Board of Technical Education

332- KVT POLYTECHNIC (Govt. Aided)

C.V.V. CAMPUS,

CHICKBALLAPUR-562101

PYTHON PROGRAMMING

20CS31P

LAB MANUAL

Third Semester

Department of Computer Science

Prepared By:

RAVI KUMAR H.K, B.E., M.Tech

Selection grade Lecturer, Dept. of CSE

Student Name :- _____

Reg No:- 332CS_____

Fundamental Concepts

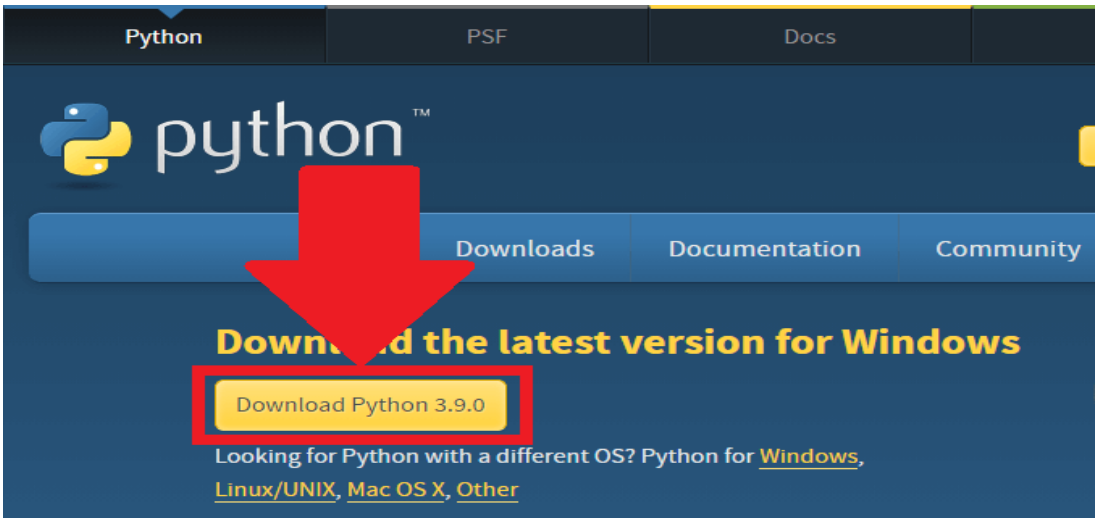
Week 1 Practice Exercises

1. Setup python environment

Steps to Download and Install Python 3.9 on Windows

Step 1: Download Python 3.9

To start, go to python.org/downloads and then click on the button to download the latest version of Python:

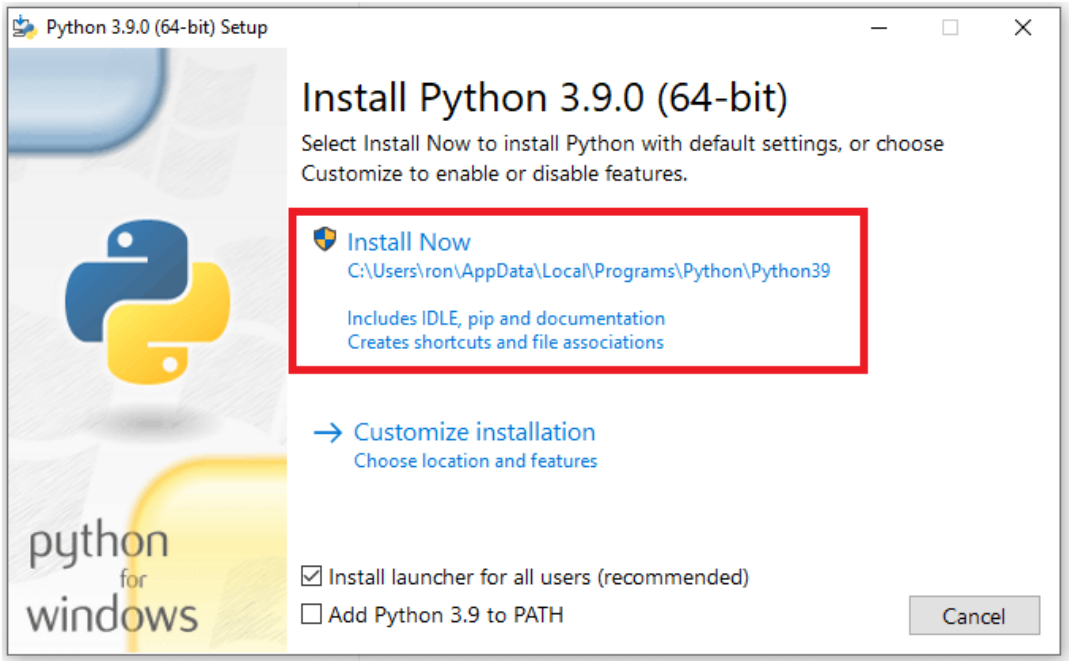


Step 2: Run the .exe file

Next, run the .exe file that you just downloaded:

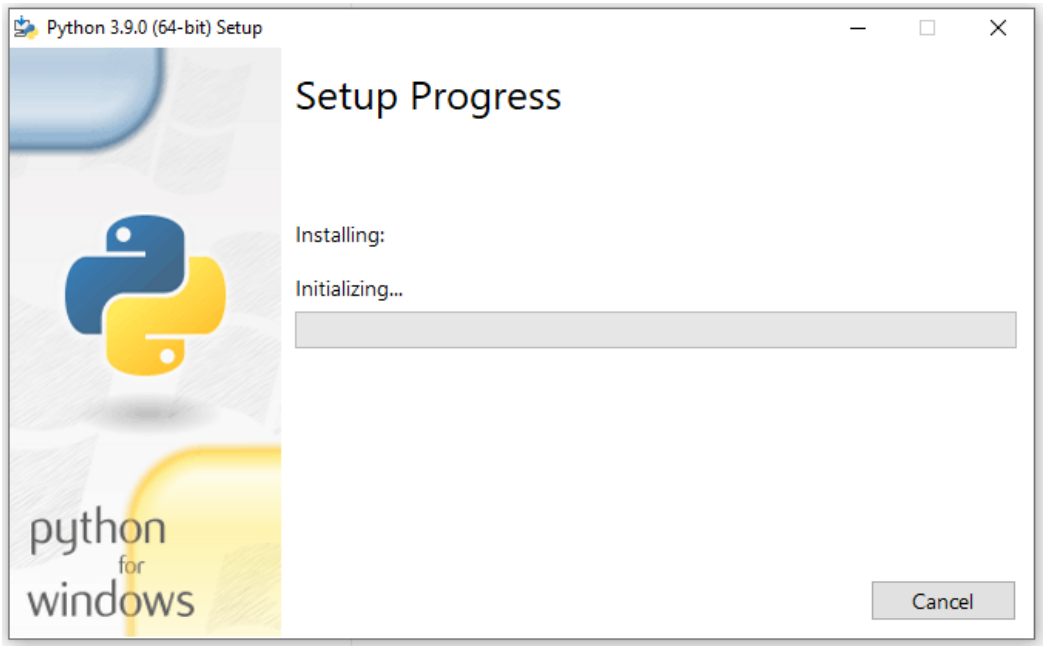
Step 3: Install Python 3.9

You can now start the installation of Python by clicking on **Install Now**:

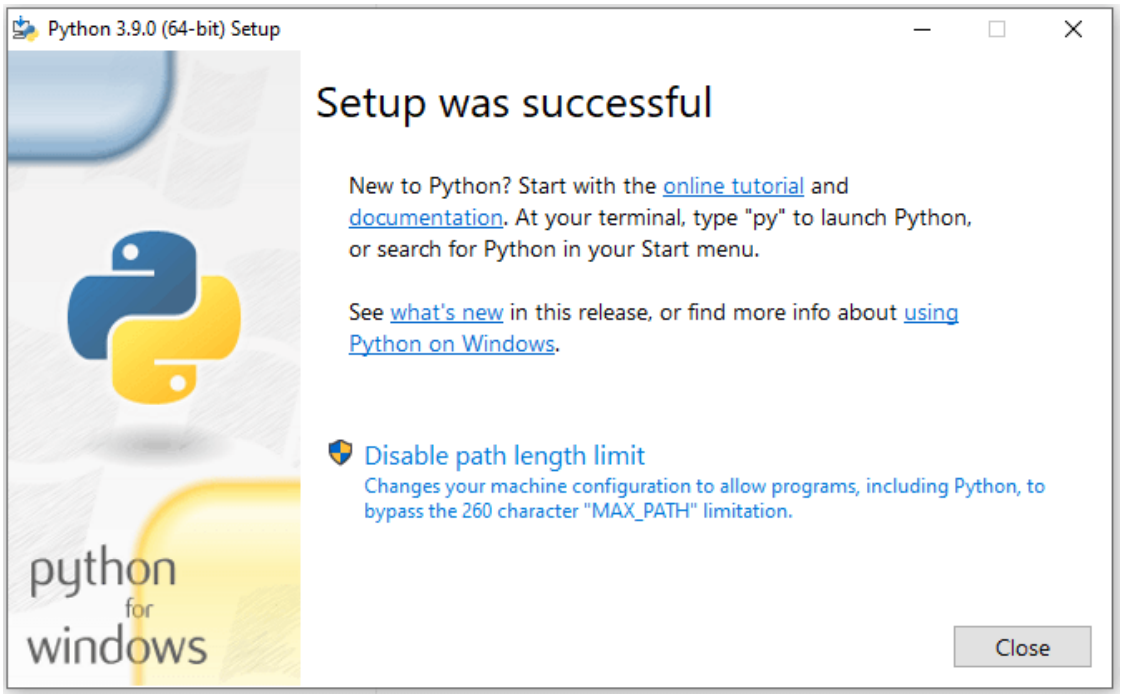


Note that depending on your needs, you may also check the box to add Python to the Path.

Your installation should now begin:



After a short period of time, your setup would be completed:

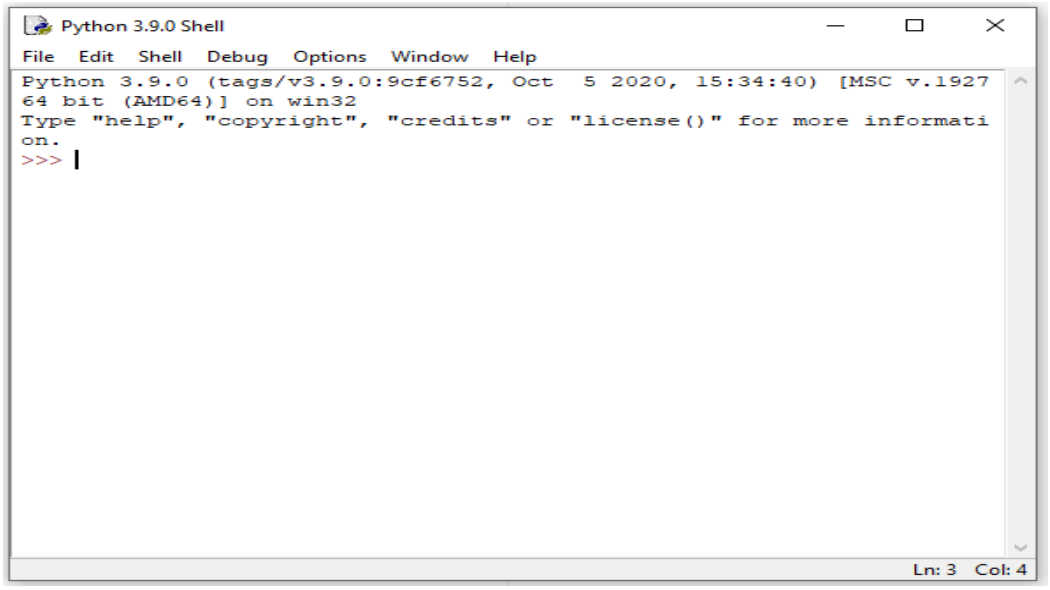

Python 3.9.0 (64-bit) Setup
Setup was successful
New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.
See what's new in this release, or find more info about using Python on Windows.
Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.
python for windows
Close

2. Run a Code in Python

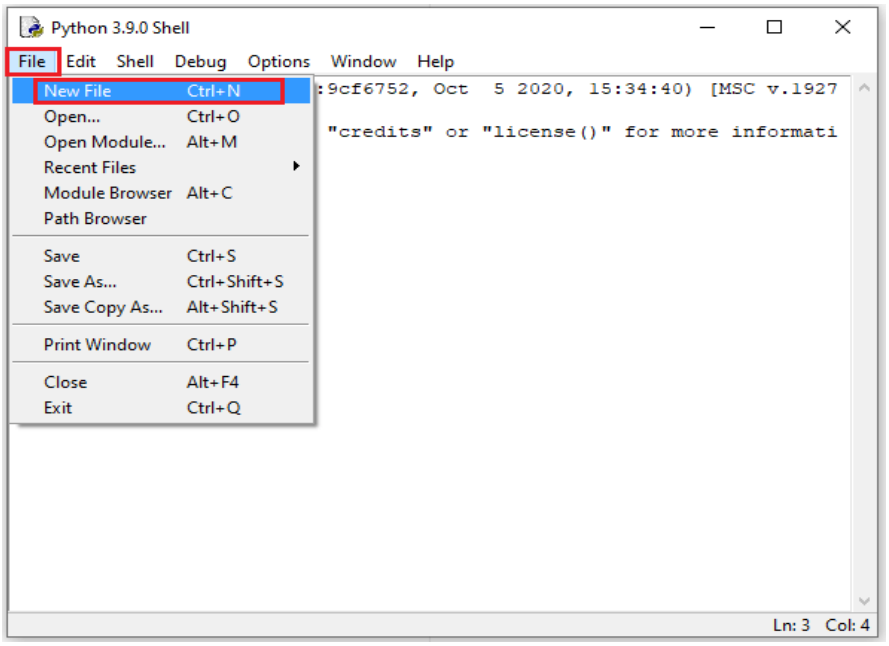
You can run a code in Python via the Python IDLE.

