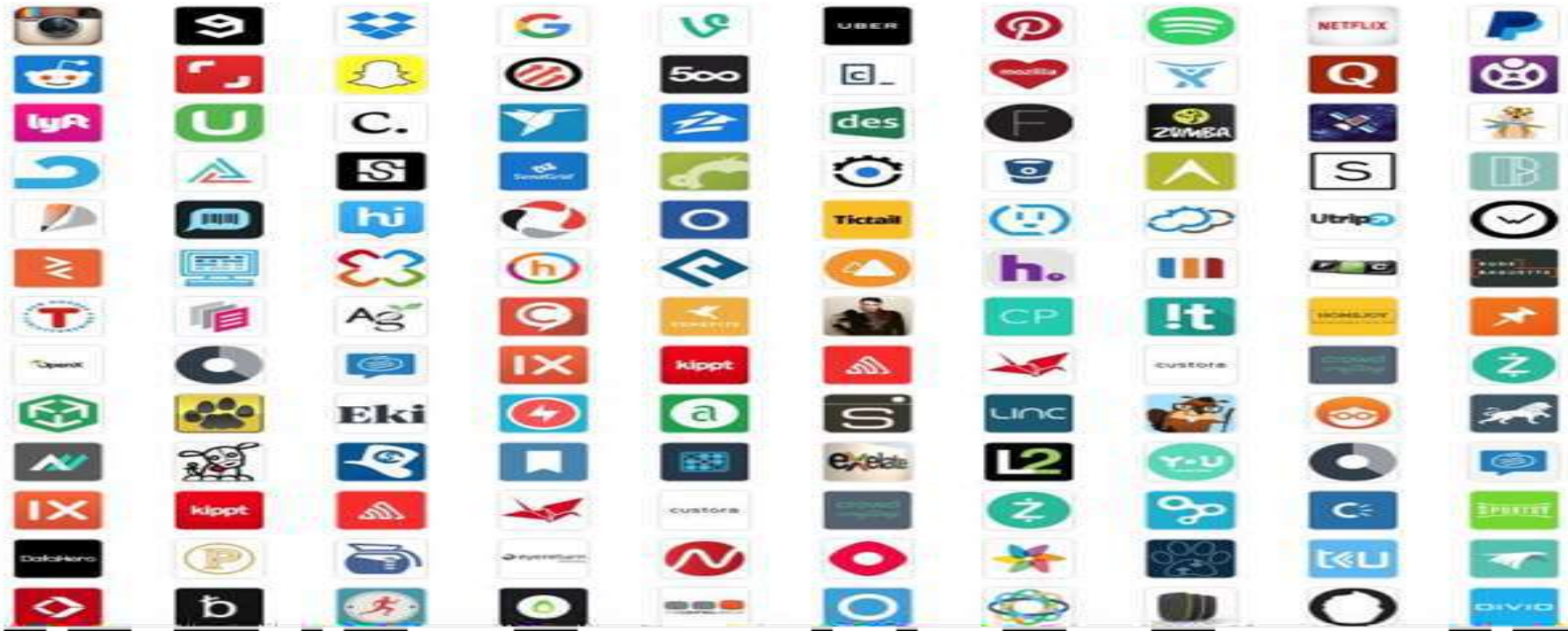
Who uses Python



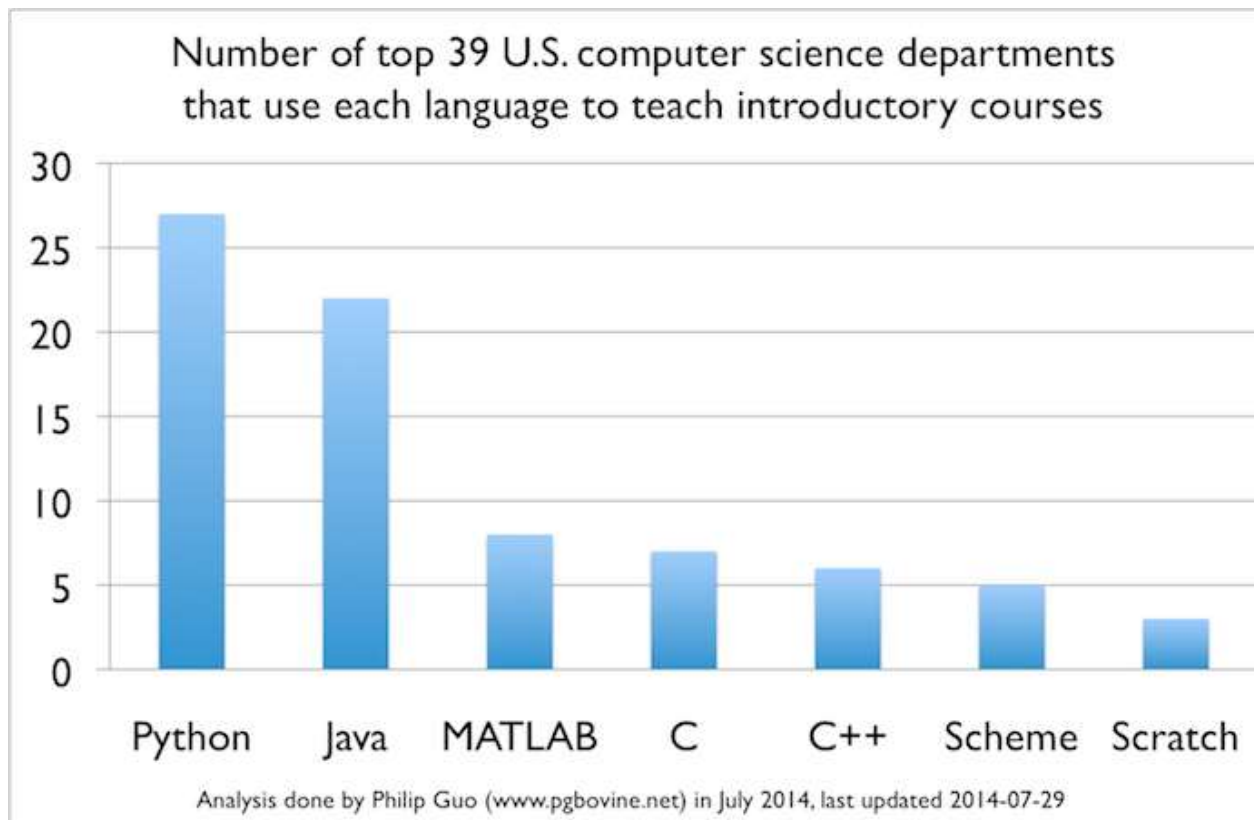
COMPANIES USING PYTHON



COMPANIES USING PYTHON



Why Python ? ? ?



The chart above shows how many of the top 39 departments teach either CS0 or CS1 using the seven most common languages. The bar heights add up to more than 39 since many schools offer both CS0 and CS1.

Different Versions of Python by Date

- **Python 1.0 - January 1994**
 - Python 1.5 - December 31, 1997
 - Python 1.6 - September 5, 2000
- **Python 2.0 - October 16, 2000**
 - Python 2.1 - April 17, 2001
 - Python 2.2 - December 21, 2001
 - Python 2.3 - July 29, 2003
 - Python 2.4 - November 30, 2004
 - Python 2.5 - September 19, 2006
 - Python 2.6 - October 1, 2008
 - Python 2.7 - July 3, 2010
- **Python 3.0 - December 3, 2008**
 - Python 3.1 - June 27, 2009
 - Python 3.2 - February 20, 2011
 - Python 3.3 - September 29, 2012
 - Python 3.4 - March 16, 2014
 - Python 3.5 - September 13, 2015
 - **Python 3.6 - December 23, 2016**

- Expressive Language

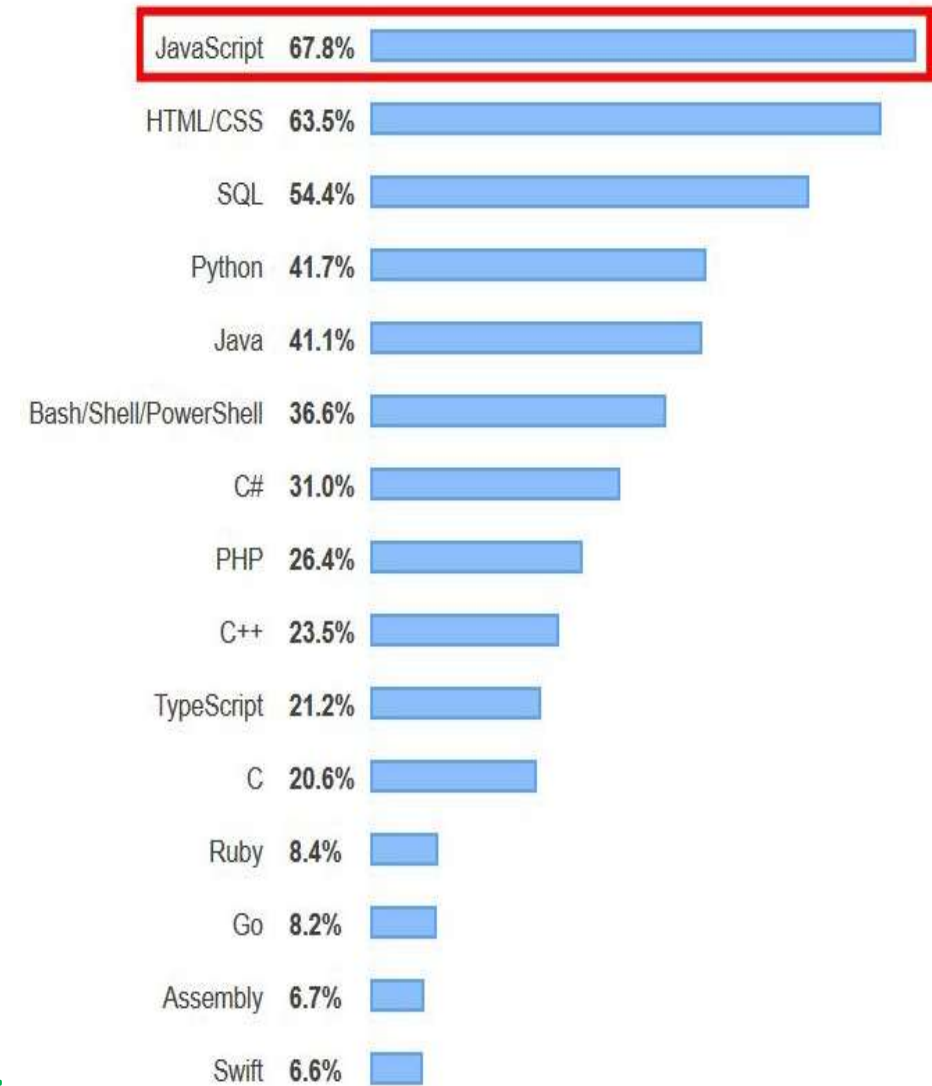
In C++	In Python
<pre>int a=2, b=3, temp; temp=a; a=b; b=temp;</pre>	<pre>a, b = 2, 3 a, b = b, a</pre>

- Interpreted Language
- Completeness (**Batteries included**)
 - Emails, web-pages, databases, GUI, development, network and many more...
- Cross-Platform Language (Portable)
- Free and Open Source
- Variety of Usage/ Applications

Limitations

- **Not the fastest Language**
- Lesser Libraries than C, Java, Perl
- Not Strong on Type-binding
(Declare int and later store a string-No worry)
- **Not Easily Convertible**
(Not Structured. So, Translating into another programming language-difficult)

It is the 4th most popular programming language after Java Script, HTML/CSS and SQL