A quick way to find your Python IDLE on Windows is by clicking on the *Start* menu. You should then see the IDLE under "Recently added"

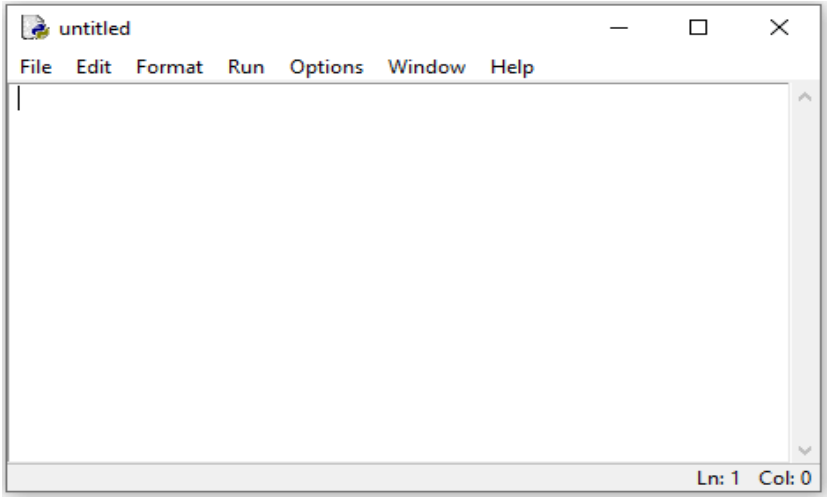
Once you click on the Python IDLE, you'll see the Shell screen:



Click on **File** and then select **New File** (alternatively, you may use the keyboard shortcut of Ctrl+N):



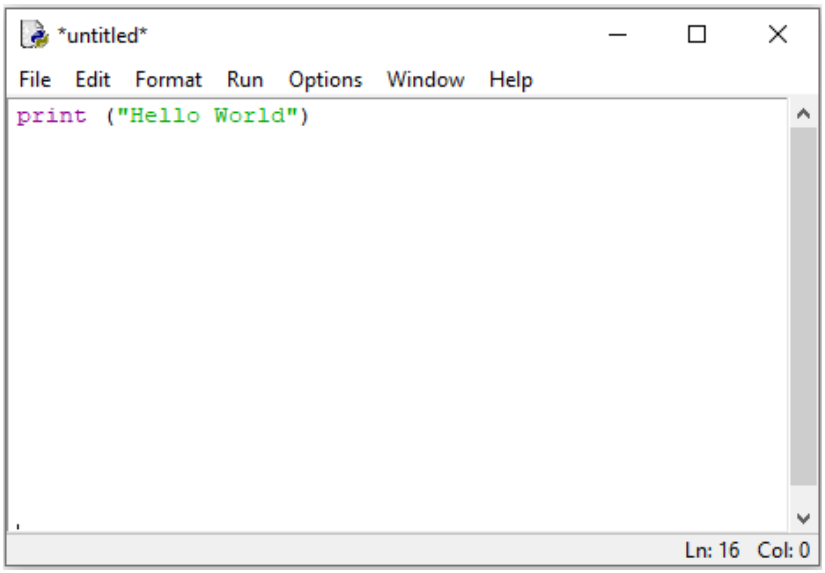
You would now see the following “untitled” box, where you can type your Python code:



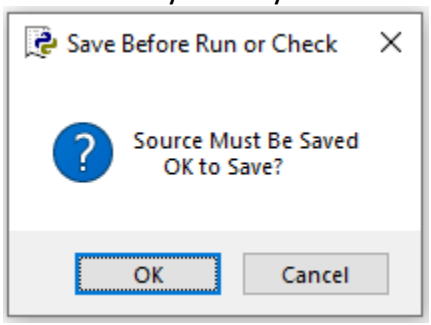
For example, type the command below. This command will print the famous expression of “Hello World”

```
print ("Hello World")
```

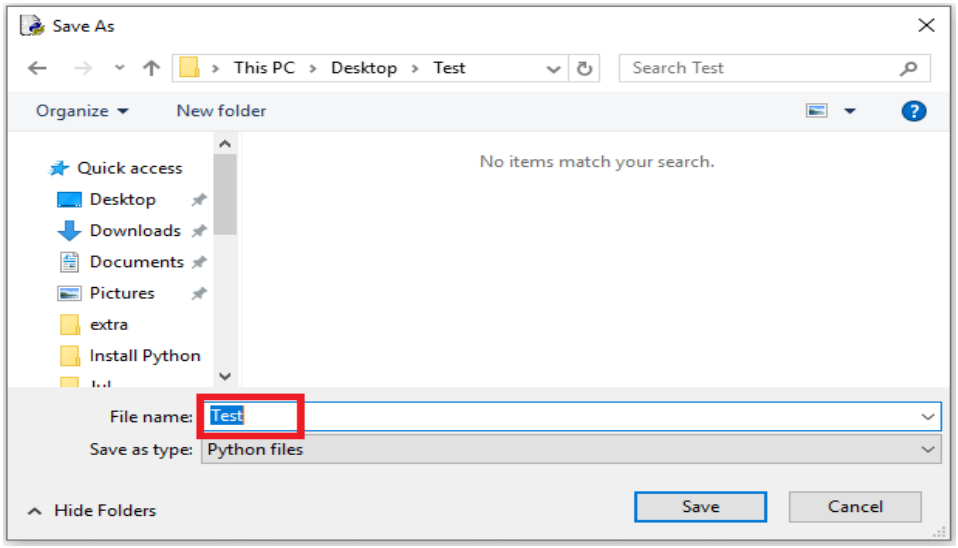
This is how the syntax would look like in the “untitled” box:



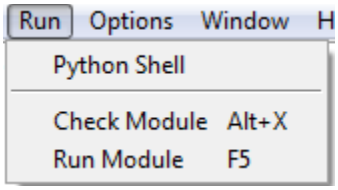
Press **F5** on your keyboard. You will then get the following message to save your code:



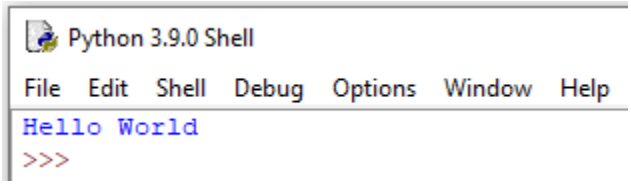
Choose a location where the Python file will be saved on your computer. You'll also need to type a name for your file. For example, type "Test" for your file name:



Once you're done, press **Save.**
Click on **Run** and then select **Run Module** (alternatively, you may use the keyboard shortcut F5):



you'll then see the "Hello World" expression printed on your Python Shell:



Input :
Print('Hello Narendra,Chethan')

Output :
Hello Narendra,chethan
>>>

3. Executing python: Explore different ways to run python program

What is an IDE?

An IDE (Integrated Development Environment) is a software application used by developers for creating programs. IDEs are meant to make the developer's job easier by combining tools that are necessary during software development. Your typical IDE will contain tools such as:

- a text editor;
- a compiler and/or interpreter;
- a debugger and code profiler;
- version control integration;
- a number of supporting utilities to interface with external tooling (Docker, cloud deployments, etc.)

…all combined into a single user interface.

IDE vs. code editor

An IDE is a complex tool that will have your back during the entire process of software development. However, for smaller projects—or those people who would value customization above everything else—a code editor might be enough.

A code editor doesn't have the word "integrated" in the name for a reason; it's just an editor, with additional features like syntax highlighting and code formatting.

The best Python IDEs and code editors

- PyCharm
- Visual Studio Code
- Sublime Text
- Vim
- Atom
- Jupyter Notebook
- Eclipse + PyDev + LiClipse
- GNU Emacs
- Spyder
- Thonny

Here are the ways with which we can run a Python script.

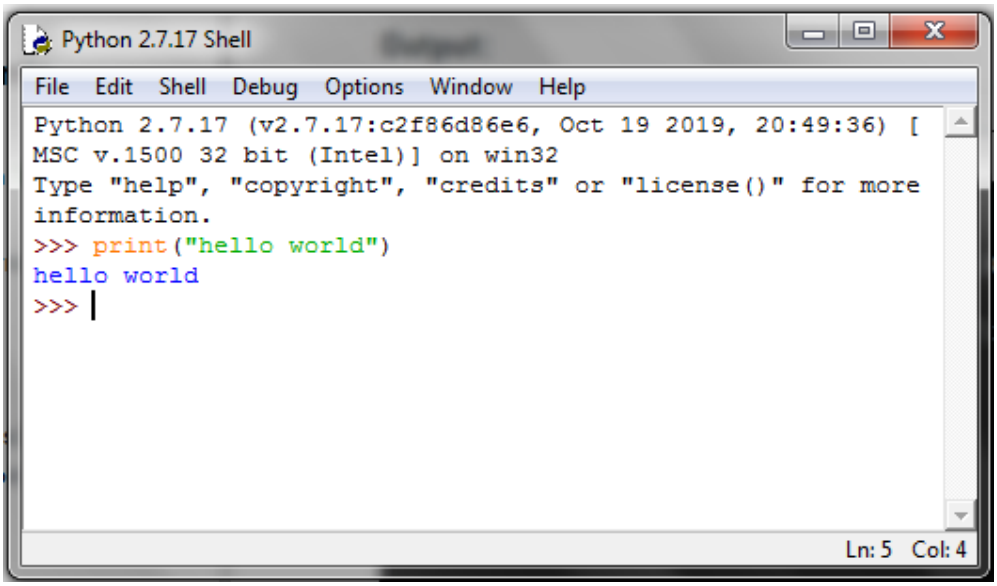
1. Interactive Mode
2. Command Line
3. Text Editor (Notepad)
4. IDE (Python IDLE)

3.1 Interactive Mode:

In Interactive Mode, you can run your script line by line in a sequence. To enter in an interactive mode, you will have to open Command Prompt on your windows machine.

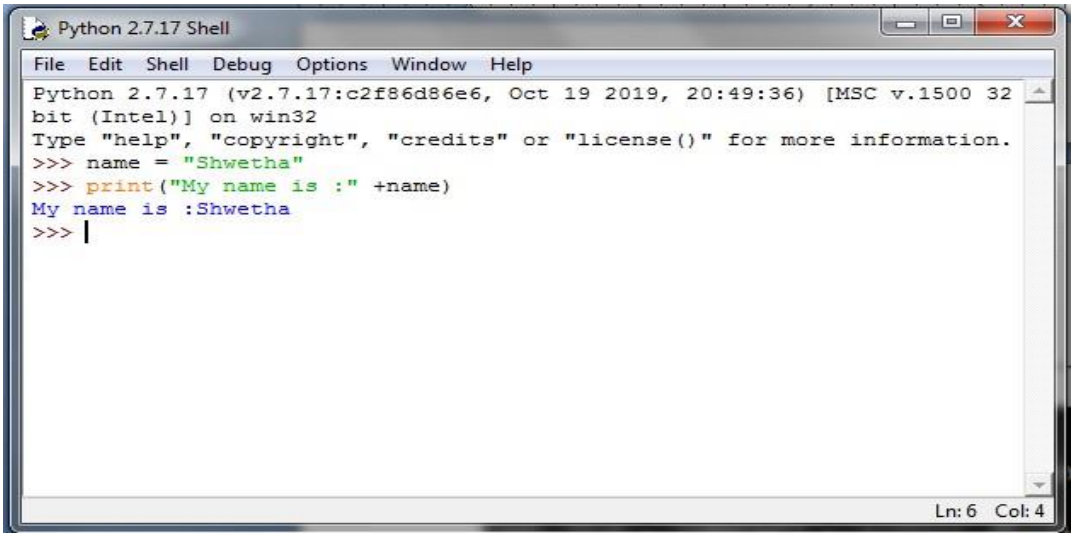
Example 1:

Run the following line in the interactive mode:



Example 2:

Run the following lines one by one in the interactive mode:



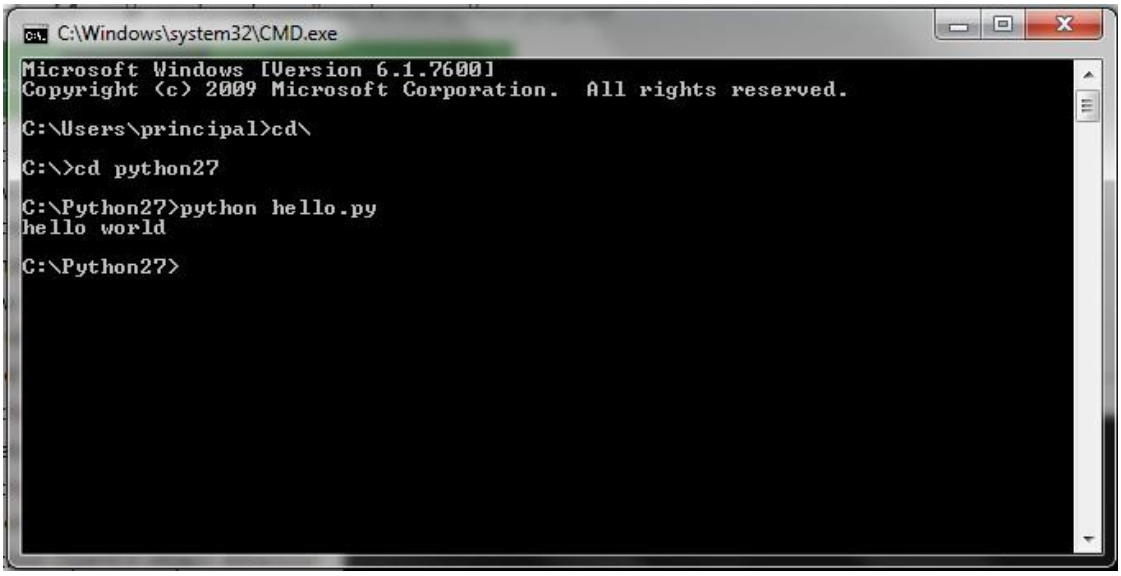
Note: To exit from this mode, press 'Ctrl+Z' and then press 'Enter' or type 'exit()' and then press Enter.

3.2. Command Line

To run a Python script store in a '.py' file in command line, we have to write 'python' keyword before the file name in the command prompt.

python hello.py

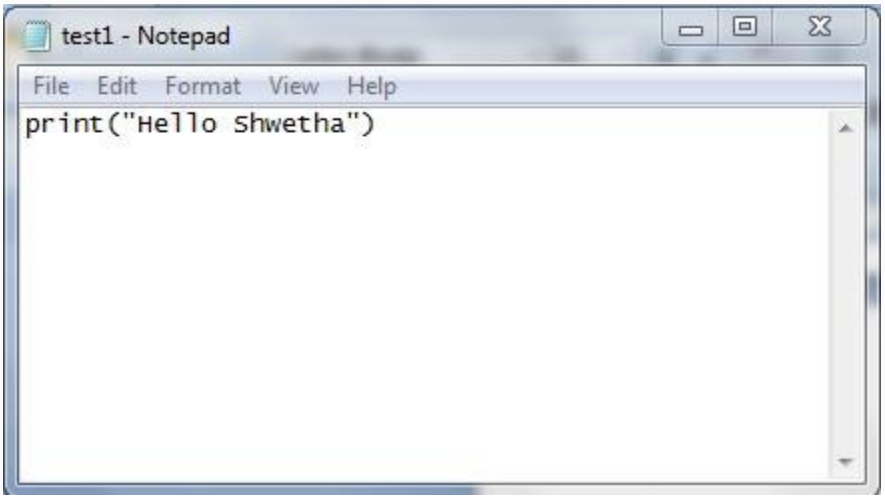
You can write your own file name in place of '**hello.py**'.