Survey: 2020

Source: Internet

Installation

- Pre-installed in most of the version of Linux OS
- Download from
 - www.python.org/download (select the python distribution)
- Two Mode of Operation
 - Interactive mode (>>> prompt / python shell)
 - Script Mode (File extension **.py**)

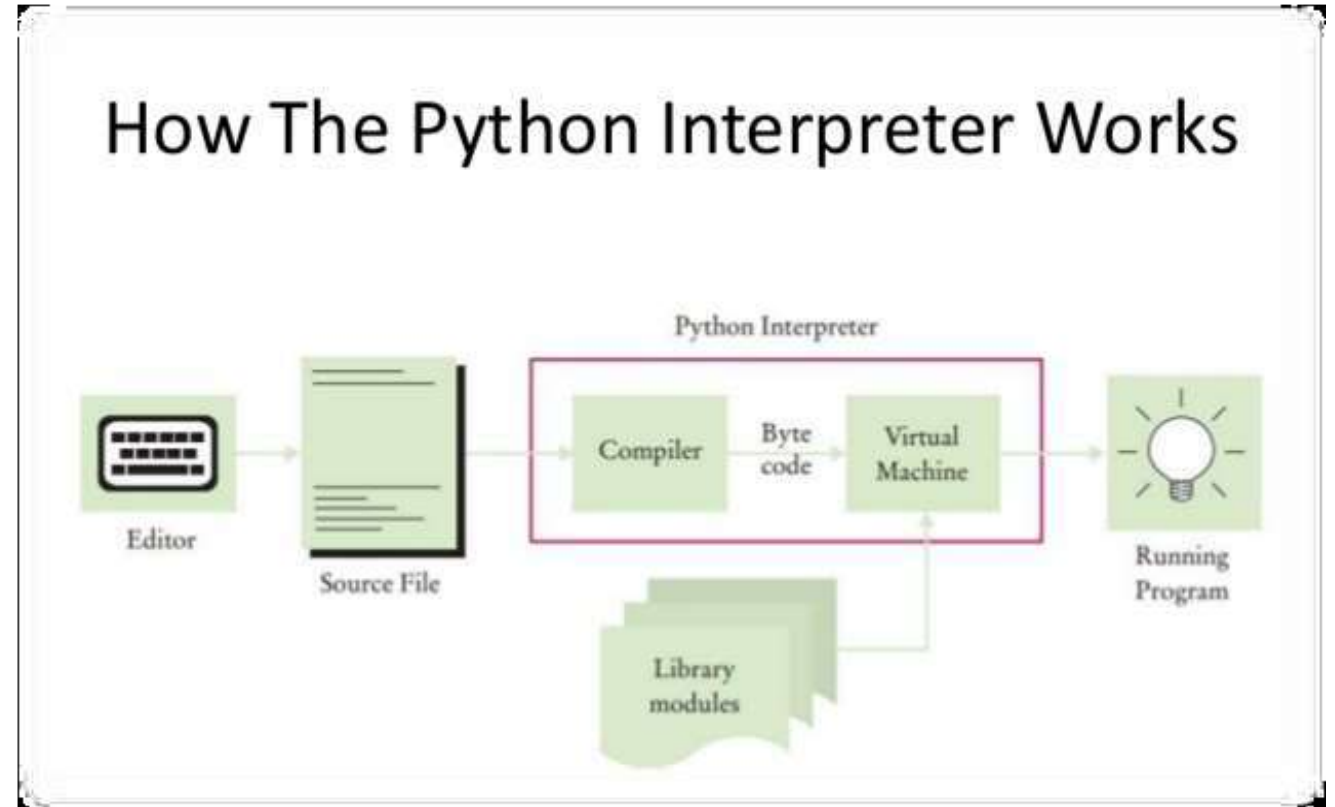
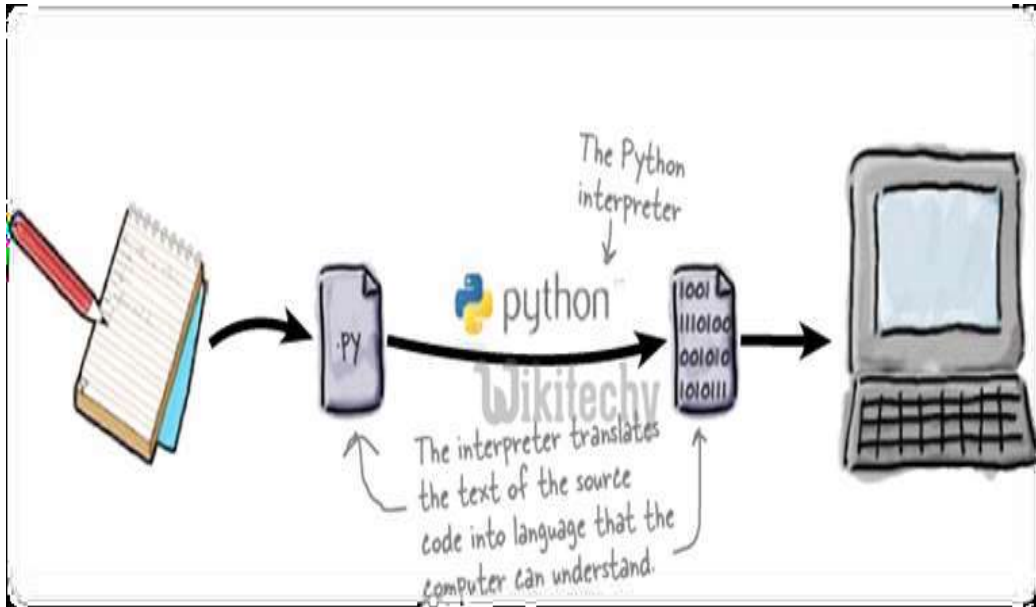
Scripts are reusable.
A script is a text file
containing the
statements that
comprise
a Python program

A **shell** is usually an "interactive
shell",
usually termed a REPL which
stands for
"Read - Execute - Print - Loop"

Execution of a Python Program

- Compiler converts Python program into **Byte Code**
 - **Byte Instructions are system independent or platform independent.**
 - **Size of each byte code instruction is 1 byte (Hence called ByteCode)**
 - **The Byte Code instructions are contained in the file “filename.pyc” (compiled file)**
 - **PVM (Python Virtual Machine) is used to convert Byte code to Machine code**
 - **PVM uses an Interpreter**
- *PVM understands the processor & OS in our computer and converts into the desired machine code format required for computer. Machine code instructions are then executed by the processor**

Python uses Interpreter



Hands-on

Type the following on Interactive window and analyze result

```
>>>3 * 5
```

```
>>>print (3 * 5)
```

```
>>>print (" hello")
```

```
>>>a=20
```

```
>>>print ("my age is " , a)
```



GUESS THE OUTPUT
???

```
>>>3 ** 3
```

```
>>>6 + 2 * 4
```

```
>>>(6 + 2) * 4
```

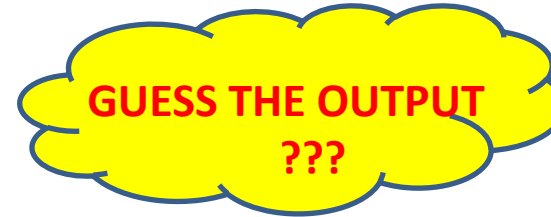
```
>>>5 - 3 - 3
```

```
>>>k = 5 - (3 - 3)
```


Hands-on Contd...

- Open **idle editor**
- Create a program file (name it as **myfile.py**)
- Type the following :

```
print ('hello Ashok', 'hello Pooja')  
Var1 = 'hello Ashok'  
Var2 = 'hello Pooja'  
Name1 = 'Ashok'  
Name2 = 'Pooja'  
print (Var1, Var2)  
print ('hello', Name1, ',', Name2)  
Name1 = 'Bikram'  
Name2 = 'Annu'  
print ('hello', Name1, ',', Name2)
```



(save with **.py** and run it with **python filename** command)

Example

Type

>>>type(4) <<int>>

Value

>>>a Content of a

Id

>>>id(4) Memory Address of object 4

Hands-on Contd...

```
>>> a = 7
```

```
>>> type(a)
```

```
>>> type(99.9)
```

```
>>> type('ABC')
```

```
>>> 5
```

```
>>> 015
```

```
>>> 9 / 5
```

```
>>> 9 // 5
```

```
>>> a //= 3
```

```
>>> 5 / 0
```

```
>>> a *= 2
```

```
>>> 9 % 5
```

```
>>> divmod(9,5)
```



GUESS THE OUTPUT
???

Hands-on Contd...

```
>>>int(True)
```

```
>>>int(False)
```

```
>>>int(98.6)
```

```
>>>int(1.0e4)
```

```
>>>int('99')
```

```
>>>int('-45')
```

```
>>>int('+12')
```

```
>>>True + 3
```

```
>>>False + 5.0
```

```
>>>googol=10 ** 100
```

```
>>>googol * googol
```



GUESS THE OUTPUT
???

(called googol)

Python Character Set

- **Letters:** A-Z, a-z
- **Digits :** 0-9
- **Special Symbols:** space + - * / ** \ () [] // = != == <> . ' " "" , ; : % ! & # <= >= @ >>> << >> _
- **Whitespaces :** Blank space, tab, carriage return, newline, form-feed
- **Other Characters:** Python can process all ASCII and unicode characters

Tokens

The smallest individual units of a program is called token or lexical unit.

- Keywords
- Identifiers
- Literals
- Operators
- Punctuators

Keywords

```
>>>import keyword  
>>>print(keyword.kwlist)
```

```
False      def        if          raise  
None       del        import      return  
True       elif       in          try  
and        else       is          while  
as         except    lambda     with  
assert     finally   nonlocal   yield  
break      for  
class      from  
continue   global    pass
```

```
>>> keyword.iskeyword(s)
```

Identifier

- Arbitrary long sequence of letters and digits
- First character is a letter (_ is counts as a letter)
- All Characters are significant
- The digits can be a part of identifier except the first character
- Must not be a python keyword
- No special character except _

Python is Case Sensitive

Literals

- String ('Silicon', "Silicon", "Silicon", ""Silicon"")
- Numeric
- Boolean
- Special (None)
- Literal collection

Escape Sequences

Escape Sequence	Meaning
<code>\newline</code>	Ignored
<code>\\</code>	Backslash (<code>\</code>)
<code>\'</code>	Single quote (<code>'</code>)
<code>\"</code>	Double quote (<code>"</code>)
<code>\a</code>	ASCII Bell (BEL)
<code>\b</code>	ASCII Backspace (BS)
<code>\f</code>	ASCII Formfeed (FF)
<code>\n</code>	ASCII Linefeed (LF)
<code>\N{name}</code>	Character named <i>name</i> in the Unicode database (Unicode only)
<code>\r</code>	ASCII Carriage Return (CR)
<code>\t</code>	ASCII Horizontal Tab (TAB)
<code>\uxxxx</code>	Character with 16-bit hex value <i>xxxx</i> (Unicode only)
<code>\Uxxxxxxxx</code>	Character with 32-bit hex value <i>xxxxxxxx</i> (Unicode only)
<code>\v</code>	ASCII Vertical Tab (VT)
<code>\ooo</code>	Character with octal value <i>ooo</i>
<code>\xhh</code>	Character with hex value <i>hh</i>

String Types

- Single line Strings
- Multi line Strings

a) Single line Strings

Enclosed in ' ' or " "

```
>>>text1= ' hello
```

```
there ' ERROR
```

b) Multi line Strings

```
>>>text1 = ' hello\
```

```
there ' OK
```

OR

```
>>>text1= ''' hello
```

```
there ''' OK
```

```
>>>text1= " " "hello
```

```
there " " " OK Try with print(text1)
```

Size of a String

```
>>>len(' \\ ' )      1
>>>len('abc')        3
>>>len("\ab")        2
>>>len("Seema\'s pen") 11
>>>len("Amys")        4

>>>str3 = ' ' ' a      5
b
c ' ' '

>>>str4 = ' a\         3
b\
c '
```

Hands-on Contd...