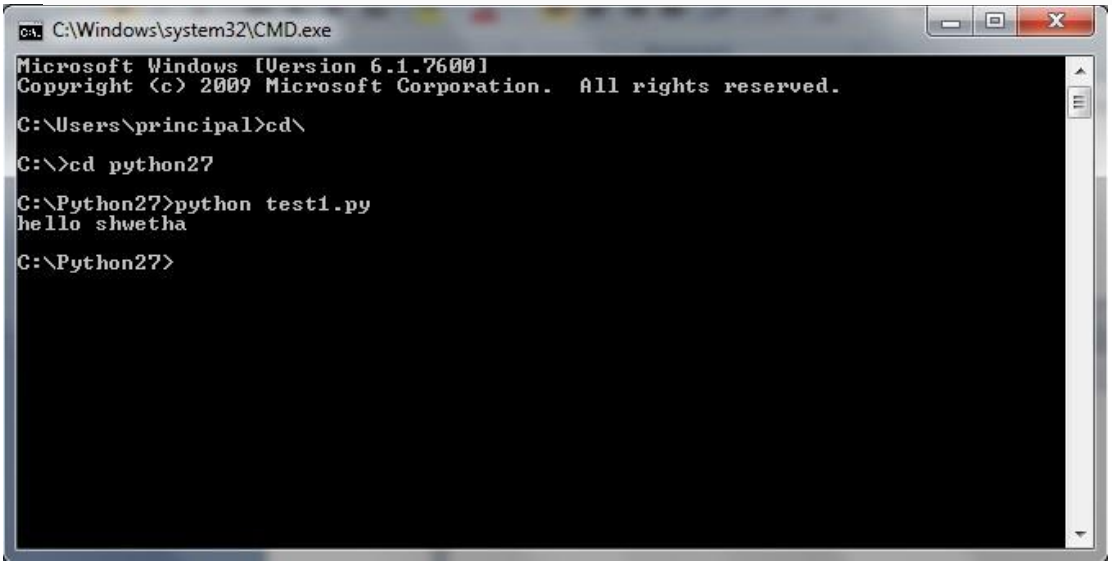
3.3 Use Notepad

A programmer can use any text editor to write a Python script. Open new file in notepad and type the following:

print ("Hello World")



- Save the File as "test1.py" under C:\Python27
- Use the Python Interpreter to Run the Program



3.4 IDE (Python IDLE)

To run Python script on a IDE (Integrated Development Environment), refer ***Section 2 Run a Code in Python***

5. Debug python code

Debugging means the process of finding errors, finding reasons of errors and techniques of their fixation. An error, also known as a bug, is a programming code that prevents a program from its successful interpretation.

Errors are of three types–

- Compile Time Error
- RunTime Error
- Logical Error

Compile time error:
These errors are basically of 2types– Syntax errors and semantics error.
Syntax Error: Violation of formal rules of a programming language results in syntax error.

For ex-
len('hello') = 5

File "<stdin>", line 1
SyntaxError: can't assign to function call

Semantics Error: Semantics refers to the set of rules which sets the meaning of statements. A meaningless statement results in semantics error.

For ex-
x * y = z

Logical Error
If a program is not showing any compile time error or runtime error but not producing desired output, it may be possible that program is having a logical error.

Some example-
•Use a variable without an initial value.

•Provide wrong parameters to a function

•Use of wrong operator in place of correct operator required for operation

X=a + b (here – was required in place of + as per requirement)

Runtime Error
These errors are generated during a program execution due to resource limitation. Python is having provision of checkpoints(Exceptions) to handle these errors.

For ex-

```
a=10
b=int(input("enter a number"))
c=a/b
```

Value of b to be entered at runtime and user may enter 0 at runtime that may cause runtime error, because any number can't be divided by 0.

In python debugging can be done through

- **Print line debugger**
- **Debugging tool**

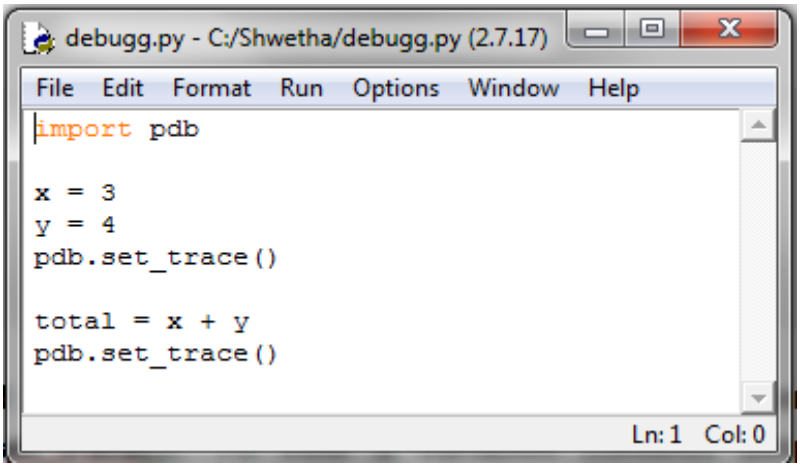
Python Debugger: pdb

The module pdb supports setting breakpoints. A breakpoint is an intentional pause of the program. where you can get more information about the programs state.

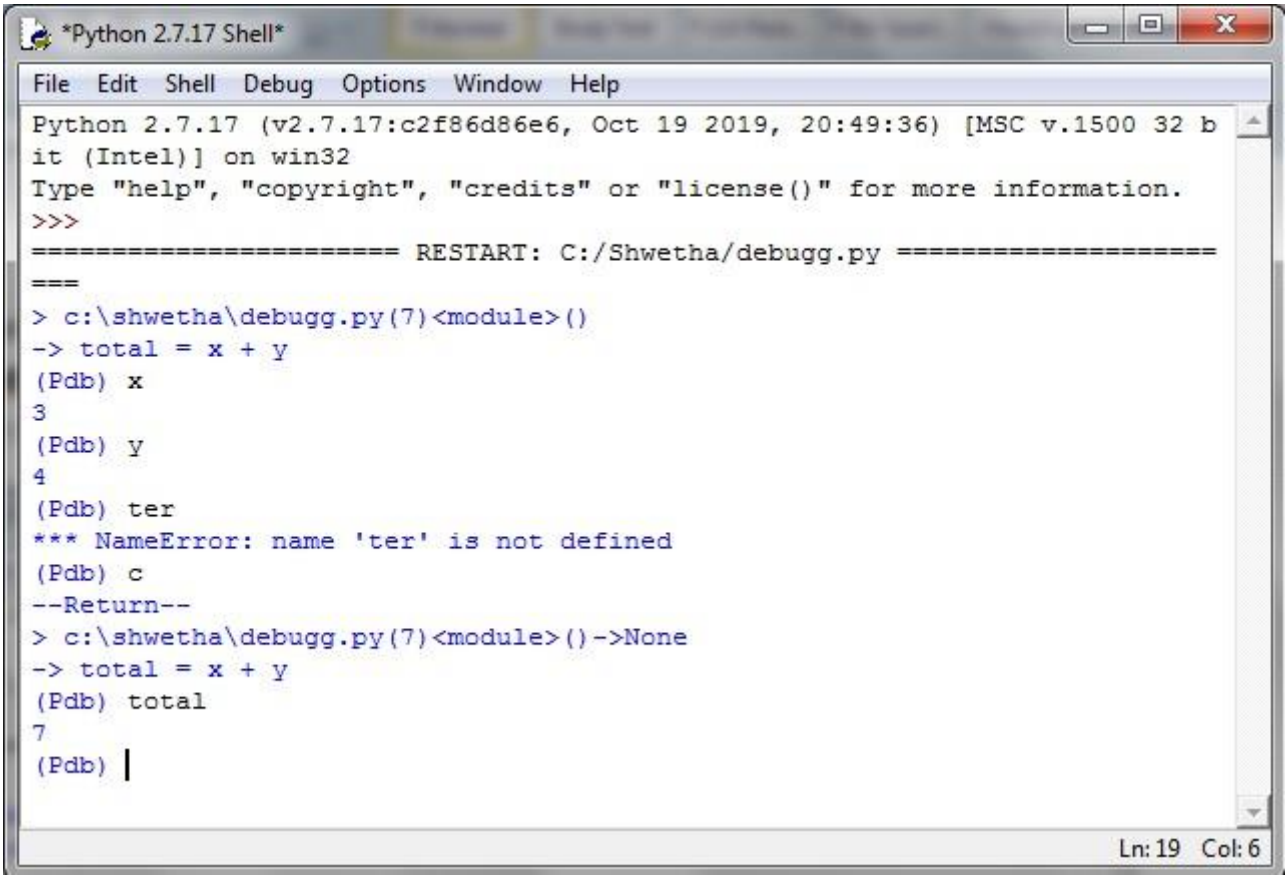
To set a breakpoint, insert the line

```
pdb.set_trace()
```

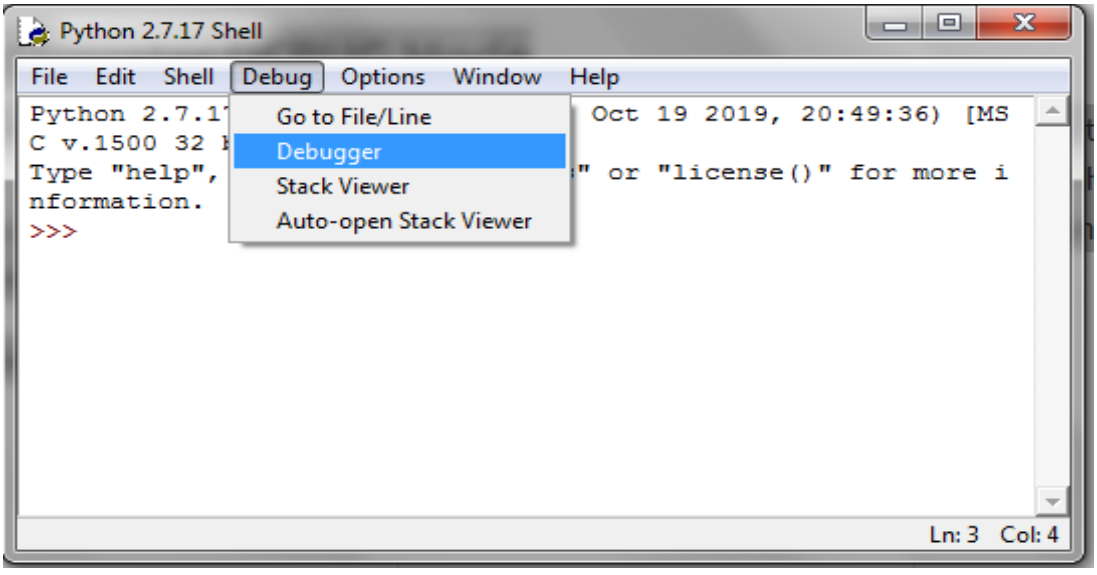
Example



We have inserted a few breakpoints in this program. The program will pause at each breakpoint (**pdb.set_trace()**). To view variables contents simply type the variable name. Press **c** or **continue** to go on w ith the programs execution until the next breakpoint.

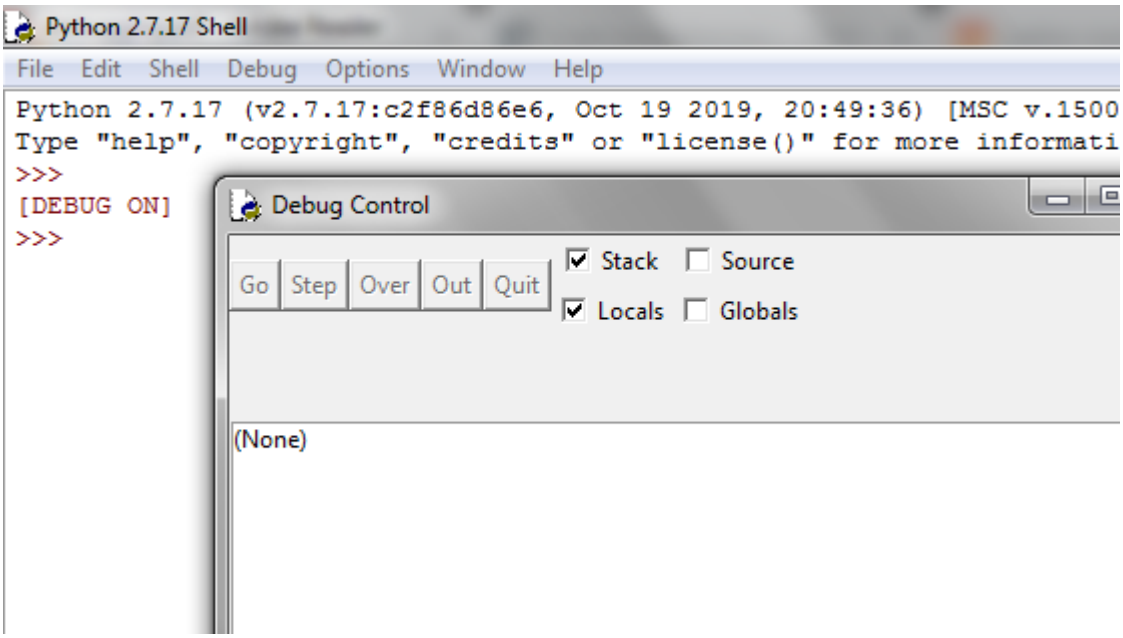


Interpreter DEBUG Mode

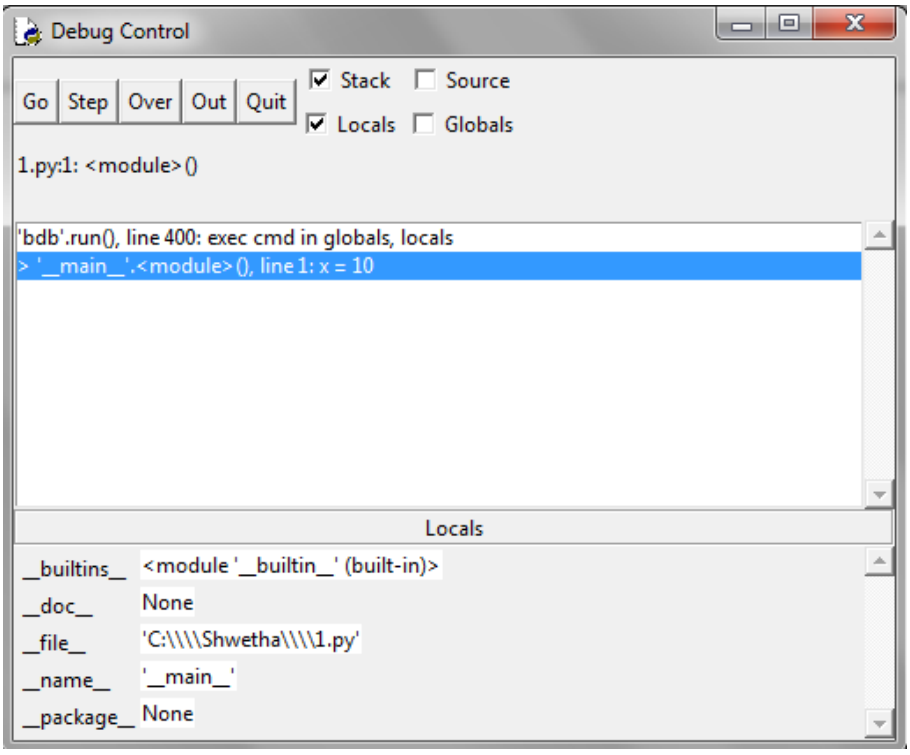


If you want to run your code with the built-in debugger, then you'll need to turn this feature on. To do so, select *Debug* → *Debugger* from the Python IDLE menu bar. In the interpreter, you should see [DEBUG ON] appear just before the prompt (>>>), which means the interpreter is ready and waiting.