>>> ' '

>>> " "

>>> | | | | |

>>> " " " " " "

```
>>> bottle = 99
```

```
>>>base = ''
```

```
>>>base += 'current inventory: '
```

```
>>> base += str(bottle)
```

```
>>> base
```

GUESS THE OUTPUT

???

Hands-on Contd...

```
>>>start = ' Na ' * 4 + '\n'  
>>>middle = ' Hey ' * 3 + '\n'  
>>>end = ' Goodbye. '  
>>>print ( start + middle + end)
```



GUESS THE OUTPUT
???

```
>>>letters = ' abcdefghijklmnopqrstuvwxyz '  
>>>letters[0]  
>>>letters[1]  
>>>letters[-1]  
>>>letters[-2]  
>>>letters[100]
```

```
>>>name = ' Henny '  
>>>name[0] = ' P '      ERROR  
>>>name.replace(' H ', ' P ' )  
>>> ' P ' + name[1:]
```

Hands-on Contd...

[start: end: step]

```
>>>letters = 'abcdefghijklmnopqrstuvwxyz '
```

```
>>>letters[:]
```

```
>>>letters[20:]
```

```
>>>letters[12:15]
```

```
>>>letters[-3:]
```

```
>>>letters[18:-3]
```

```
>>>letters[-6:-2]
```

```
>>>letters[::7]
```

```
>>>letters[4:20:3]
```

```
>>>letters[19::4]
```

```
>>>letters[:21:5]
```

```
>>>letters[-1::-1]
```

```
>>>letters[-51:-50]
```



GUESS THE OUTPUT
???

Hands-on Contd...

```
>>>t = 'get gloves, get mask, give cat vitamins, call ambulance'
```

```
>>>t.split(',')
```

```
>>>t.split( )
```

-Store a paragraph of text in a variable **t**

-extract first 13 characters **(t[:13])**

-find total no of characters **(len(t))**

-does it starts with letters 'All' **(t.startswith('All'))**

-does it ends with letters 'Bye' **(t.endswith('Bye'))**

-find the offset of the first occurrence of a word **(t.find('word'))**

-find the offset of the last occurrence of a word **(t.rfind('word'))**

-How many times the word 'the' occurs in the text **(t.count('the'))**

Hands-on Contd...

Try out the following function on a string variable:

<code>strip()</code>	<code>#t.strip()</code> Removes leading and trailing whitespace characters
<code>capitalize()</code>	<code>#t.capitalize()</code> Convert First Character in Upper case
<code>title()</code>	<code>#t.title()</code> Convert First Character of each word in Upper case
<code>upper()</code>	<code>#t.upper()</code> Convert All Characters in Upper case
<code>lower()</code>	<code>#t.lower()</code> Convert All Characters in Lower case
<code>swapcase()</code>	<code>#t.swapcase()</code> Swap case of each Characters
<code>center(n)</code>	<code>#t.center(30)</code> Place the text 't' at the center of 30 characters
<code>ljust()</code>	<code>#t.ljust(30)</code> Place the text 't' left justified among 30 characters
<code>rjust()</code>	<code>#t.rjust(30)</code> Place the text 't' right justified among 30 characters

**GUESS THE OUTPUT
???**

Numeric Literals

- **int (signed integers)**

Decimal

Binary

Octal

Hexa

>>>10

>>>0b10

>>>0o10

>>>0x10

- **long (long integers)**

- **float (floating point real values)**

- **Fractional Form** (2.0, -13.7, -0.0076)

- **Exponent Form** (172E05, 1.7E07, 0.152E08, -0.007E-3)

- **Complex (complex numbers)**

Boolean Literal

- True
- False

Special Literal

- None

None Keyword

>>> **None == 0** **False**

>>> **None == []** **False**

>>> **None == False** **False**

>>> **x = None**

>>> **y = None**

>>> **x == y** **True**

Operators

- **Unary Operators**

- +

- -

- ~ (bitwise complement)

- not (logical negation)

- **Arithmetic Operators** + - * / % ** //

- **Bitwise Operators** & ^ | << >> is is not

- **Relational Operators** < > <= >= == != <>

Operators Contd...

- Assignment Operators = /= += *= %= -= **= // =
- Logical Operators and or
- Membership Operators in not in

Punctuators

(Symbols used to organize sentence structure and indicate the rhythm and emphasis of expressions, statements, and program structure)

‘ “ # \ () [] { } @ , : . ` = ;

Random

```
import random
```

```
print random.random()      (0.0----1.0)
```

```
print random.randint(15,35)
```

```
print random.randint(3,10)-3    (0-7)
```


Barebones of a Python Program

#This program shows a program's components

#Definition of seeyou() follows

def seeyou():

print ('Time to say good bye !!')

Function Definition

#main program code follows now

a = 15

b = a – 10

print (a+3)

if b > 5:

print ('value of 'a' is more than 5')

else:

print ('value of 'a' is less than 5')

seeyou() #Call of Function

**GUESS THE OUTPUT
???**

Input - Output

```
Name = raw_input('what is your name')
```

```
Age = raw_input('what is your age')
```

-It always returns a value of string type

```
>>>Age+5      ????      ERROR
```

```
Age = int(raw_input('what is your age'))
```

```
Amount = float (raw_input('what is your salary'))
```

Or

```
Age = (raw_input('what is your age'))
```

```
Age = int(Age)
```

raw_input () is used in Python 2.7.X

What happen if you input your age as Sixteen ???

Input - Output

```
>>>Age = input('Enter your age')  
>>>Retire_age = input('Enter age:')  
>>>Enter age: 40 + 20
```

input () is used in Python 3.7.X onwards



GUESS THE OUTPUT
???

-input is not a stable function on all installation

Print statement without any argument prints a blank line

Sample Program

Problem: Compute Area of a Circle

Solution:

```
pi = 3.14159
```

```
radius = 2.2
```

```
# area of circle equation <- this is a comment
```

```
area = pi*(radius**2)
```

```
print(area)
```

```
# change values of radius
```

```
# use comments to help others understand what you are doing in code
```

```
radius = radius + 1
```

```
print(area) # area doesn't change
```

```
area = pi*(radius**2)
```

```
print(area)
```

Sample Program

Problem: Input a filename and print the extension of that

Solution:

```
filename = input('Input the Filename: ' )  
file_extns = filename.split('.')  
print ('The extension of the file is :' + (file_extns[-1]))
```

Problem: Input an integer (n) and computes the value of $n + nn + nnn$

Solution:

```
n = int(input('Input an integer :'))  
nn = n*10+n  
nnn = nn*10+n  
print (n+nn+nnn)
```

Sample Program

Problem: Write a Python program to calculate number of days between two dates.

Sample dates : (2014, 7, 2), (2014, 7, 11)

Expected output : 9 days

Solution:

```
from datetime import date
f_date = date(2014, 7, 2)
l_date = date(2014, 7, 11)
delta = l_date - f_date
print(delta.days)
```

Problem: Print the calendar of a given month and year

Solution:

```
import calendar
y = int(input('Input the year :'))
m = int(input('Input the month :'))
print(calendar.month(y, m))
```

Sample Program

Problem: Get the Python version and Display the current date and time

Solution:

```
import sys
import datetime
print('Python version«', sys.version)
print('Version info.«', sys.version_info)
print ('Current date and time :',datetime.datetime.now())
```


Sample Program

Problem: Compute the distance between two points

Solution:

```
import math  
p1 = [4, 0]  
p2 = [6, 6]  
distance = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )  
print(distance)
```

Sample Program

Problem: Sort three inputted numbers in ascending order without using any conditional or looping statements.

Solution:

```
x = int(input("Input first number: "))
y = int(input("Input second number: "))
z = int(input("Input third number: "))
a1 = min(x, y, z)
a3 = max(x, y, z)
a2 = (x + y + z) - a1 - a3
print("Numbers in sorted order: ", a1, a2, a3)
```

Sample Program

Problem:

Solution:

```
x = 'true'
```

```
x = int(x == 'true')
```

```
print(x)
```

```
x = 'abcd'
```

```
x = int(x == 'true')
```

```
print(x)
```

Assignment

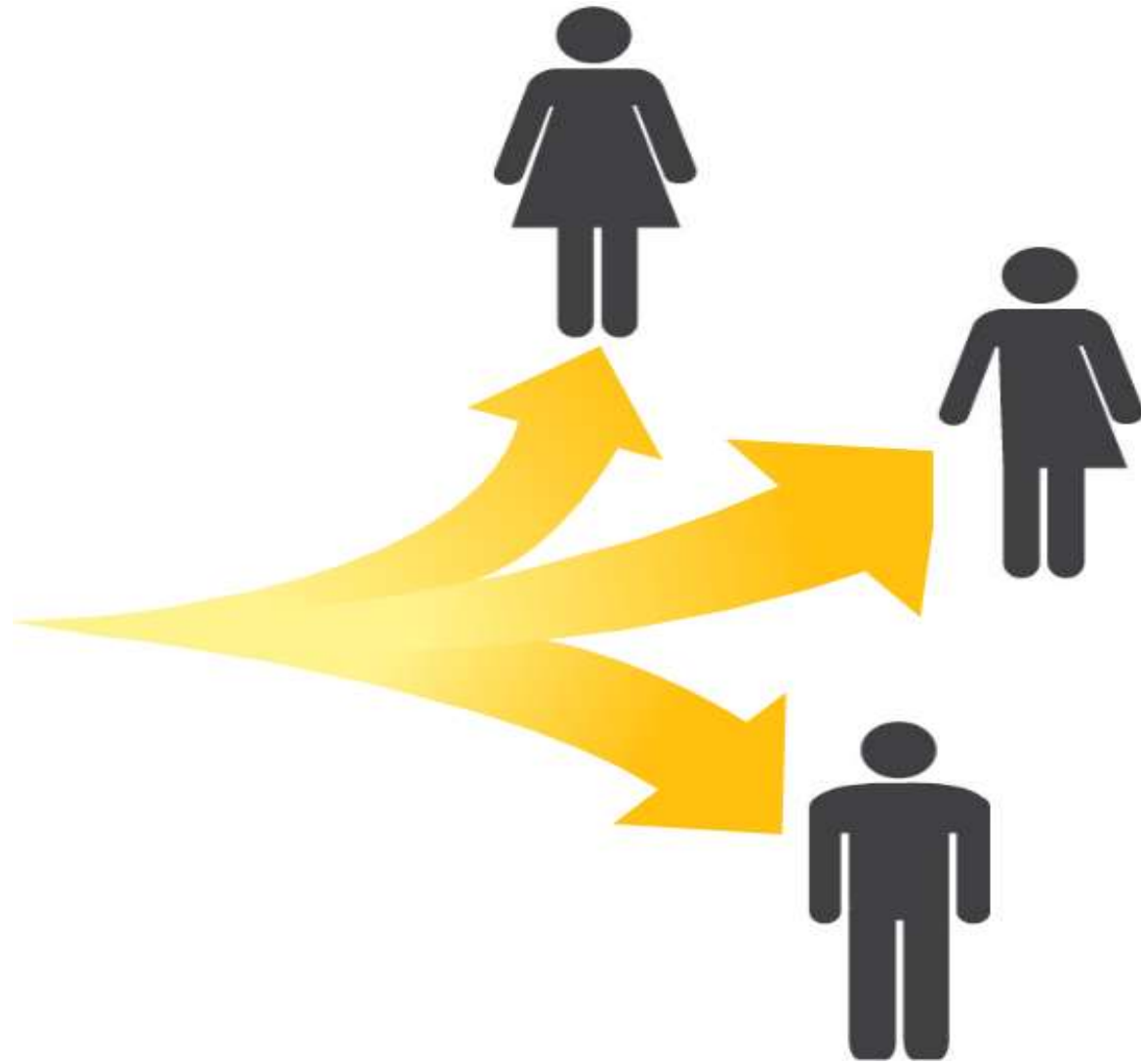
- Write a python program to input two numbers and do all basic arithmetic operations on them
- Write a python program to input two number and swap their values without using any third variable
- Write a python program to input the temperature in Fahrenheit and change it to Celsius.
- Write a python program to input the basic salary of a person and compute its TA (20% of basic), DA (120% of basic), HRA (30% of basic), Gross (basic + ta + da + hra).