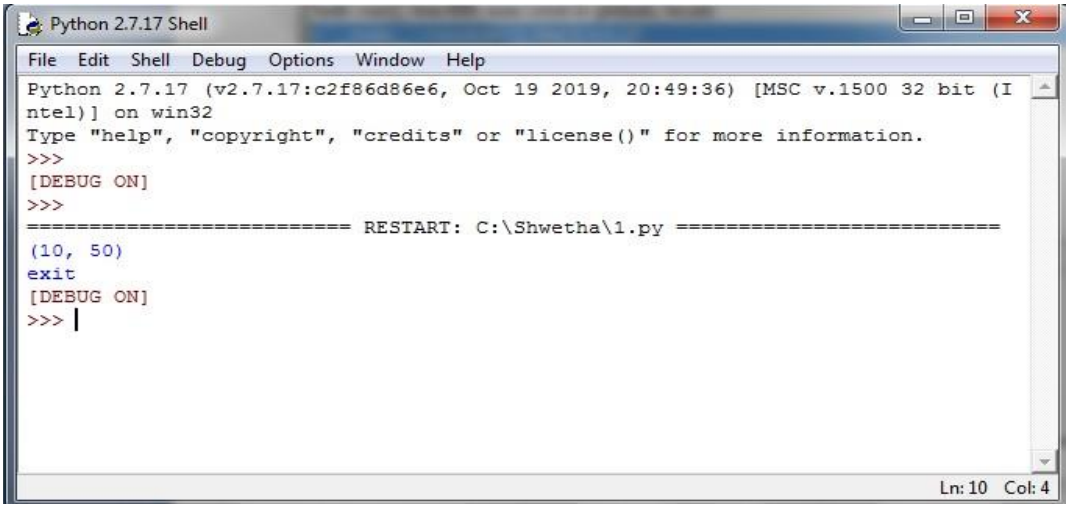
When you execute your Python file, the debugger window will appear:



Then, we will open our program from file menu and will run it.



Click on STEP button for each line execution one by one and result will be displayed in output window. When we will get wrong value, we can stop the program there and can correct the code.



In debug window, you can inspect the values of your local and global variables as your code executes. This gives you insight into how your data is being manipulated as your code runs.

You can also click the following buttons to move through your code:

- **Go:** Press this to advance execution to the next breakpoint. You'll learn about these in the next section.
- **Step:** Press this to execute the current line and go to the next one.
- **Over:** If the current line of code contains a function call, then press this to step *over* that function. In other words, execute that function and go to the next line, but don't pause while executing the function (unless there is a breakpoint).
- **Out:** If the current line of code is in a function, then press this to step *out* of this function. In other words, continue the execution of this function until you return from it.

Be careful, because there is no reverse button! You can only step forward in time through your program's execution.

You'll also see four checkboxes in the debug window:

1. **Globals:** your program's global information
2. **Locals:** your program's local information during execution
3. **Stack:** the functions that run during execution
4. **Source:** your file in the IDLE editor

When you select one of these, you'll see the relevant information in your debug window.

Breakpoints

A **breakpoint** is a line of code that you've identified as a place where the interpreter should pause while running your code. They will only work when *DEBUG* mode is turned on, so make sure that you've done that first.

To set a breakpoint, right-click on the line of code that you wish to pause. This will highlight the line of code in yellow as a visual indication of a set breakpoint. You can set as many breakpoints in your code as you like. To undo a breakpoint, right-click the same line again and select *Clear Breakpoint*.

Once you've set your breakpoints and turned on *DEBUG* mode, you can run your code as you would normally. The debugger window will pop up, and you can start stepping through your code manually.

Errors and Exceptions

When you see an error reported to you in the interpreter, Python IDLE lets you jump right to the offending file or line from the menu bar. All you have to do is highlight the reported line number or file name with your cursor and select *Debug → Go to file/line* from the menu bar. This is will open up the offending file and take you to the line that contains the error. This feature works regardless of whether or not *DEBUG* mode is turned on.

Python IDLE also provides a tool called a **stack viewer**. You can access it under the *Debug* option in the menu bar. This tool will show you the traceback of an error as it appears on the stack of the last error or exception that Python IDLE encountered while running your code. When an unexpected or interesting error occurs, you might find it helpful to take a look at the stack. Otherwise, this feature can be difficult to parse and likely won't be useful to you unless you're writing very complicated code.

Tutorial Activity

1. Compare and contrast excel and python

2. Identify various python IDEs and identify differences between them.

Basics I/O operations

Week 2 Practice Exercises

EXERCISE - 2(a) Use i/o statements

Aim: **write a program to accept two numbers from the user and return their sum**

Description:

Python 3.6 uses the input() method.
Python 2.7 uses the raw_input() method.

input() → The input() function allows user input.

Syntax: **input(*prompt*)** Prompt- A String, representing a default message before the input.

Print()->The print() function prints the specified message to the screen, or other standard output device.

Syntax: **print*(object(s)*, sep=*separator*, end=*end*, file=*file*, flush=*flush*)**

Parameter	Description
object(s)	Any object, and as many as you like. Will be converted to string before printed
sep='*separator*'	Optional. Specify how to separate the objects, if there is more than one. Default is ' '
end='*end*'	Optional. Specify what to print at the end. Default is '\n' (line feed)
file	Optional. An object with a write method. Default is sys.stdout
flush	Optional. A Boolean, specifying if the output is flushed (True) or buffered (False). Default is False

```
print("Hello, how are you?")
print('Hello', 'World', sep='--- ', end='!!!')
```

```
Hello, how are you?
Hello---World!!!
```

Algorithm:

Input: Enter two integers

Output: Sum of two integers

Step1: Start

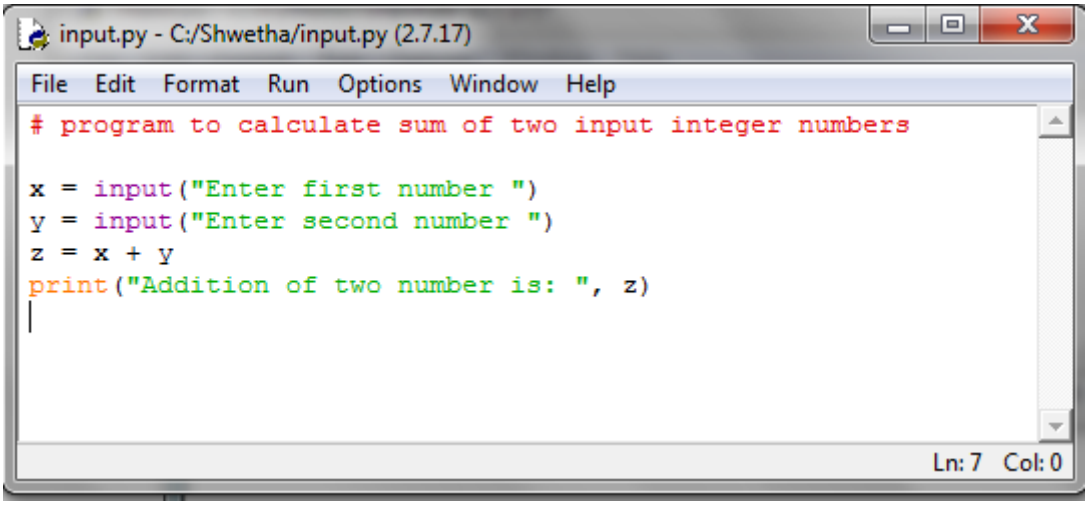
Step2: Enter two integers

Step3: Find sum

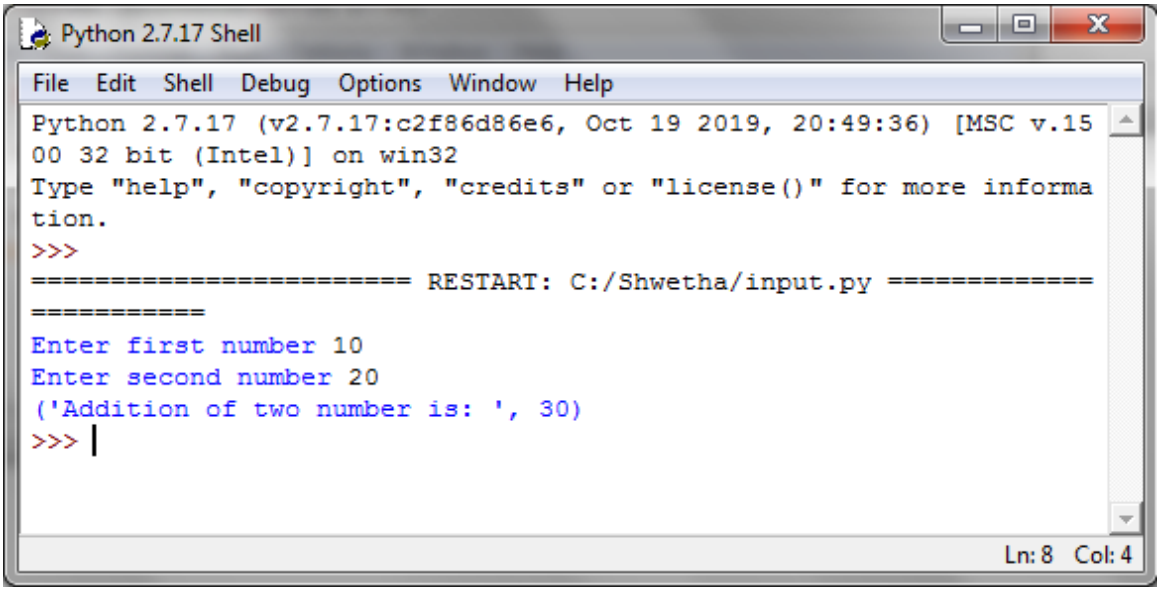
Step4: Print Sum

Step5: stop

Program:



Ouput:



2 (b) Evaluate expressions and displays formatted output

Aim: Write a program to evaluate expression

Description:

An **expression** is a combination of values, variables, and operators. If you type an expression on the command line, the interpreter **evaluates** it and displays the result.

The *evaluation of an expression* produces a value, which is why expressions can appear on the right hand side of assignment statements. A value all by itself is a simple expression, and so is a variable.

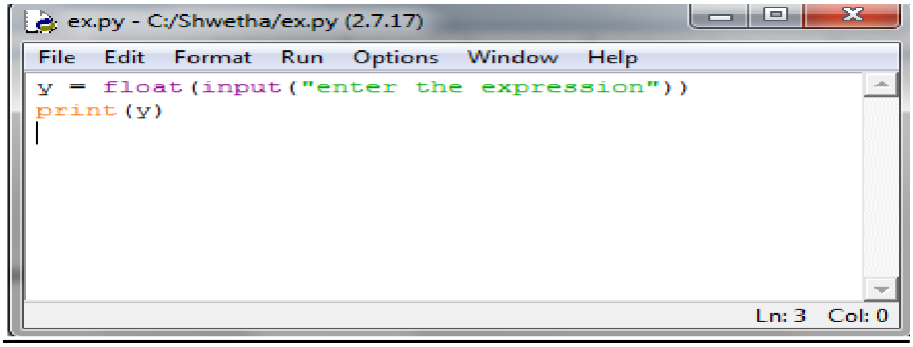
Algorithm:

Input: Enter the expression

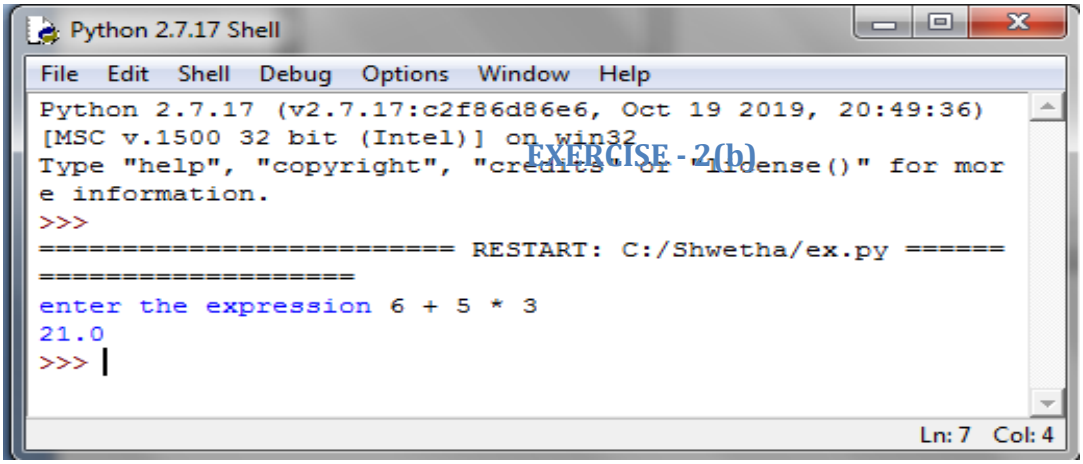
Output: Result after evaluating

Step1: Start
Step2: Enter the expression
Step3: Evaluate the expression
Step4: Print Result
Step5: Stop

Python program:



Output:



EXERCISE - 2(b)

2(c) **Evaluate expressions to examine the operator precedence**

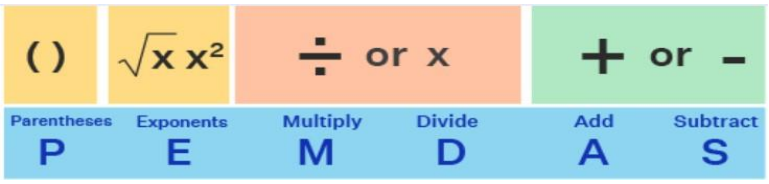
Aim: Write a program evaluate an expression using operator precedence prints its result.

Description:

An expression is made with combinations of variables, values, operators and function calls. The Python interpreter evaluates the valid expression.

Python Operators Precedence Rule – PEMDAS

P – **Parentheses**
E – **Exponentiation**
M – **Multiplication**
D – **Division**
A – **Addition**
S – **Subtraction**



Note: Operators in the same row are evaluate left to right except ** (Exponentiation) which is right to left binding

((((6 + 4) * 2) – 10) // 2) – 4 * 2

Let's break down the evaluation:

((((6 + 4) * 2) – 10) // 2) – 4 * 2
(((10 * 2) – 10) // 2) – 4 * 2
(20 – 10) // 2) – 4 * 2
(10 // 2) – 4 * 2
5 – 4 * 2
5 – 8
-3
	(6+4) = 10
	(10*2) = 20
	(20-10) = 10
	(10//2) = 5
	4*2 = 8
	5-8 = -3

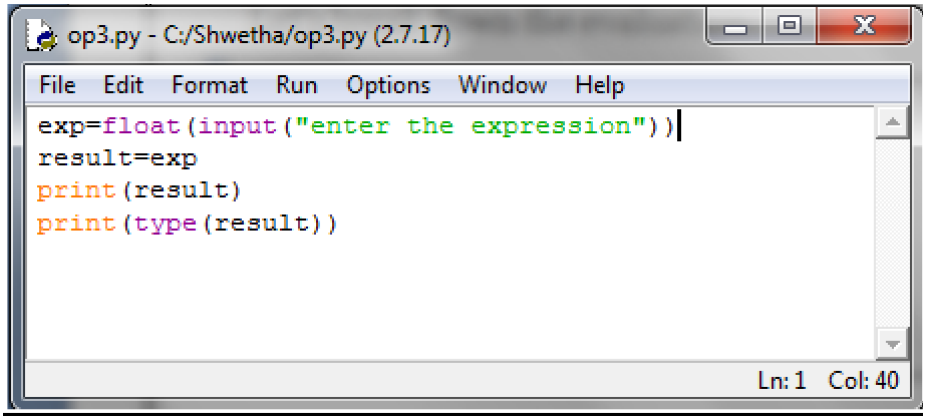
Algorithm:

Input: Enter the expression

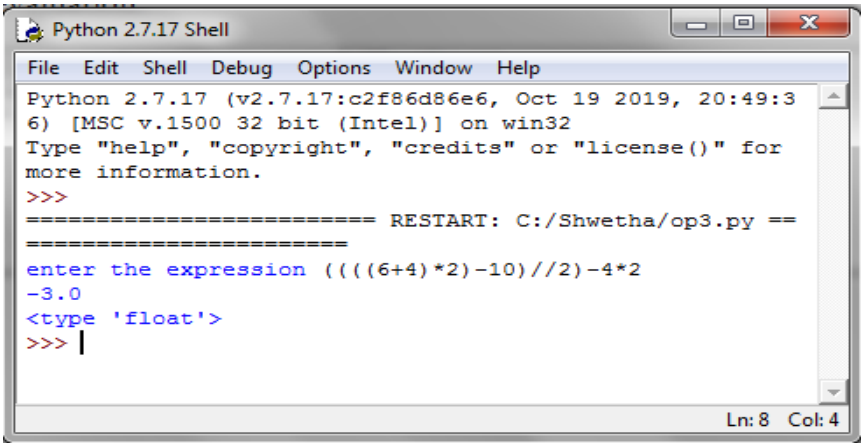
Output: Result after evaluating using operator precedence

Step1: Start
Step2: Enter the expression
Step3: Evaluate the expression according to PEDMAS rule
Step4: Print Result
Step5: Stop

Python program:



Output:



Viva questions:

1. How to represent complex numbers in python?
2. Write a print statement to truncate the number 12.34567 to 2 decimal places?
3. How many reserved keywords are there in python 2.7 and python 3.6?
4. List the bitwise operators in python?
5. What is the importance of type conversion?
6. What are the differences between Membership and Identity operators?
7. How to determine the type of a variable in Python?

Perimeter	Square Rectangle	P = 4a P = 2(l+b)
Circumference	Circle	C = 2*pi*r
Area	Square Rectangle Triangle Trapezoid Circle	$A = a^2$ A = l*b A = bh/2 $A = (b_1+b_2)h/2$ $A = pi*r^2$
Surface Area	Cube Cylinder Cone Sphere	$S = 6a^2$ $S = 2*pi*r*h$ $S = pi*r*l$ $S = 4*pi*r^2$
Volume	Cylinder Cone Sphere	V = bh V = bh/3 $V = 4 * pi * r^3/3$
Pythagorus Theorem	$a^2+b^2 = c^2$	
Distance Formula	$d = sqrt[(x_2-x_1)^2+(y_2-y_1)^2]$	
Slope of a line	$m = (y_2-y_1)/(x_2-x_1)$	
Midpoint Formula	$M = ((x_1+x_2)/2,(y_1+y_2)/2)$	

Week 3: Control Flow: Conditional blocks

In Python, there are four types of Python Conditional Statements are exists.

1. If Statement
2. If-else statement
3. Nested if statements
4. if-elif-else statement

Syntax:

If Statement :	**If-else statement :**
if (test condition): statement1 ..... statement n statement x	**if** (test condition): statement1 **else**: statement2
Nested if Statements :	**if-elif-else statement :**
if (test condition1): statement1 **if** (test condition2): statement2 ..... **if** (test condition n): statement n	**if** (test condition1): statement1 **elif** (test condition2): statement2 ..... **elif** (test condition n): statement n **else**: statement n

EXERCISE - 3(a)

Aim: Write a Program for checking whether the given number is a even number or not.

Description:

If a number is exactly divisible by 2(Remainder is 0) then it is said to be an Even number otherwise, the number is said to be an Odd number. In Python, We use modulus(%) operator to find the remainder.

Decision making is used to specify the order in which the statements are executed. We can use if…else statement to check whether the number is even or odd.

if...else statement:

This is a two-way decision making statement that decides what to do when the condition is true and what to do when the condition is false.

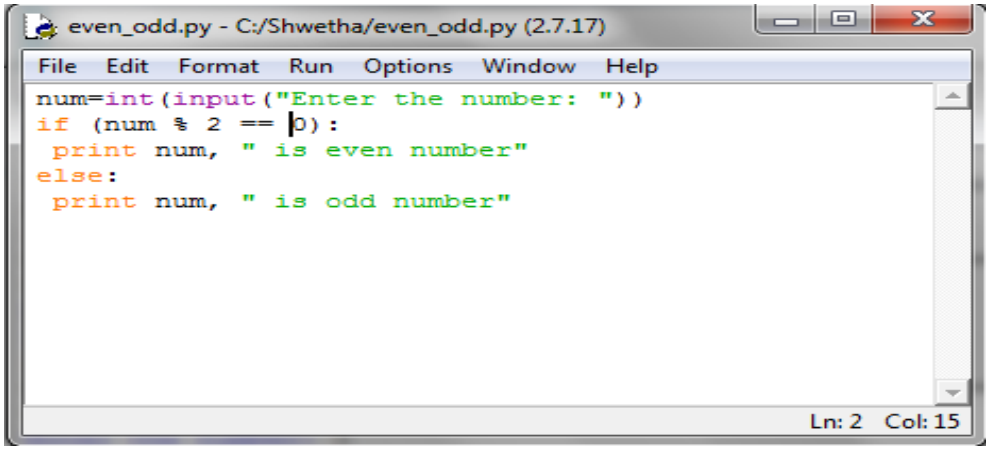
Algorithm:

Input: A Number

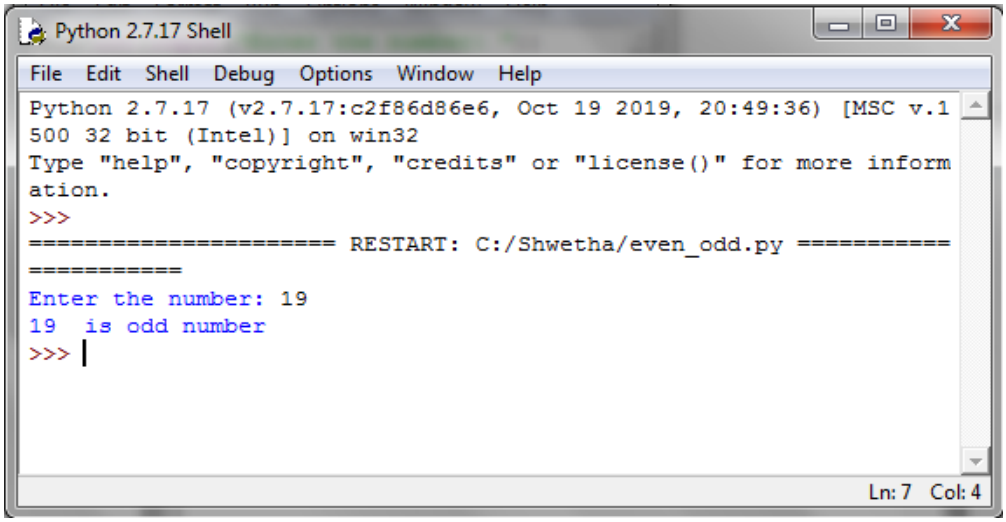
Output: A Message

Step1: Start
Step2: Read num
Step3: Check whether the num is divisible by 2 or not. If yes, goto Step4. else, goto Step5
Step4: Display “The number is even” and goto step6
Step5: Display “The number is odd"and goto step6
Step6: Stop

Program:



Ouput:



Real-time example:-

(a) If you a*re going out to play?, put on a hat*

program.py >

```
1 answer = input('Are you going out to play? ')
2 if answer == 'yes':
3   print('Put on a hat!')
```

(b) If it is raining, take an umbrella if not put on a hat

program.py >

```
1 answer = input('Is it raining? ')
2 if answer == 'yes':
3   print('Take an umbrella!')
4 else:
5   print('Put on a hat!')
```

Exercise:

- If I finish dinner then I can have dessert.
- If my homework is done then I can go outside to play.
- If it is a Tuesday then I have ballet class.
- If it's raining then I wear my gumboots, else I wear my runners.
- If it's Saturday then I watch a movie after dinner, else I read a book.
- If there is chocolate then I will pick that flavour ice-cream, else I will pick strawberry.
- If it's raining then I wear my gumboots, else I wear my runners
- If age is greater than 18, then you are eligible to cast your vote

EXERCISE - 3(b)

Aim: Program to Find the Largest of Three Numbers

Description:

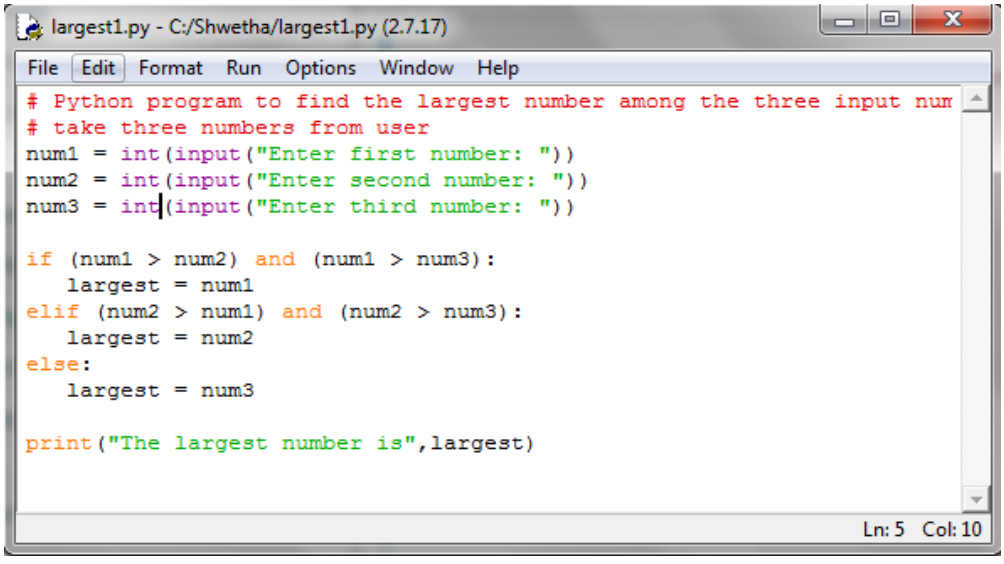
Take the three numbers as input. Check the first number if it greater than other two. Repeat the checking for other two numbers. Print the number which is greater among all and exit.