Assignment

- Write a python program to input the radius of a circle and print its area and perimeter
- Write a python program to input marks in 5 subjects of a student and print its average mark
- Write a python program to input a number and print its square, cube and fourth power.
- Write a python program to input a the sides of a triangle and print its area
- Write a python program to compute SI and CI
- Write a python program to input a number and print its first 5 multiples

Assignment

- Write a python program to swap three variables
- Write a python program to input the time in second and print in hr, min and sec
- Write a python program to evaluate the expression
 $4x^4 + 3y^3 - 9z + 6\pi$
- Write a python program to take a input in uppercase and change it to lower case



CONDITIONAL STATEMENTS

Dr. Pradyumna Kumar Tripathy, Silicon Institute of
Technology, Bhubaneswar

If Statement

if test expression:
statement(s)

```
num = 3
if num > 0:
    print(num, "is a positivenumber.")
print("This is always printed.")
```

```
num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

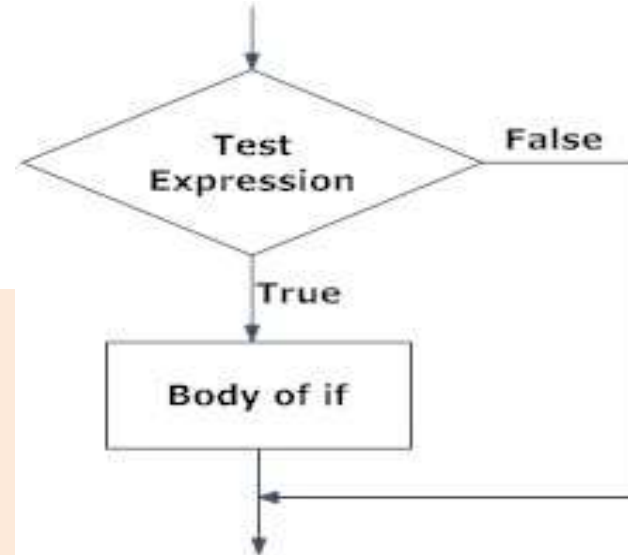
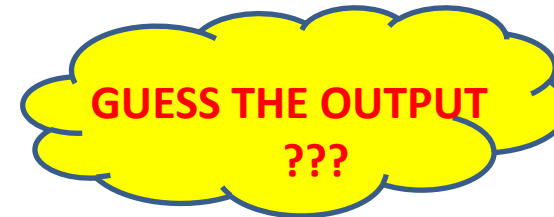


Fig: Operation of if statement



If-else statement

if test expression:

Body of if

else:

Body of else

```
num = 3
if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

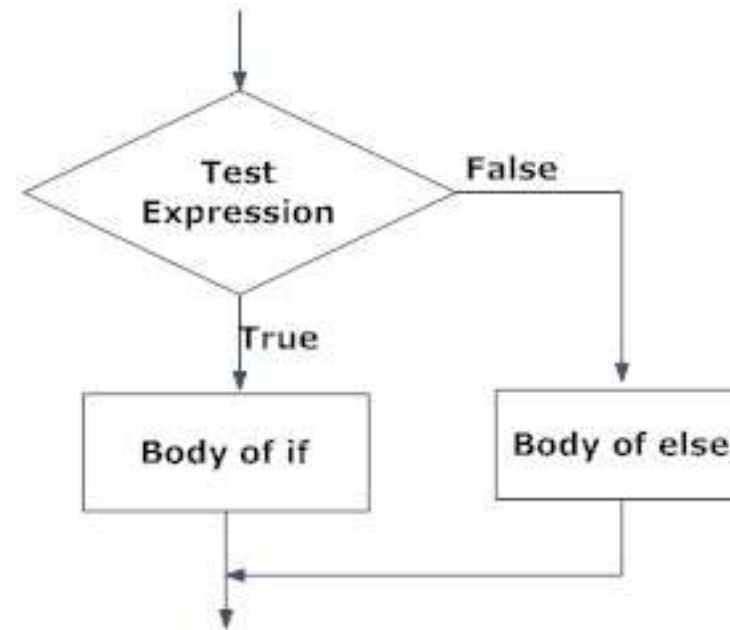


Fig: Operation of if...else statement

**GUESS THE OUTPUT
???**

if...elif...else Statement

if test expression:

Body of if

elif test expression:

Body of elif

else:

Body of else

```
num = 3.4
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
print('Thank you')
```

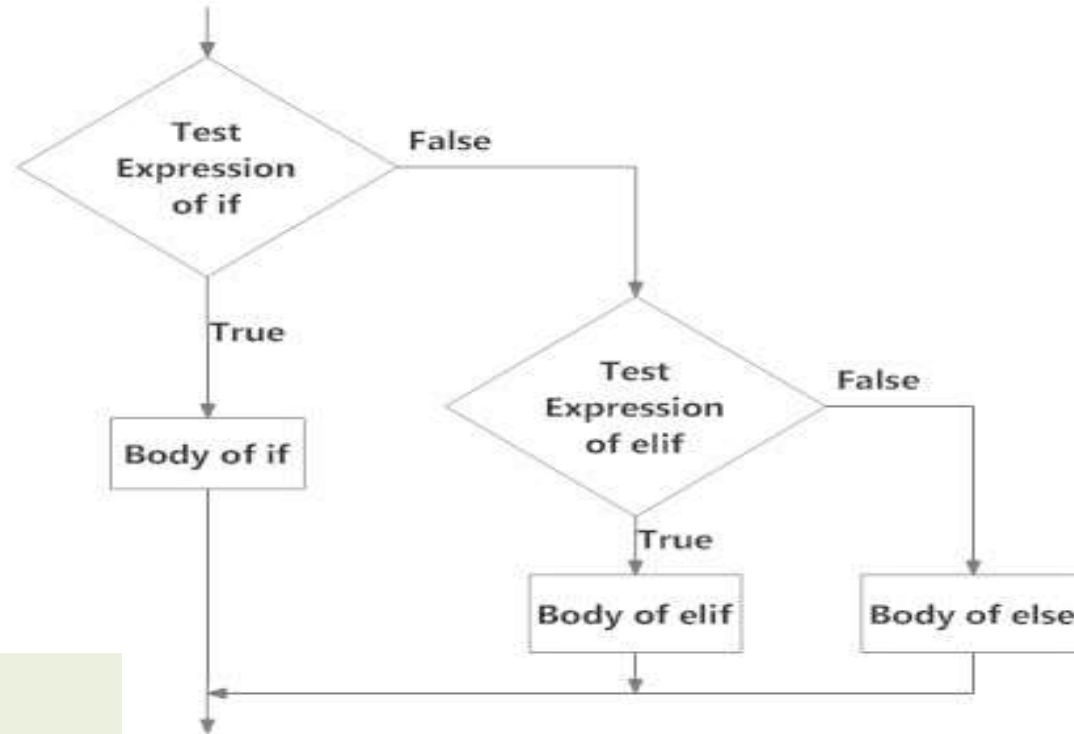


Fig: Operation of if...elif...else statement

**GUESS THE OUTPUT
???**

Nested if else

```
num = float(input("Enter a number: "))  
if num >= 0:  
    if num == 0:  
        print("Zero")  
    else:  
        print("Positive number")  
else:  
    print("Negative number")
```



GUESS THE OUTPUT
???

Comparison Operators

```
x = 5
```

```
if x == 5 :
```

```
    print('Equals 5')
```

Equals 5

```
if x > 4 :
```

```
    print('Greater than 4')
```

Greater than 4

```
if x >= 5 :
```

```
    print('Greater than or Equals 5')
```

Greater than or Equals 5

```
if x < 6 :
```

```
    print('Less than 6')
```

Less than 6

```
if x <= 5 :
```

```
    print('Less than or Equals 5')
```

Less than or Equals 5

```
if x != 6 :
```

```
    print('Not equal 6')
```

Not equal 6

One-Way Decisions

```
x = 5
```

```
print('Before 5')
```

```
if x == 5 :
```

```
    print('Is 5')
```

```
    print('Is Still 5')
```

```
    print('Third 5')
```

```
print('Afterwards 5')
```

```
print('Before 6')
```

```
if x == 6 :
```

```
    print('Is 6')
```

```
    print('Is Still 6')
```

```
    print('Third 6')
```

```
print('Afterwards 6')
```