Algorithm:

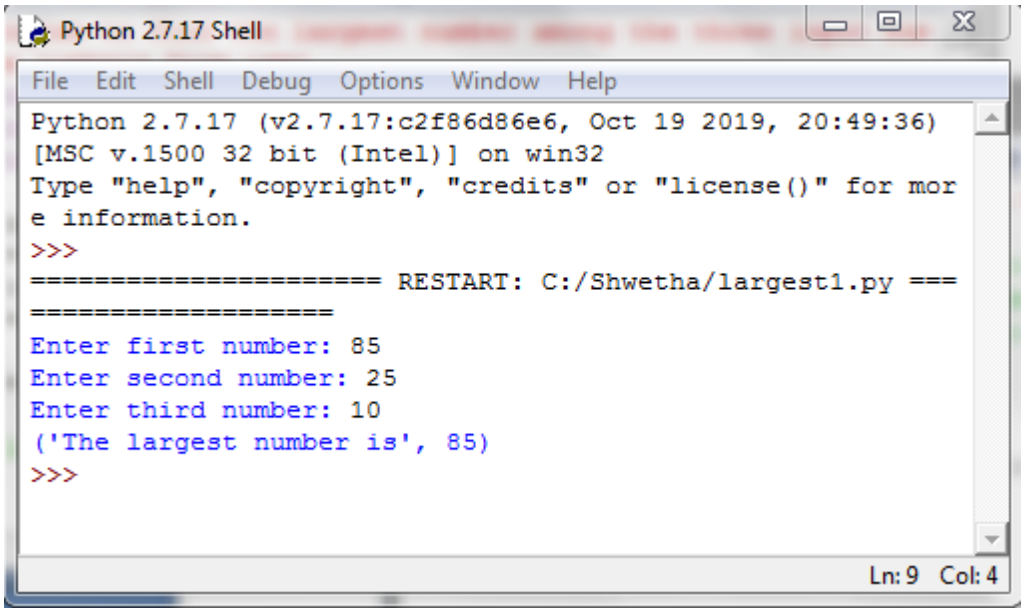
Input: Read 3 integers
Output: print largest number

```
Step 1. Start
Step 2. Read the three numbers to be compared, as A, B and C.
Step 3. Check if A is greater than B.

   If true, then check if A is greater than C.
   If true, print 'A' as the greatest number.
   If false, print 'C' as the greatest number.

   If false, then check if B is greater than C.
   If true, print 'B' as the greatest number.
   If false, print 'C' as the greatest number.
Step 4. End
```

Program:



Output:


Python 2.7.17 Shell
File Edit Shell Debug Options Window Help
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 20:49:36)
[MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for mor
e information.
>>>
========== RESTART: C:/Shwetha/largest1.py ===
==========
Enter first number: 85
Enter second number: 25
Enter third number: 10
('The largest number is', 85)
>>>
Ln: 9 Col: 4

WEEK 4: Control Flow: Loops

Loop or Iterative statements are decision control statements that are used to repeat the execution of a list of statements. In Python, there are two types of loops that are supported.

- while loop
- for loop

Syntax:

while loop :	**for loop :**
statement x	**for** variable **in range**(initial **range** value, terminate **range** value)
while(test condition):	statement
statement	**or**
statement y	
	for variable **in range**(beg, end, step)
	statement

4 a) Write a python program to print multiples of a entered number for entered number of times.

Algorithm:

Input: An integer

Output: Multiples of an entered integer

Step1: Start

Step2: Enter a number

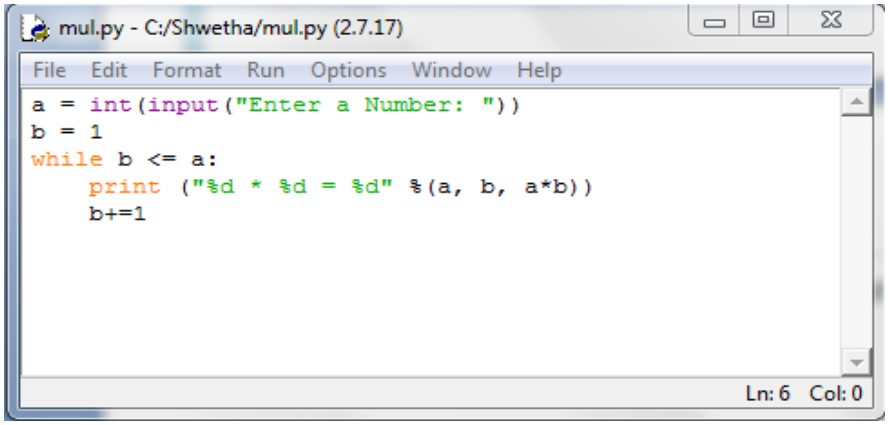
Step3: Initialize variable b = 1

Step4: Check condition b <= a is true goto step5, when condition is false goto step 6

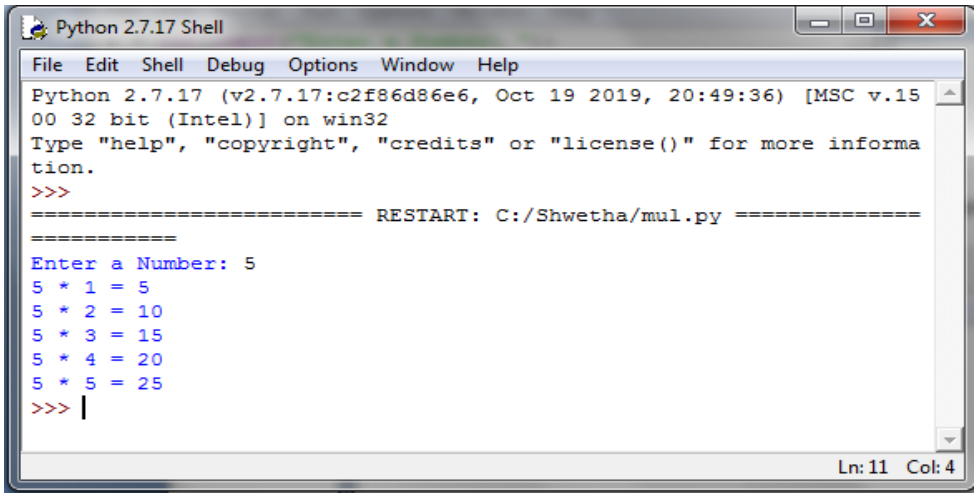
Step5: Print value and increment b by 1

Step6: stop

Program:



Output:



4 b) Write a Python program to count the number of even and odd numbers from a series of numbers.

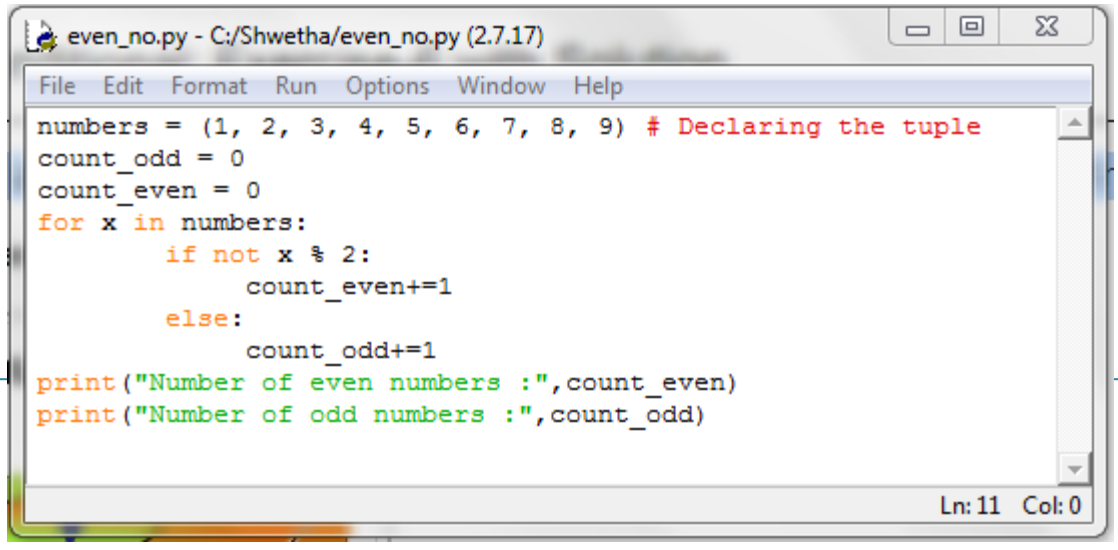
Algorithm:

Input: Declared as tuple in code

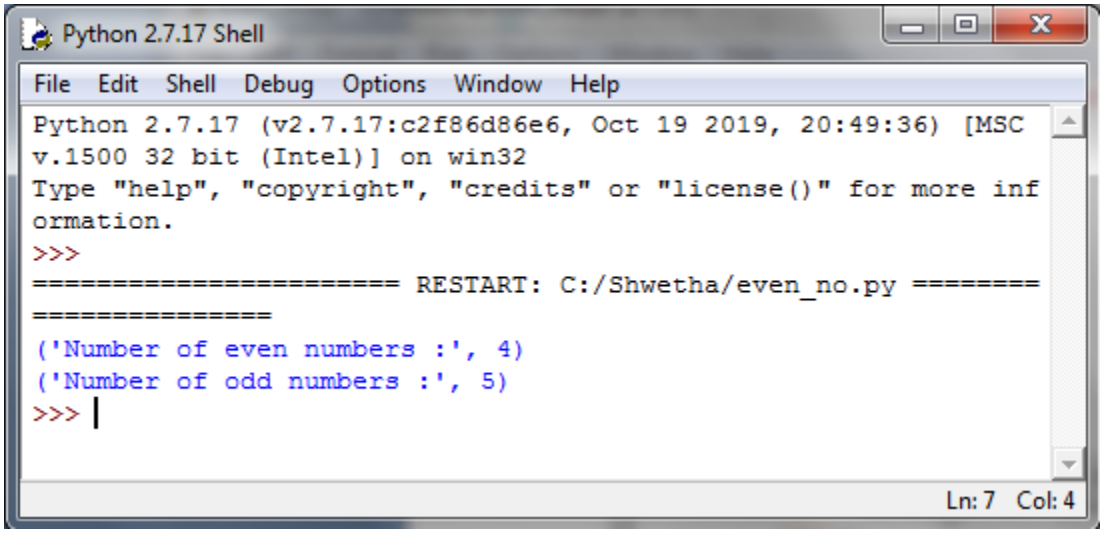
Output: print count of even and odd numbers

Step 1: start
Step2: Declare a tuple between 1 to 9
Step3: initialize count_odd = 0 and count_even = 0
Step4: For every tuple, check if not x mod 2, increment count_even by 1 else count_odd by 1.
Step 5: Print even numbers
Step 6: Print odd numbers
Step 7: Stop

Python Code:



Output:



- Program for analysis of people of certain age groups who are eligible for getting a suitable job if their condition and norms get satisfied using nested if statement.
- To attract customers AB shopping mall has announced various offers as:

  Shop for 999 and get 10% off

  Shop for 2999 and get 15% off

  Shop for 5999 and above get 40% off.

  Design an algorithm and implement the same to help AB shopping mall to calculate exact discounts based on customer bill.

Week 5: Sets

add(x) Method: It adds the item x to a set if it is non-preexisting.

Operators for Sets

Sets and frozen sets support the following operators -

```
key in s          # containment check
key not in s    # non-containment check
s1 == s2        # s1 is equivalent to s2
s1 != s2        # s1 is not equivalent to s2
s1 <= s2    # s1is subset of s2 s1 < s2     # s1 is proper subset of s2 s1 >= s2
s1 > s2     # s1 is proper superset of s2
s1 | s2         # the union of s1 and s2
s1 & s2         # the intersection of s1 and s2
s1 - s2         # the set of elements in s1 but not s2
s1 ^ s2         # the set of elements in precisely one of s1 or s2
```

In Python, below quick operands can be used for different operations.

| for union.

& for intersection.

– for difference

^ for symmetric difference

5 a) Program to perform different set operations

Input :

A = {0, 2, 4, 6, 8}

B = {1, 2, 3, 4, 5}

Output :

Union: [0, 1, 2, 3, 4, 5, 6, 8]

Intersection: [2, 4]

Difference: [8, 0, 6]

Symmetric difference: [0, 1, 3, 5, 6, 8]

Algorithm:

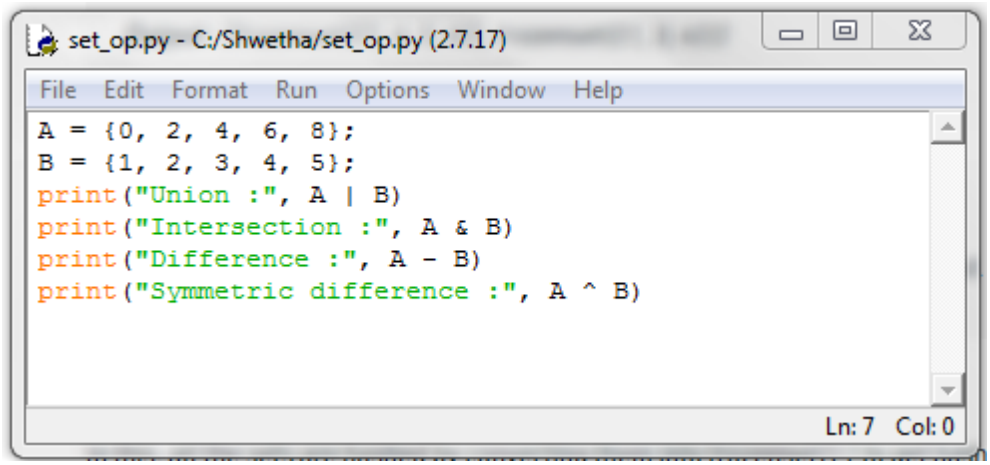
Step1: Start

Step 2: Intialize set A and B

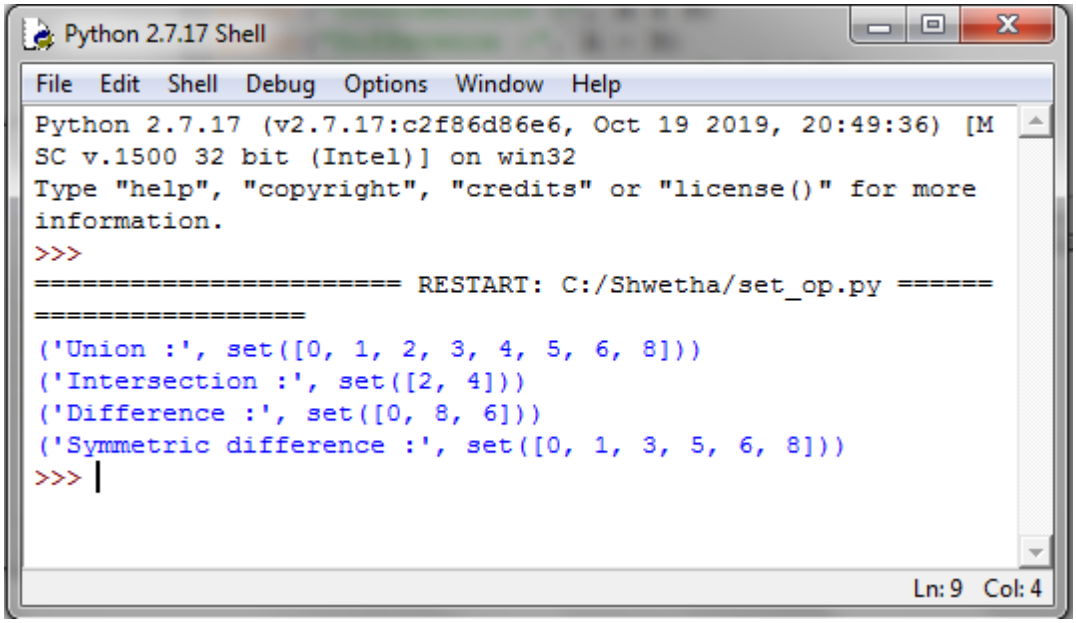
Step 3 : Print union, intersection, difference and symmetric difference

Step 4: End

Code:



Ouput:



5b) Program to create a new set with squares of element of an existing set.

Input: myset = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

Output: The newly created set using set comprehension.

Algorithm:

Step1: Start

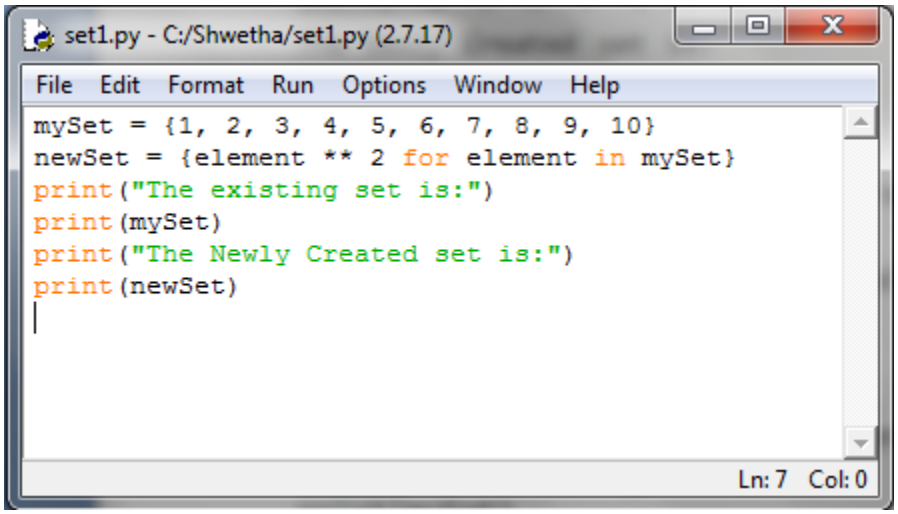
Step 2: Declare myset = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

Step 3: Declare newset, for each value in myset find the exponential by 2

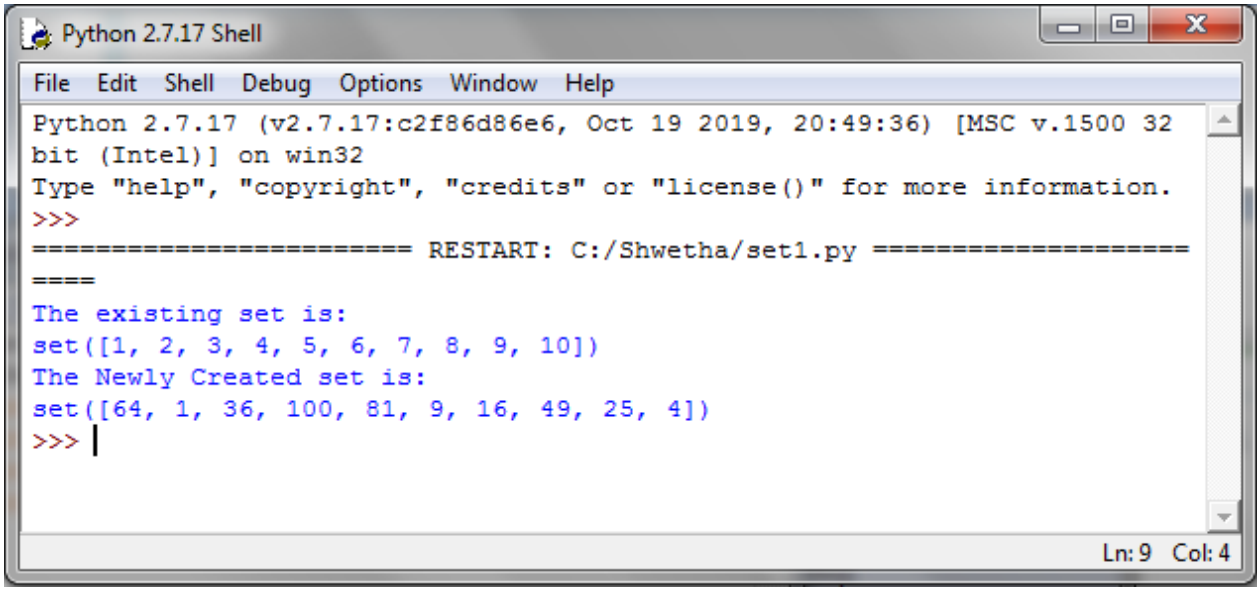
Step 4: Print myset and newest

Step 5: Stop

Code:



Output:



5 c) Program to perform basic tuple operations.

Basic Tuple Operations

Operation	Expression	Output
Length – returns the length of the tuple	tup=(1,2,3,4,5) print(len(tup))	5
Concatenation – combines two tuples and store it in another tuple	tup1=(1,2,3,4,5) tup2=(6,7,8,9,10) tup3=tup1+tup2 print(tup3)	(1,2,3,4,5,6,7,8,9,10)
Repetition – repeat the elements of the tuple	tup=(1,2,3) print(tup*2)	(1,2,3,1,2,3)
Membership (in)	tup=(1,2,3,4,5) print(3 in tup)	True
Maximum (max) – returns the maximum element in the tuple	tup=(1,2,3,4,5) print(max(tup))	5
Minimum (min) – returns the minimum element in the	tup=(1,2,3,4,5) print(tup)	1

Input: Any 2 set of tuples

Output: Operations output.

Algorithm:

Step 1: Start

Step 2: Declare any two tuples

Step 3: Find the length of tuple and print it.

Step 4: Concatenate two tuples and print it

Step 5: Print tuple2 5 times

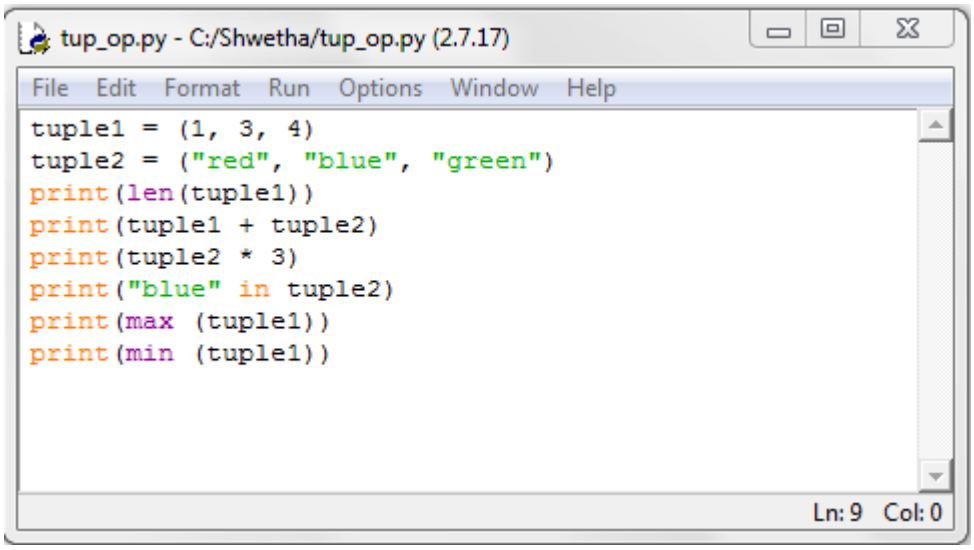
Step 6: Check whether a value is present in tuple and print it.

Step 7: Find the maximum value of a tuple and print it.

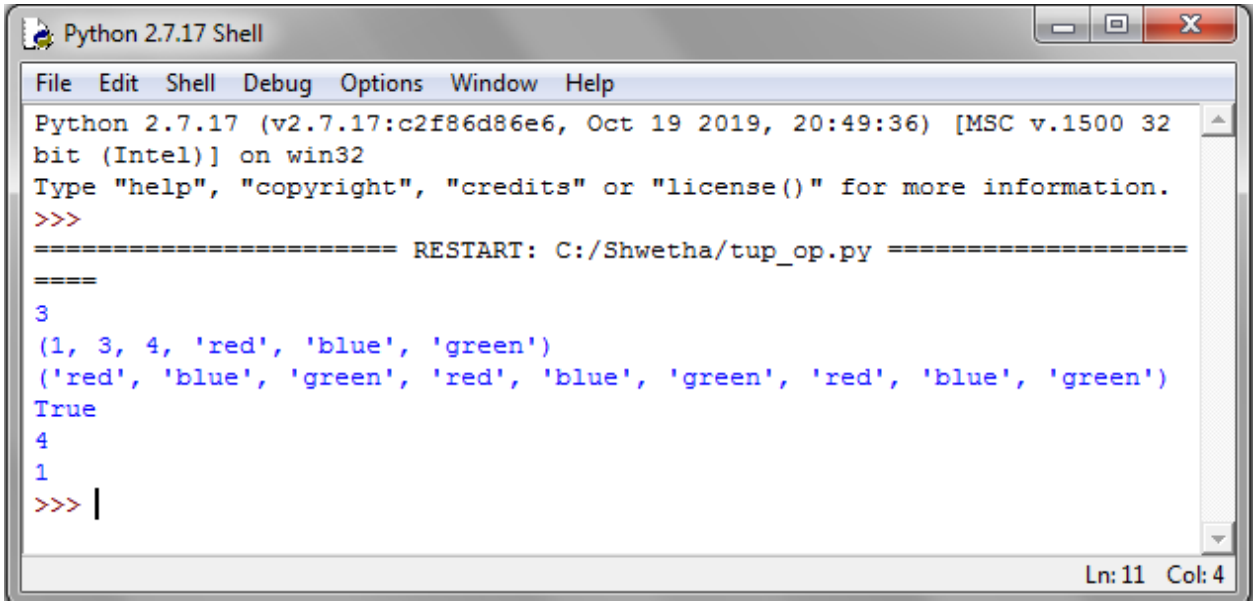
Step 8: Find the minimum value of a tuple and print it.

Step 9: Stop

Code:



Output:



5 d) Code to represent indexing and slicing from tuple.

Input: Tuple

Output: Indexing and slicing from tuple declared.

Algorithm:

Step 1: Start
Step 2: Declare a tuple.
Step 3: Print original list using t[:]
Step4: print Element at index 0 using t[0]
Step 5: print Element at index 4 using t[4]
Step 6:Print Last element of the tuple using t[-1]
Step 7:Print Elements from 2nd to 5th index using t[1:6]
Step 8: Print Elements from beginning to 4th index using t[:-3]
Step 9: Print alternate elements from tuple, from index 0 to last using t[::2]
Step 10:Print tuple in reverse order using t[: :-1]
Step 11: Stop

```
1  t = (10, 8, 5, 2, 10, 15, 10, 8, 5, 8, 8, 2)
2  print("Original list",t[:])
3  print("Indexing:")
4  print("Element at index 0", t[0])
5  print("Element at index 4", t[4])
6  print("Last element of the tuple", t[-1])
7  print("Slicing:")
8  print("Elements from 2nd to 5th is:",t[1:6])
9  print("Elements beginning to 4th is:",t[:-3])
10 print("Alternate elements from tuple",t[::2])
11 print("tuple in reverse",t[::-1])
12
13
```

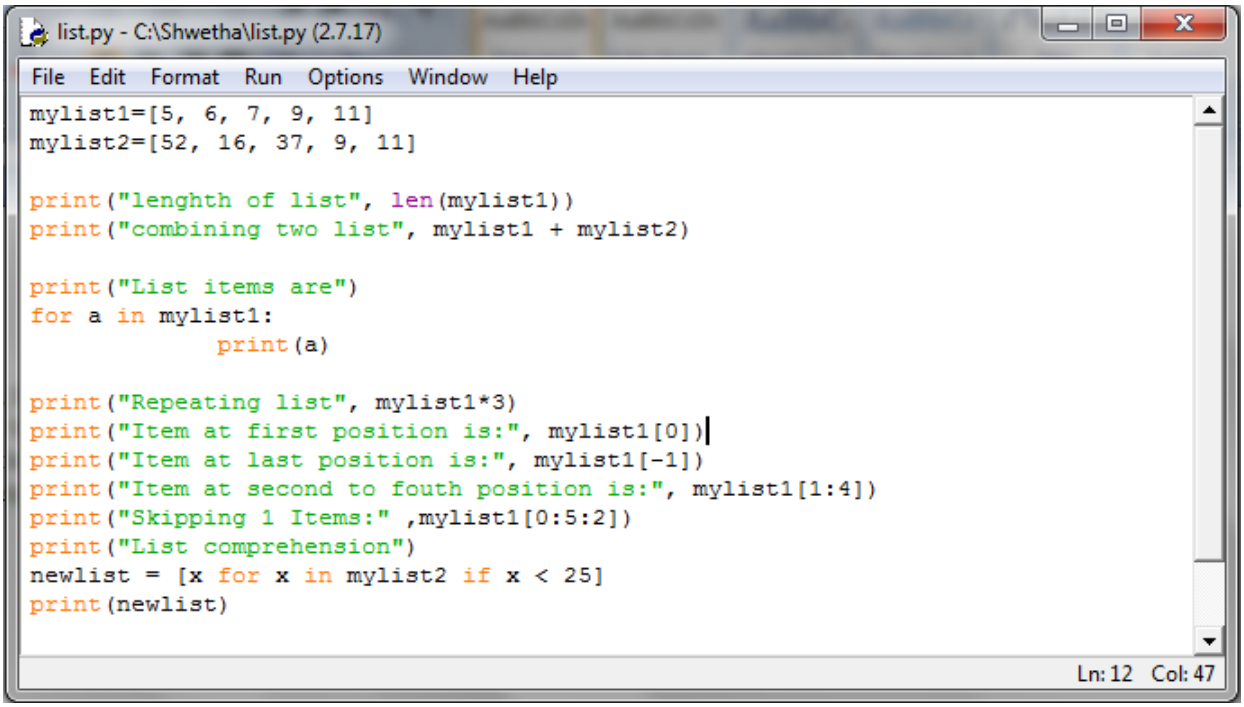
```
Original list (10, 8, 5, 2, 10, 15, 10, 8, 5, 8, 8, 2)
Indexing:
Element at index 0 10
Element at index 4 10
Last element of the tuple 2
Slicing:
Elements from 2nd to 5th is: (8, 5, 2, 10, 15)
Elements beginning to 4th is: (10, 8, 5, 2, 10, 15, 10, 8, 5)
Alternate elements from tuple (10, 5, 10, 10, 5, 8)
tuple in reverse (2, 8, 8, 5, 8, 10, 15, 10, 2, 5, 8, 10)
```