Before 5

Is 5

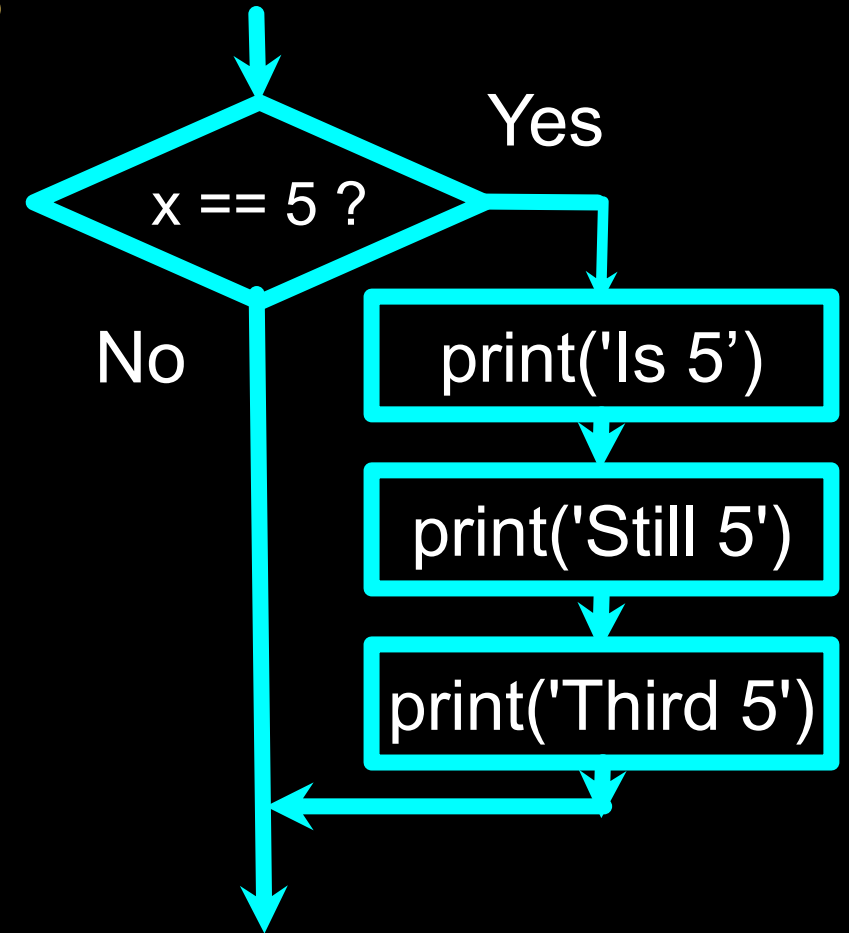
Is Still 5

Third 5

Afterwards 5

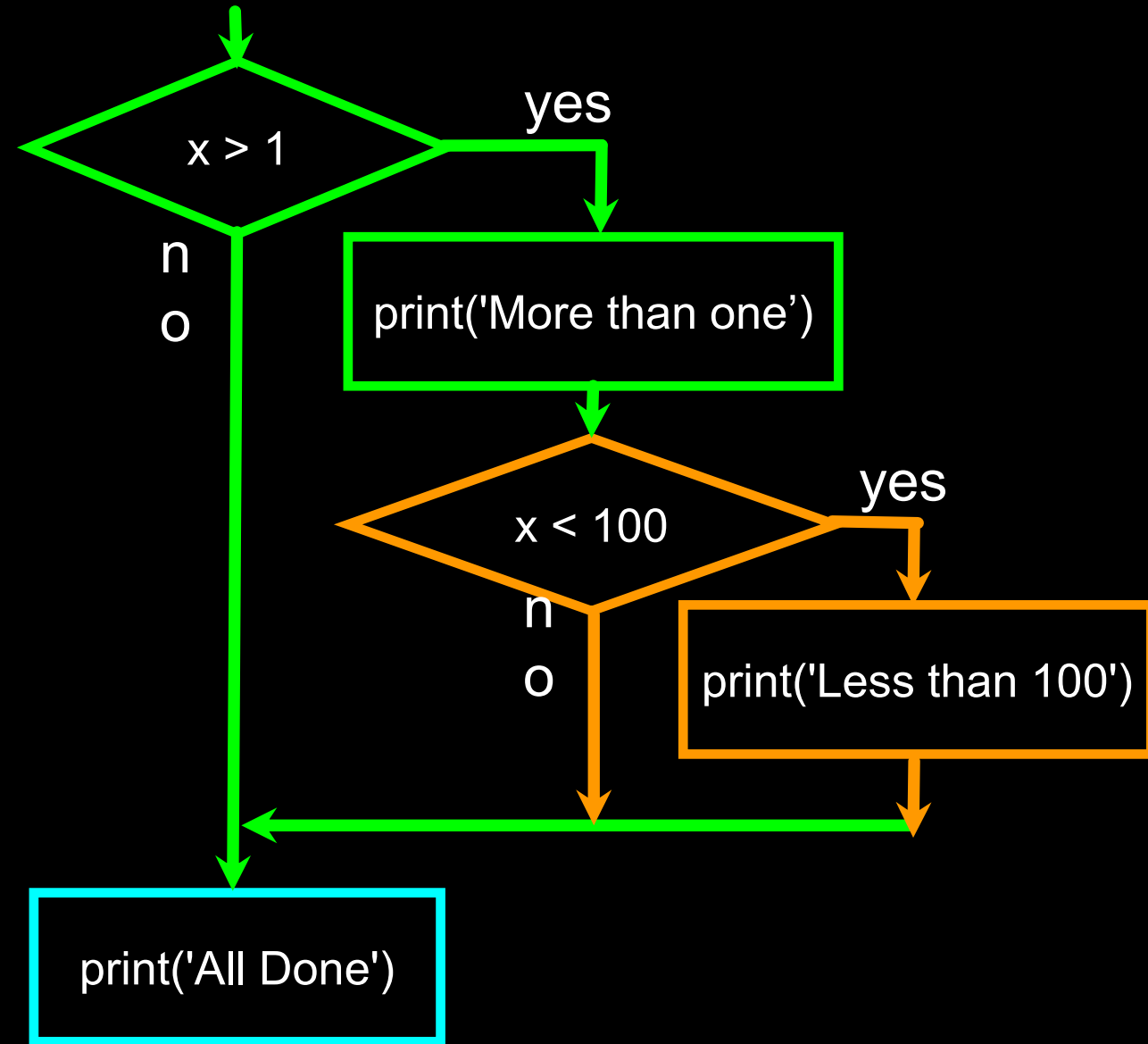
Before 6

Afterwards 6



Nested Decisions

```
x = 42
if x > 1 :
    print('More than one')
    if x < 100 :
        print('Less than 100')
print('All done')
```

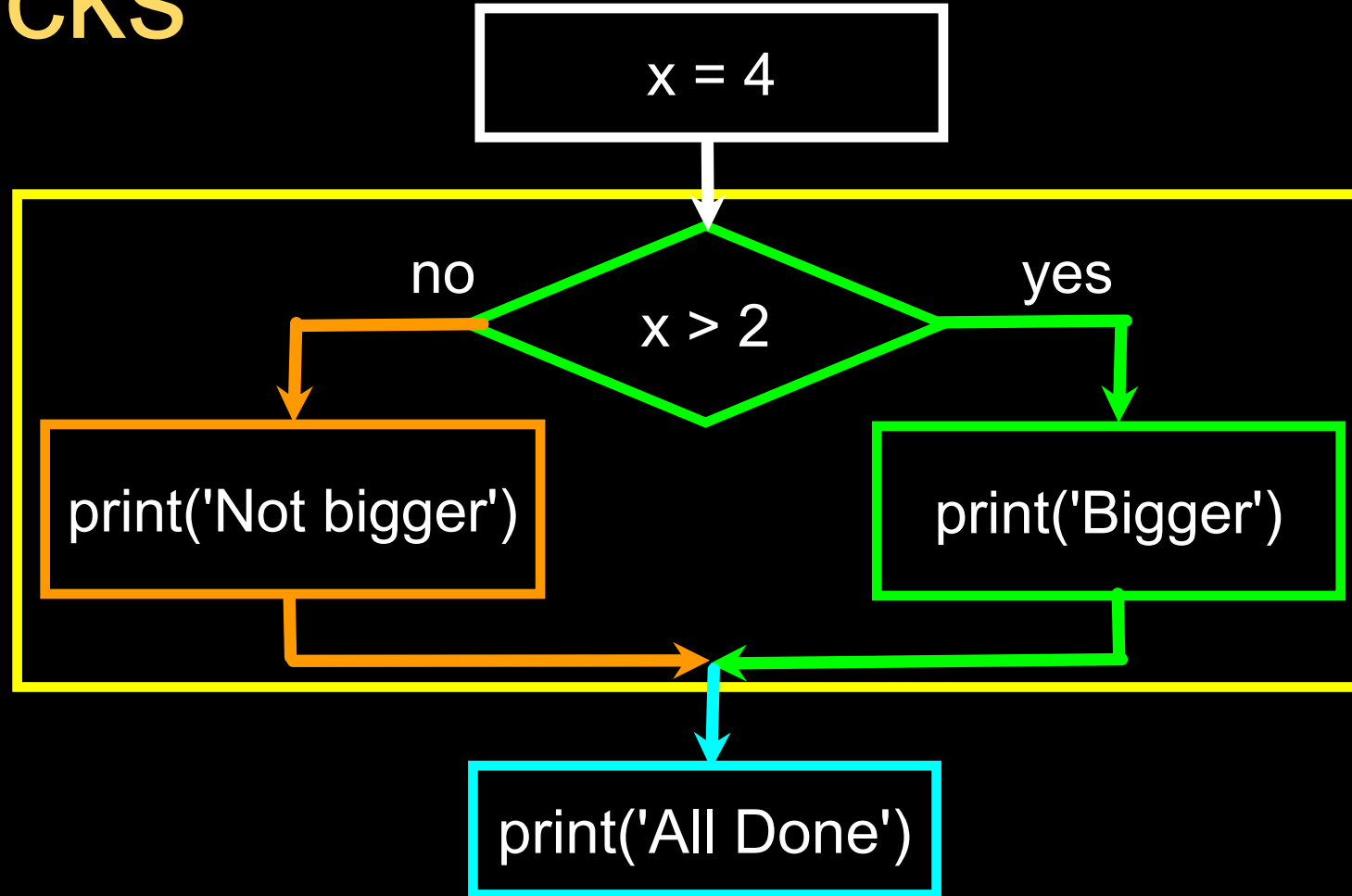


Visualize Blocks

```
x = 4
```

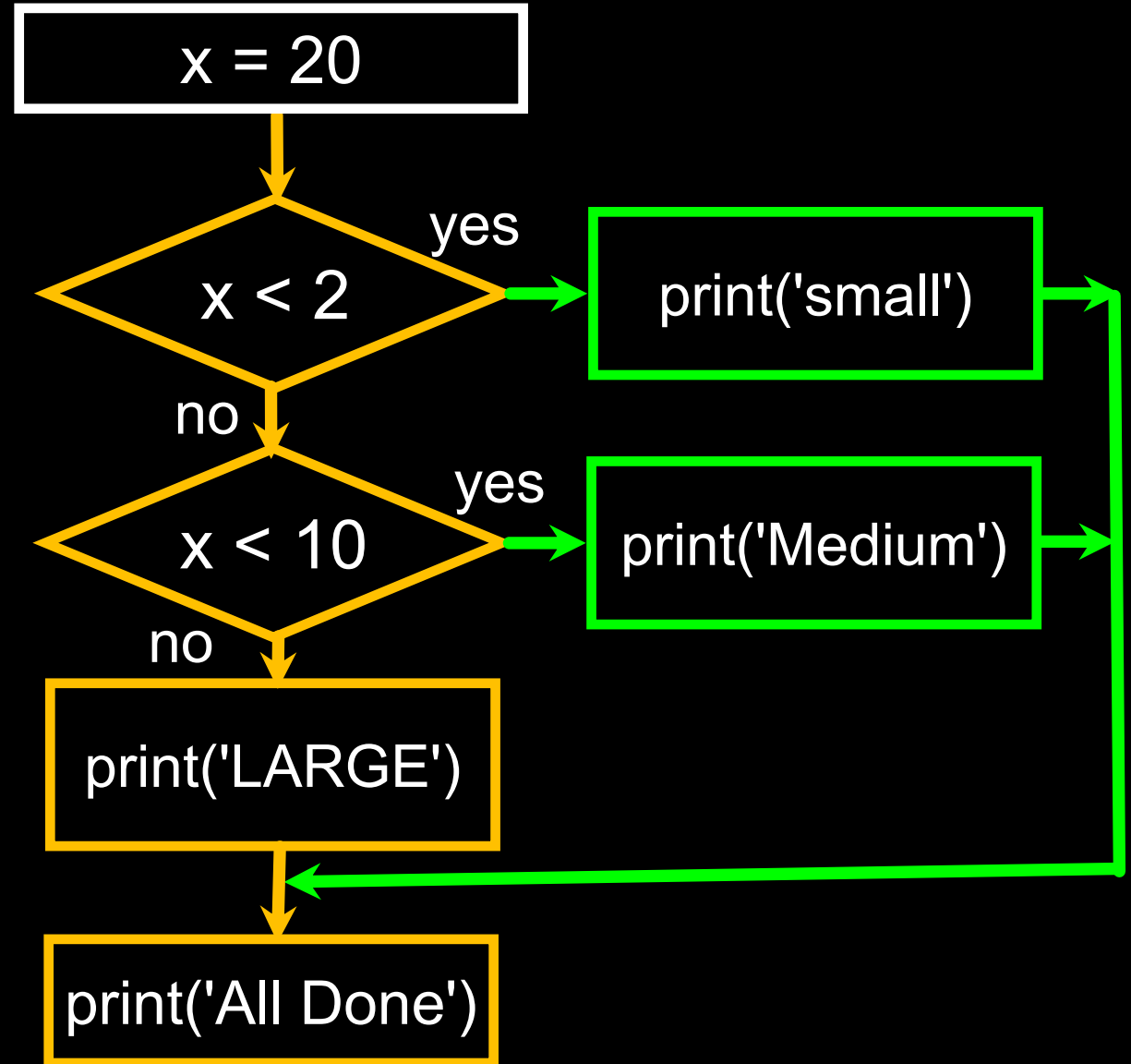
```
if x > 2 :  
    print('Bigger')  
else :  
    print('Smaller')
```

```
print('All done')
```



Multi-way

```
x = 20
if x < 2 :
    print('small')
elif x < 10 :
    print('Medium')
else :
    print('LARGE')
print('All done')
```



Multi-way

```
# No Else
x = 5
if x < 2 :
    print('Small')
elif x < 10 :
    print('Medium')

print('All done')
```

```
if x < 2 :
    print('Small')
elif x < 10 :
    print('Medium')
elif x < 20 :
    print('Big')
elif x < 40 :
    print('Large')
elif x < 100:
    print('Huge')
else :
    print('Ginormous')
```

Sample Program

Problem: WAP to find largest among 3 numbers.

Solution:

```
num1 = int(input("Enter first number: "))  
num2 = int(input("Enter second number: "))  
num3 = int(input("Enter third number: "))
```

```
if num1 > num2 and num1 > num3:  
    print(num1,"is largest number")  
elif num2 > num1 and num2 > num3:  
    print(num2,"is largest number")  
else:  
    print(num3,"is largest number")
```

Sample Program

Problem: Calculate the electric bill using different conditions. For first 100 units Rs1.50 per unit. For next 100 units Rs3.75 per unit. For next 100 units Rs4.25. For rest units Rs5.65 per unit.

Solution:

```
current = int(input("Enter current meter reading: "))
previous = int(input("Enter previous meter reading: "))
total = current - previous

if total <= 100:
    bill = total * 1.5
elif total > 100 and total <=200:
    bill = (100*1.5) + ((total - 100) * 3.75)
elif total > 200 and total <=300:
    bill = (100*1.5) + (100*3.75) + ((total - 200) * 4.25)
else:
    bill = (100*1.5) + (100*3.75) + (total - 200*4.25) + ((total - 300) * 5.65)

print(bill," is total amount")
```

Sample Program

Problem: Get a new string from a given string where 'Is' has been added to the front. If the given string already begins with 'Is' then return the string unchanged

Solution:

```
str=input("Enter a String")
if len(str) >= 2 and str[:2] == "Is":
    print(str)
else:
    print("Is" + str)
```

Sample Program

Problem: Get a the height and weight of a person, compute the BMI and based on BMI print the appropriate message

Solution:

```
height = float(input("What is your height? "))
weight = float(input("What is your weight? "))
bmi = weight / height ** 2
print(bmi)
if bmi < 15:
    print("Very severely underweight")
elif bmi < 16:
    print("Severely underweight")
elif bmi < 18.5:
    print("Underweight")
elif bmi < 25:
    print("Normal (healthy weight)")
elif bmi < 30:
    print("Overweight")
elif bmi < 35:
    print("Obese Class I (Moderately obese)")
elif bmi < 40:
    print("Obese Class II (Severely obese)")
else:
    print("Obese Class III (Very severely obese)")
```

Switch Statement

- A switch statement is a multiway branch statement that compares the value of a variable to the values specified in case statements.
- **Python language doesn't have a switch statement.**
- **Python uses dictionary mapping to implement switch statement in Python**

Assignment

1. Write a program that asks the user to enter a length in centimetres. If the user enters a negative length, the program should tell the user that the entry is invalid. Otherwise, the program should convert the length to inches and print out the result. 1inch = 2.54 centimetres
2. Ask the user for a temperature. Then ask them what units, Celsius or Fahrenheit, the temperature is in. Your program should convert the temperature to the other unit. The conversions are $F = 9/5C + 32$ and $C = 5/9(F - 32)$.
3. Ask the user to enter a temperature in Celsius. The program should print a message based on the temperature: If the temperature is less than -273.15, print that the temperature is invalid because it is below absolute zero. • If it is exactly -273.15, print that the temperature is absolute 0. • If the temperature is between -273.15 and 0, print that the temperature is below freezing. • If it is 0, print that the temperature is at the freezing point. • If it is between 0 and 100, print that the temperature is in the normal range. • If it is 100, print that the temperature is at the boiling point. • If it is above 100, print that the temperature is above the boiling point.

4. Write a program that asks the user how many credits they have taken. If they have taken 23 or less, print that the student is a freshman. If they have taken between 24 and 53, print that they are a sophomore. The range for juniors is 54 to 83, and for seniors it is 84 and over.
5. Generate a random number between 1 and 10. Ask the user to guess the number and print a message based on whether they get it right or not.
6. A store charges \$12 per item if you buy less than 10 items. If you buy between 10 and 99 items, the cost is \$10 per item. If you buy 100 or more items, the cost is \$7 per item. Write a program that asks the user how many items they are buying and prints the total cost.
7. Write a program that asks the user for two numbers and prints Close if the numbers are within .001 of each other and Not close otherwise.
8. A year is a leap year if it is divisible by 4, except that years divisible by 100 are not leap years unless they are also divisible by 400. Write a program that asks the user for a year and prints out whether it is a leap year or not.
9. Write a program that asks the user for an hour between 1 and 12, asks them to enter am or pm, and asks them how many hours into the future they want to go. Print out what the hour will be that many hours into the future, printing am or pm as appropriate. An example is shown below.

Enter hour: 8 am How many hours ahead? 5 New hour: 1 pm



Loops

for loop

Syntax:

for <variable> **in** <sequence>:
 <block of codes>

for each item **in** a set (or in a sequence, or in a collection):
 do something with the item

Example

Ex-1

```
str = '123456789'
```

```
sum = 0
```

```
for ch in str:
```

```
    sum += int(ch)
```

```
print(sum)
```



GUESS THE OUTPUT
???

Ex-2

```
words = ['got' , 'me' , 'looking' , 'so' , 'crazy' , 'right' , now' ]
```

```
for w in words:
```

```
    print (w)
```

Ex-3

```
phrase = 'Silicon Institute Of Technology, Bhubaneswar'  
for w in phrase:  
    print (w)
```

Ex-4

Program to find the sum of all numbers stored in a list

```
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]  
sum = 0  
for val in numbers:  
    sum = sum + val  
print("The sum is", sum)
```



**GUESS THE OUTPUT
???**

Iterating Over tuples and lists

```
for i in [1, 2, 3]:  
    print(2 * i, ', ')  
for word in ['Hello!', 'Ciao!', 'Hi!']:  
    print(word.upper(), ', ')
```



**GUESS THE OUTPUT
???**

range()

- Since we often want to range a variable over some numbers, we can use the **range()** function which gives us a list of numbers from 0 up to but not including the number we pass to it.
- range(5) returns [0,1,2,3,4] So we could say:

```
print(list(range(10)))
```

Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```
list(range(42,-12,-7))
```

Output: [42, 35, 28, 21, 14, 7, 0, -7]

```
start, end, step = 13, 2, -3
```

```
print(list(range(start, end, step)))
```

#Output: [13, 10, 7, 4]

```
for x in range(5):
```

#Output: 0 1 2 3 4 (in separate line)

```
    print x
```

```
for x in range(3, 6):
```

```
    print(x)
```

Prints out 3, 4, 5

```
for x in range(3, 8, 2):
```

```
    print(x)
```

Prints out 3, 5. 7

while loop

While loop repeats as long as certain boolean condition is met

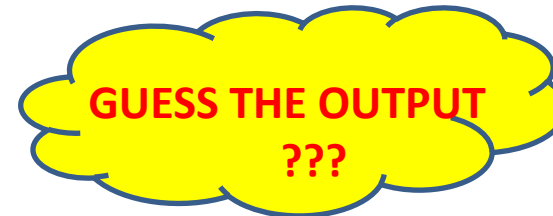
Ex. 1 :

```
count = 0
while count < 5:
    print(count)
    count+=1
```

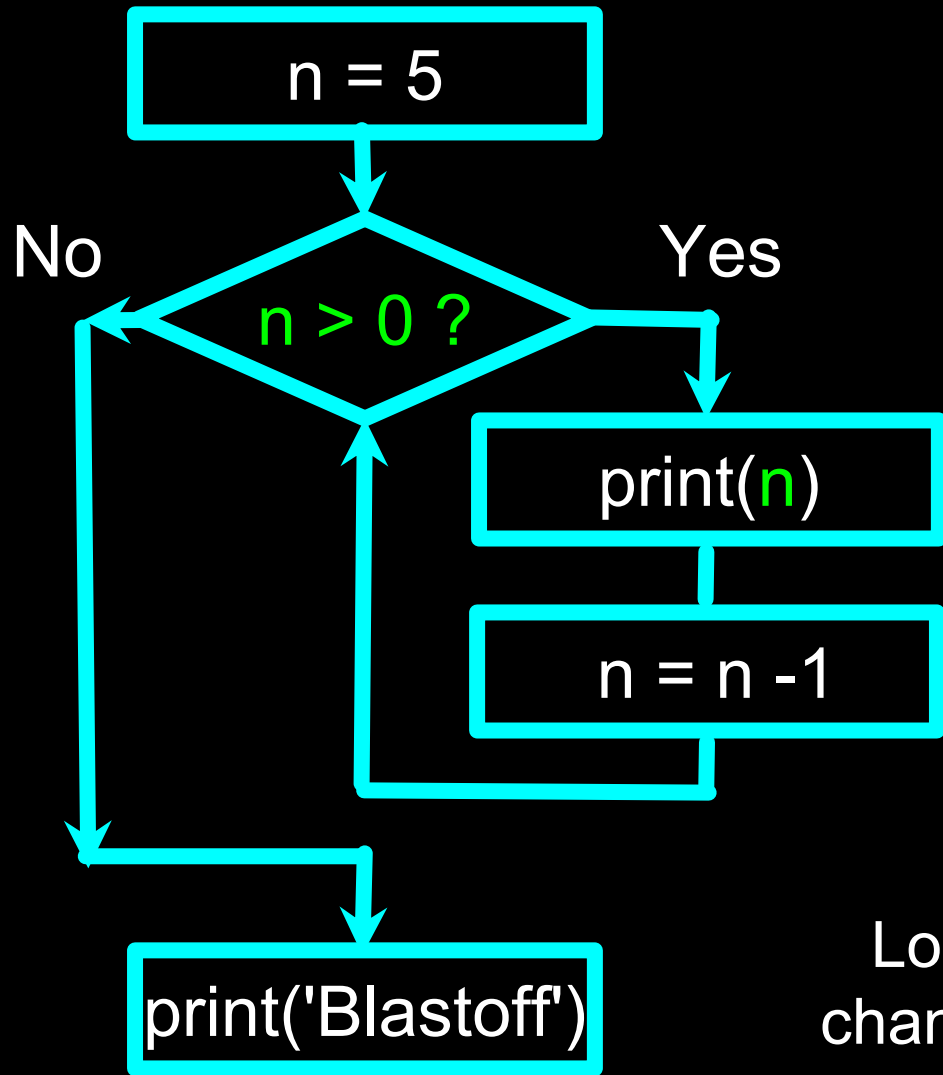


Ex. 2 :

```
n = 10
i = 1
sum = 0
while i <= n:
    sum = sum + i
    i = i + 1          # update counter
print("The sum is", sum)
```



Repeated Steps



Program:

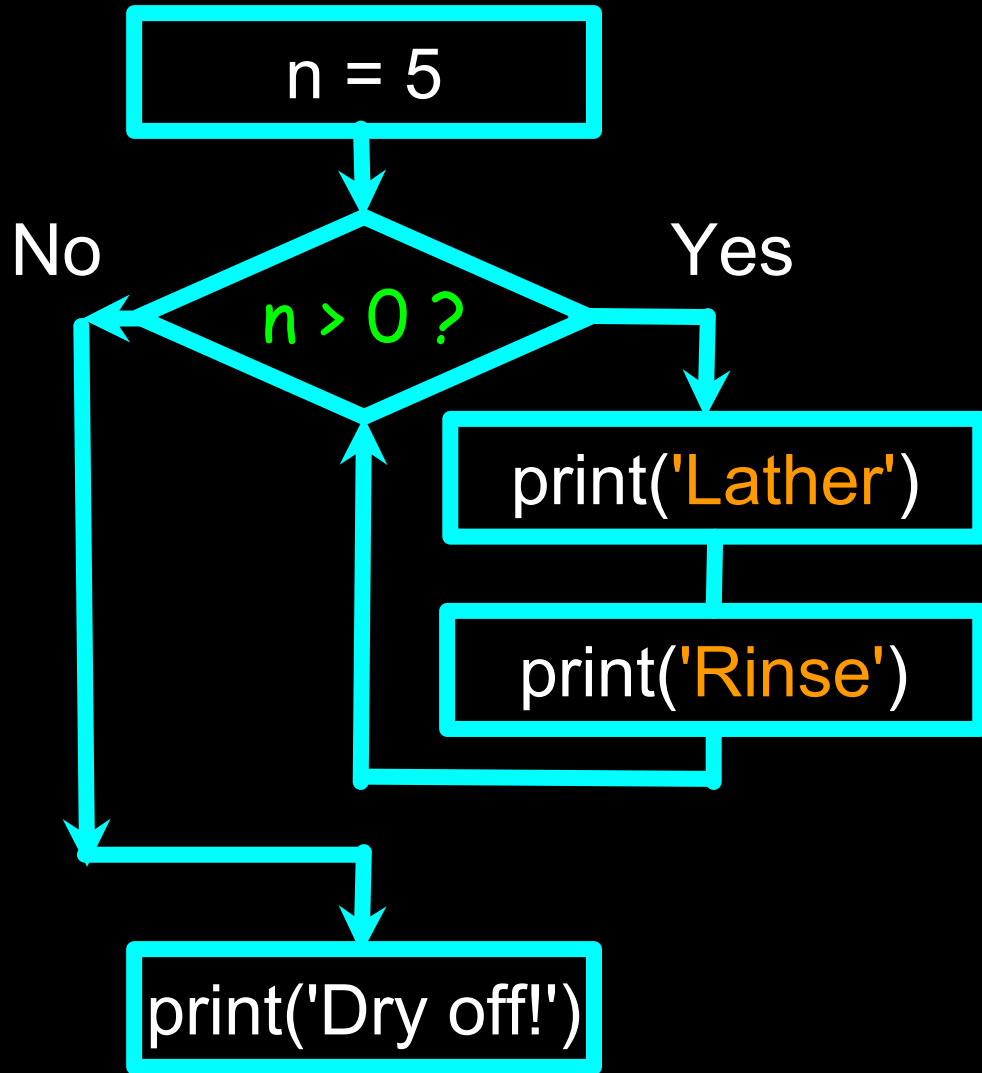
```
n = 5
while n > 0 :
    print(n)
    n = n - 1
print('Blastoff!')
print(n)
```

Output:

5
4
3
2
1
Blastoff!
0

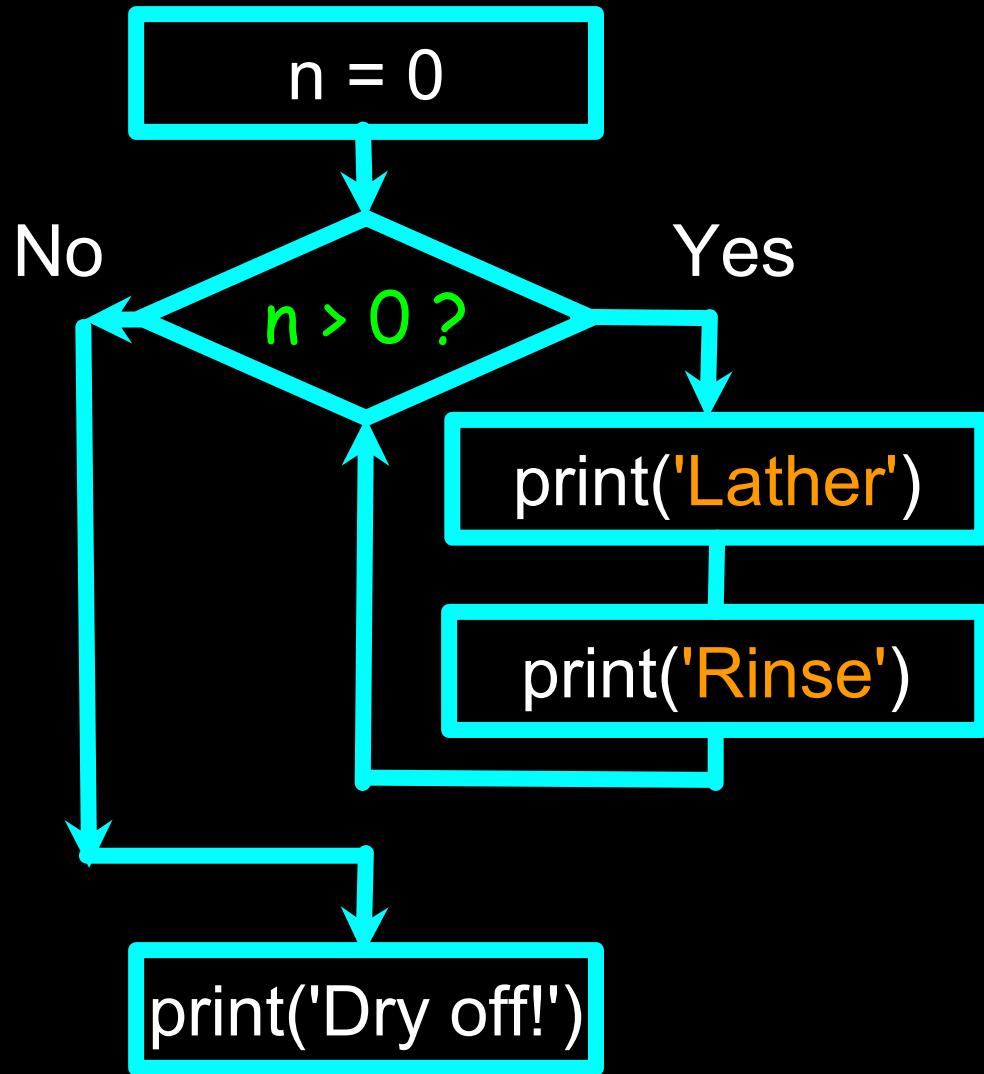
Loops (repeated steps) have **iteration variables** that change each time through a loop. Often these **iteration variables** go through a sequence of numbers.

An Infinite Loop



```
n = 5
while n > 0 :
    print('Lather')
    print('Rinse')
print('Dry off!')
```

What is wrong with this loop?



Another Loop

```
n = 0
while n > 0 :
    print('Lather')
    print('Rinse')
print('Dry off!')
```

What is this loop doing?

break and continue

- **break** is used to exit a for loop or a while loop
- **continue** is used to skip current block, and return to "for" or "while" statement.

Ex-1

```
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break
```



GUESS THE OUTPUT
???

Ex-2

```
for x in range(10):
    if x % 2 == 0:
        continue
    print(x)
```

Check if x is even

Breaking Out of a Loop

- The **break** statement ends the current loop and jumps to the statement immediately following the loop
- It is like a loop test that can happen anywhere in the body of the loop

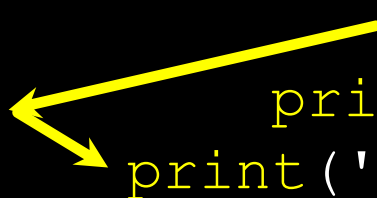
```
while True:
    line = input('> ')
    if line == 'done' :
        break
    print(line)
print('Done!')
```

```
> hello there
hello there
> finished
finished
> done
Done!
```

Breaking Out of a Loop

- The **break** statement ends the current loop and jumps to the statement immediately following the loop
- It is like a loop test that can happen anywhere in the body of the loop

```
while True:
    line = input('> ')
    if line == 'done' :
        break
    print(line)
print('Done!')
```



```
> hello there
hello there
> finished
finished
> done
Done!
```

Ex :

```
for val in "string":  
    if val == "i":  
        continue  
    print(val)  
print("The end")
```



**GUESS THE OUTPUT
???**

Ex :

```
for val in "string":  
    if val == "i":  
        break  
    print(val)  
print("The end")
```

for loop with else

The else part is executed if the items in the sequence used in for loop exhausts. break statement can be used to stop a for loop. In such case, the else part is ignored. Hence, a for loop's else part runs if no break occurs.

Ex :

```
digits = [0, 1, 5]
```

```
for i in digits:
```

```
    print(i)
```

```
else:
```

```
    print("No items left.")
```

Sample Program

- **Problem:** Find a number is Armstrong or not?

Solution:

```
num = int(input("Enter a number: "))
```

```
sum = 0
```

```
temp = num
```

```
while temp > 0:
```

```
    digit = temp % 10
```

```
    sum += digit ** 3
```

```
    temp //= 10
```

```
if num == sum:
```

```
    print(num, "is an Armstrong number")
```

```
else:
```

```
    print(num, "is not an Armstrong number")
```

Sample Program

- **Problem:** Find a number is Prime or not?

Solution:

```
num = int(input("Enter a number: "))
count=0;
if num > 1:
    for i in range(2,num):
        if (num % i) == 0:
            count = count+1;

if count == 0:
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```


Sample Program

Problem: Reverse a word

Solution:

```
word = input("Input a word to reverse: ")
for char in range(len(word) - 1, -1, -1):
    print(word[char], end="")
print('\n')
```

Problem: Write a Python program that prints each item and its corresponding type from the following list.

Sample List : `datalist = [1452, 11.23, 1+2j, True, 'SiliconResource', (0, -1), [5, 12], {'class':'V', 'section':'A'}]`

Solution:

```
datalist = [1452, 11.23, 1+2j, True, 'Silicon', (0, -1), [5, 12], {"class":'V', "section":'A'}]
for item in datalist:
    print ("Type of ",item, " is ", type(item))
```

Sample Program

Problem: Write a Python program that accepts a string and calculate the number of digits and letters.

Sample Data : Python 3.2

Expected Output :

Letters 6

Digits 2

Solution:

```
s = input("Input a string")
```

```
d=l=0
```

```
for c in s:
```

```
    if c.isdigit():
```

```
        d=d+1
```

```
    elif c.isalpha():
```

```
        l=l+1
```

```
    else:
```

```
        pass
```

```
print("Letters", l)
```

```
print("Digits", d)
```

Sample Program

Problem: To find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence

Solution:

```
items = []  
for i in range(100, 401):  
    s = str(i)  
    if (int(s[0])%2==0) and (int(s[1])%2==0) and (int(s[2])%2==0):  
        items.append(s)  
print( ",".join(items))
```

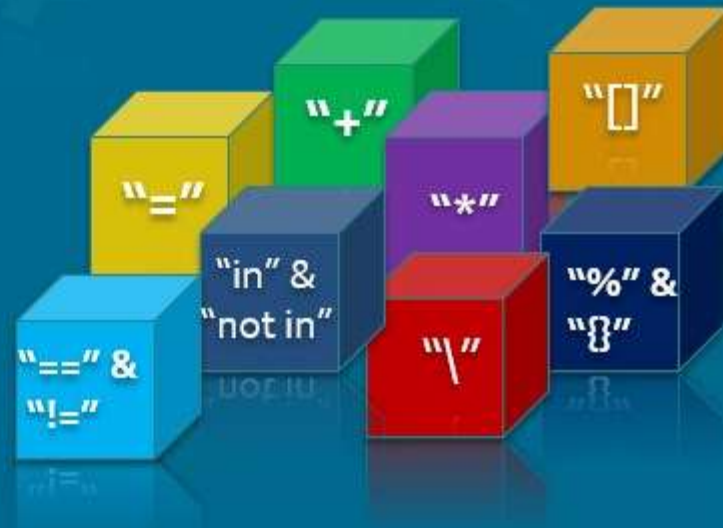
Sample Program

- **Problem:** Find the number of notes (Sample of notes: 1, 2, 5, 10, 20, 50, 100, 200 and 500)against a specified amount

Solution:

```
a=int(input('Enter the Amount:'))
Q = [500, 200, 100, 50, 20, 10, 5, 2, 1]
x = 0
for i in range(9):
    q = Q[i]
    x += int(a / q)
    a = int(a % q)
if a > 0:
    x = -1
print(x)
```

String Operators in Python



Strings are immutable:

```
Str='I love Python'  
Str[0]='U'  
Print(Str)  # Error
```

Modify:


```
Str1='I love Python'  
Str2='U'+Str1[1:]  
Print(Str2)  #U love Python
```

Address of String Object:

```
Str1='Hello'  
Str2='Hello'  
print(id(str1))  
print(id(str2))
```

Unicode Character:

```
print('apple' == 'Apple')  
print('apple' > 'Apple')  
print('A unicode is', ord('A'), ',a unicode is', ord('a'))
```



Python String Comparison

- **Python String** comparison can be performed using equality (==) and comparison (<, >, !=, <=, >=) operators. It is performed using the characters in both strings.
- The characters in both strings are compared one by one.
- When different characters are found then their Unicode value is compared.
- The character with lower Unicode value is considered to be smaller.

Strings Comparison:

```
>>S1='abcd'
>>S2='ABCD'
>>S1>S2      #True
>>S1==S2     #False
```

```
fruit1 = 'Apple'
```

<pre>print(fruit1 == 'Apple')</pre>	True
<pre>print(fruit1 != 'Apple')</pre>	False
<pre>print(fruit1 < 'Apple')</pre>	False
<pre>print(fruit1 > 'Apple')</pre>	False
<pre>print(fruit1 <= 'Apple')</pre>	True
<pre>print(fruit1 >= 'Apple')</pre>	True

Strings Contd...

```
test1.py - C:/Users/Pradyumna/Desktop/test1.py (3.7.4)
File Edit Format Run Options Window Help
print(str(1))
print(str([1.2, 'a']))
print(str())
```

```
=====
1
[1.2, 'a']

>>> |
```

```
test1.py - C:/Users/Pradyumna/Desktop/test1.py (3.7.4)
File Edit Format Run Options Window Help
mystr='ABCDEF'
mycount=len(mystr)
icount=0
for icount in range(mycount):
    print('At index', icount, '->', mystr[icount])
    icount=icount+1
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: C:/Users/Pradyumna/Desktop/test1.py
=====
At index 0 -> A
At index 1 -> B
At index 2 -> C
At index 3 -> D
At index 4 -> E
At index 5 -> F
>>> |
```


Strings Contd...

```
test1.py - C:/Users/Pradyumna/Desktop/test1.py (3.7.4)
File Edit Format Run Options Window Help
myString=input('Enter a String')
L=len(myString)
S1=S2=''
for x in range(L):
    if x%2==0:
        S1=S1+myString[x]
    else:
        S2=S2+myString[x]
print('Odd characters->',S1, 'Even Characters->', S2)
```

```
>>>
===== RESTART: C:/Users/Pradyumna/Desktop/
=====
Enter a StringPradyumna
Odd characters-> Payma Even Characters-> rdun
>>> |
```

```
test1.py - C:/Users/Pradyumna/Desktop/test1.py (3.7.4)
File Edit Format Run Options Window Help
for x in range(5):
    y=x+1
    print(y*str(y))
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
At index 5 -> F
>>>
===== RESTART: C:/Users/Pradyumna/Desktop/test1.py =====
=====
1
22
333
4444
55555
>>> |
```

Strings Contd...

test1.py - C:/Users/Pradyumna/Desktop/test1.py (3.7.4)

File Edit Format Run Options Window Help

```
mystr='Pradyumna Kumar Tripathy'
mystrRev=mystr[::-1]
print('Input String->',mystr)
print('Reverse String',mystrRev)

if mystr==mystrRev:
    print('Pallindrome')
else:
    print('Not a pallindrome')
```

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more informatio
>>>
===== RESTART: C:/Users/Pradyumna/Desktop/test1.py =====
Input String-> Pradyumna Kumar Tripathy
Reverse String yhtapirT ramuK anmuydarP
Not a pallindrome
>>>
```

Comparison of Strings

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'boy'<'girl'
True
>>> 'Boy'<'boy'
True
>>> 'Girl'<'boy'
True
>>> |
```

7/2/2023

Dr. Pradyumna Kumar Tripathy,
Silicon Institute of Technology, Bhubaneswar

94

Difference Between Functions, Methods, and Attributes

- Functions will generally work on various objects (Ex: `bool()`)
- Methods are specific to a particular class of objects
(Ex: `object.method(arguments)`)
- Attributes are also attached to a particular object, but they don't take arguments
(Ex: `object.attribute`)

Strings Contd...

<code>str.capitalize()</code>	First character capitalized rest in lower case
<code>str.count(substr,start,stop)</code>	No of occurrence of substring in a string
<code>str.find(pattern,start,stop)</code>	Returns starting index in the string where substring is present
<code>str.isalnum()</code>	To check if all characters in a string are alphanumeric or not
<code>str.isalpha()</code>	To check if all characters in a string are alphabets or not
<code>str.isdigit()</code>	To check the inputted character is a digit or not
<code>min(str)</code>	Returns smallest character from str
<code>max(str)</code>	Returns largest character from str
<code>str.isspace()</code>	Returns True if str contains only white space
<code>str.endswith('str1')</code>	Check whether str ends with str1
<code>str.startswith('str1')</code>	Check whether str starts with str1
<code>count(str S1)</code>	Returns the number of occurrences of this substring

Strings Contd...

```
>>>S1="I have brought two chocolates, two cookies, and two cakes"
```

```
#Replace "two" by new string "three"
```

```
>>>S2=S1.replace("two", "three")
```

```
#Replaces all occurrences
```

```
>>>S2
```

```
" I have brought three chocolates, three cookies, and three cakes"
```

```
# Replace two chocolates and two cookies by three chocolates and three cookies
```

```
>>>S1="I have brought two chocolates, two cookies, and two cakes"
```

```
>>S1.replace ("two", "three",2)
```

```
# Replaces only first 2 occurrences of "two" by "three"
```

```
"I have brought three chocolates, three cookies, and two cakes"
```

Sample Program

Problem: Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '\$', except the first char itself.

Ex: Input: restart

Output: resta\$t

Solution:

```
str1=input('Enter a string')  
char = str1[0]  
str1 = str1.replace(char, '$')  
str1 = char + str1[1:]  
print(str1)
```

Sample Program

Problem: Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string

Input: 'abc', 'xyz'

Output: 'xyc, abz'

Solution:

```
a=input('Enter first string')
b=input('Enter Second string')
new_a = b[:2] + a[2:]
new_b = a[:2] + b[2:]
print(new_a + ',' + new_b)
```

Problem: Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it

Solution:

```
str1=input('Enter a string')
length = len(str1)
if length > 2:
    if str1[-3:] == 'ing':
        str1 += 'ly'
    else:
        str1 += 'ing'
print(str1)
```

Sample Program

Problem: Write a Python function to create the HTML string with tags around the word(s).

Example:

Input: 'i', 'Python'

Output: '<i>Python</i>'

'b', 'Python Tutorial'

Output: 'Python Tutorial '

Solution:

```
tag=input('Input tag')
word=input('Input word')
print("<%s>%s</%s>" % (tag, word, tag))
```

Write a Python program to count repeated characters in a string

Problem: Write a Python program to swap comma and dot in a string.

Solution:

```
amount = "32.054,23"
maketrans = amount.maketrans
amount = amount.translate(maketrans(',.', '.,'))
print(amount)
```


Sample Program

Problem: Write a Python program to find the first appearance of the substring 'not' and 'poor' from a given string, if 'not' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.

Input: 'The lyrics is not that poor!'

Output: 'The lyrics is good!'

Solution:

```
str1=input('Enter a string')
snot = str1.find('not')
spoor = str1.find('poor')
if spoor > snot and snot>0 and spoor>0:
    str1 = str1.replace(str1[snot:(spoor+4)], 'good')
print(str1)
```

Sample Program

Problem: In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

Example:

Input: abc

Output: cde

Solution:

```
realText=input('Enter the text')
step=int(input('enter the steps'))
outText = []
cryptText = []
uppercase = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
lowercase = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
for eachLetter in realText:
    if eachLetter in uppercase:
        index = uppercase.index(eachLetter)
        crypting = (index + step) % 26
        cryptText.append(crypting)
        newLetter = uppercase[crypting]
        outText.append(newLetter)
    elif eachLetter in lowercase:
        index = lowercase.index(eachLetter)
        crypting = (index + step) % 26
        cryptText.append(crypting)
        newLetter = lowercase[crypting]
        outText.append(newLetter)
print(outText)
```

Sample Program

Problem: Write a Python program to print four values decimal, octal, hexadecimal (capitalized), binary in a single line of a given integer.

Sample Output:

Input an integer: 25

Decimal Octal Hexadecimal (capitalized), Binary

25 31 19 11001

Solution:

```
i = int(input("Input an integer: "))
o = str(oct(i))[2:]
h = str(hex(i))[2:]
h = h.upper()
b = str(bin(i))[2:]
d = str(i)
print("Decimal Octal Hexadecimal (capitalized), Binary")
print(d, ' ', o, ' ', h, ' ', b)
```

Sample Program

Problem: Write a Python program to wrap a given string into a paragraph of given width.

Sample Output:

Input a string: The quick brown fox.

Input the width of the paragraph: 10

Result:

The quick
brown fox.

Solution:

```
import textwrap
s = input("Input a string: ")
w = int(input("Input the width of the paragraph: ").strip())
print("Result:")
print(textwrap.fill(s,w))
```

Sample Program

Problem: Write a Python program to move all spaces to the front of a given string in single traversal.

Solution:

```
Str1=input('Enter a string'):  
no_spaces = [char for char in str1 if char!=' ']  
space= len(str1) - len(no_spaces)  
# Create string with spaces  
result = ' '*space  
print (result + ''.join(no_spaces))
```

Contact Me



Dr. Pradyumna Kumar Tripathy

*Associate Professor & Head, Dept. of CSE,
Silicon Institute of Technology, Bhubaneswar*

Python Programming

Mobile: +91- 9437141874

Email: ptripathy@silicon.ac.in