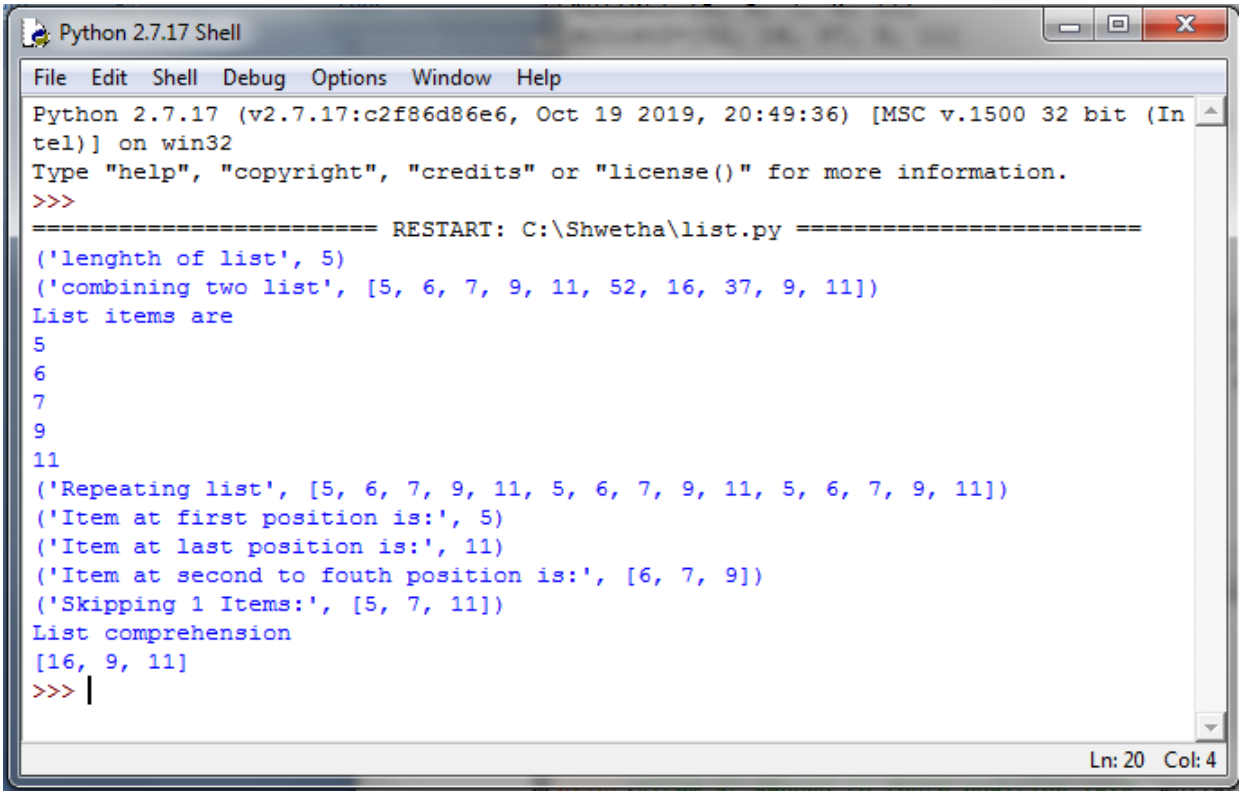
Week -6: List

6 Write code snippet to perform following on List

- Basic operations on List
- indexing and slicing
- Comprehension

Input: Any 2 set of list
Output: Operations on list.
Algorithm:
Step 1: Start
Step 2: Declare any two list
Step 3: Find the length of list and print it.
Step 4: Concatenate two lists and print it
Step 5: Iterate list and print it
Step 6: Print list2 3 times
Step 7: Print first and last item in the last.
Step 8: Slice the list and print it.
Step 9: Print list using list Comprehension.
Step 10: Stop




Python 2.7.17 Shell
File Edit Shell Debug Options Window Help
Python 2.7.17 (v2.7.17:c2f86d86e6, Oct 19 2019, 20:49:36) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Shwetha\list.py ==========
('lenghth of list', 5)
('combining two list', [5, 6, 7, 9, 11, 52, 16, 37, 9, 11])
List items are
5
6
7
9
11
('Repeating list', [5, 6, 7, 9, 11, 5, 6, 7, 9, 11, 5, 6, 7, 9, 11])
('Item at first position is:', 5)
('Item at last position is:', 11)
('Item at second to fouth position is:', [6, 7, 9])
('Skipping 1 Items:', [5, 7, 11])
List comprehension
[16, 9, 11]
>>>
Ln: 20 Col: 4

Week- 7 Dictionary

7 Code, execute and debug programs to perform basic operations on Dictionary.

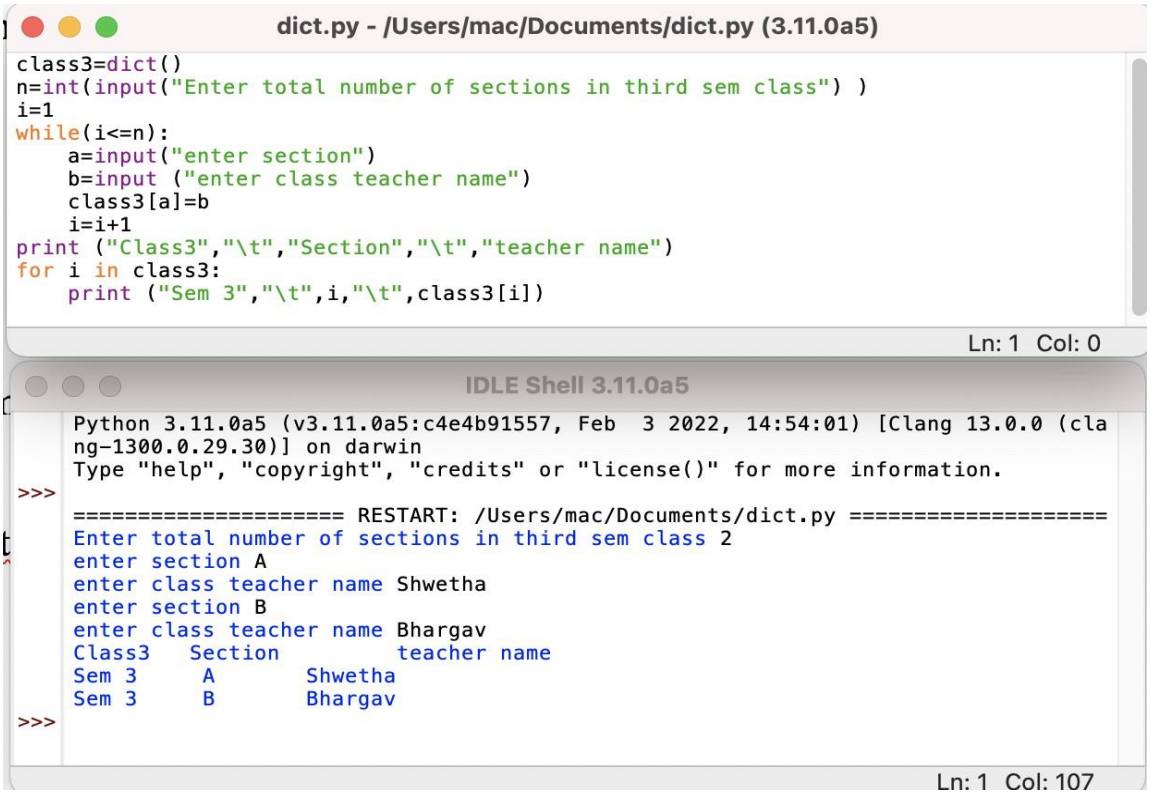
7 a) Write a program to input total number of sections and class teachers' name in third semester class and display all information on the output screen.

Input: create a dictionary during run time

Output: basic operations on Dictionary

Algorithm:

Step 1: start
Step 2: create an empty dictionary using dict( )
Step 3: Read n
Step 4: initialize i = 1
Step 5: Check i<=n, true,
Read section and store in **a**, read class teacher name and store in **b**,
Store in dictionary
Increment i by 1
False, go to step 6
Step 6: print dictionary using iteration
Step 7:stop



7b) Code, execute and debug programs to perform Dictionary indexing Iterating comprehension

Input: 'Australia' : 100, 'Germany' : 101, 'Europe' : 102, 'Switzerland' : 103

Output: perform Dictionary indexing Iterating comprehension

Algorithm:

Step 1: start

Step 2: declare an dictionary

Step 3: print dictionary

Step 4: print index value of europe

Step 5: print sorted dictionary using iteration

Step 6: print dictionary using comprehension

Step 7:stop

DICT2.py - /Users/mac/Documents/DICT2.py (3.11.0a5)

```
my_dict = {'Australia' : 100, 'Germany' : 101, 'Europe' : 102, 'Switzerland' : 103}
print("dictionary is : " + str(my_dict))
print("index value of europe is:",my_dict["Europe"])
print("Iterating Through Dictionary")
for key in sorted(my_dict):
    print(key, '->', my_dict[key])
print("Dictionary using comphrehension")

find_dict = {k: v for (k, v) in my_dict.items() if(k=="Germany")}
print(find_dict)
```

Ln: 4 Col: 37

IDLE Shell 3.11.0a5

```
Python 3.11.0a5 (v3.11.0a5:c4e4b91557, Feb  3 2022, 14:54:01) [Clang 13.0.0 (clang-13
00.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /Users/mac/Documents/DICT2.py ==========
dictionary is : {'Australia': 100, 'Germany': 101, 'Europe': 102, 'Switzerland': 103}
index value of europe is: 102
Iterating Through Dictionary
Australia -> 100
Europe -> 102
Germany -> 101
Switzerland -> 103
Dictionary using comphrehension
{'Germany': 101}
>>>
```

Ln: 14 Col: 0

Write a python program to input ‘n’ names and phone numbers to store it in a dictionary and to input any name and to print the phone number of that particular name.

Week 8:- Arrays and Strings

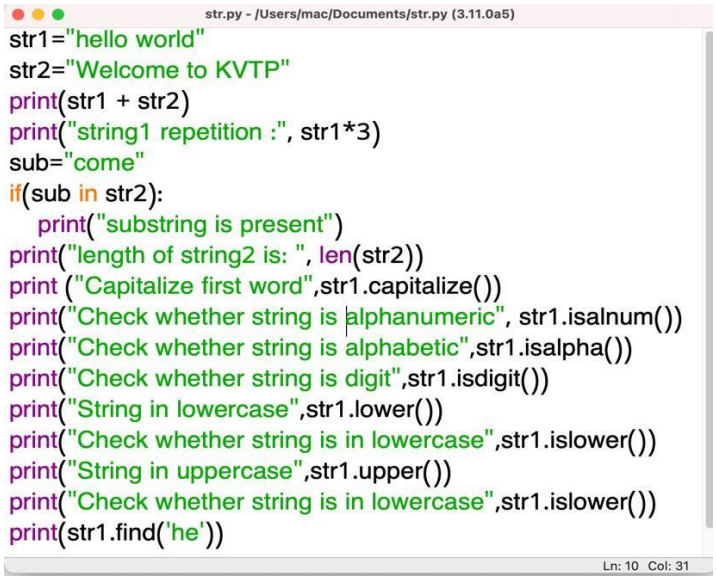
8a . Code, execute and debug programs to perform string manipulation

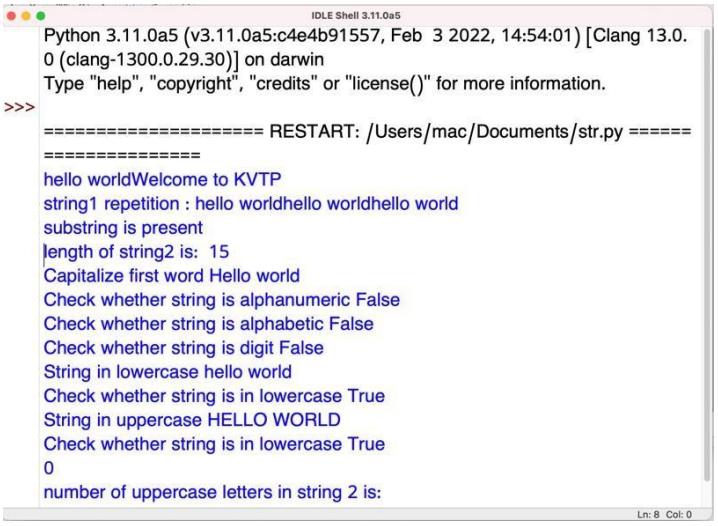
Input: Any two strings

Output: String manipulation

Algorithm:

Step1: start
Step2: declare 2 strings
Step3: print concatenated string
Step4: print string repetition
Step 5: check whether a substring 'come' is present in string2
Step 6: print length of string2
Step 7: Capitalize first word of string1and print
Step 8: Check whether string is alphanumeric and print
Step 9: Check whether string is alphabetic and print
Step 10: Check whether string is digit and print
Step 11: Print String in lowercase
Step 12: Check whether string is in lowercase and print
Step 13: Print String in uppercase
Step 14: Check whether string is in lowercase and print
Step 15: print the index of string
Step 16:stop





8 b) Code, execute and debug programs to perform array manipulation

Input: An array
Output: Array manipulation
Algorithm:
Step1: start
Step2: import array and declare an array
Step3: print new created array using iteration
Step4: Insert an item and print it
Step 5:print an array element at particular index
Step 6: Enter an index value to pop from an array, and print it
Step 7: Enter an element value to remove from an array, and print it
Step 8: print an array using slice
Step 9: stop

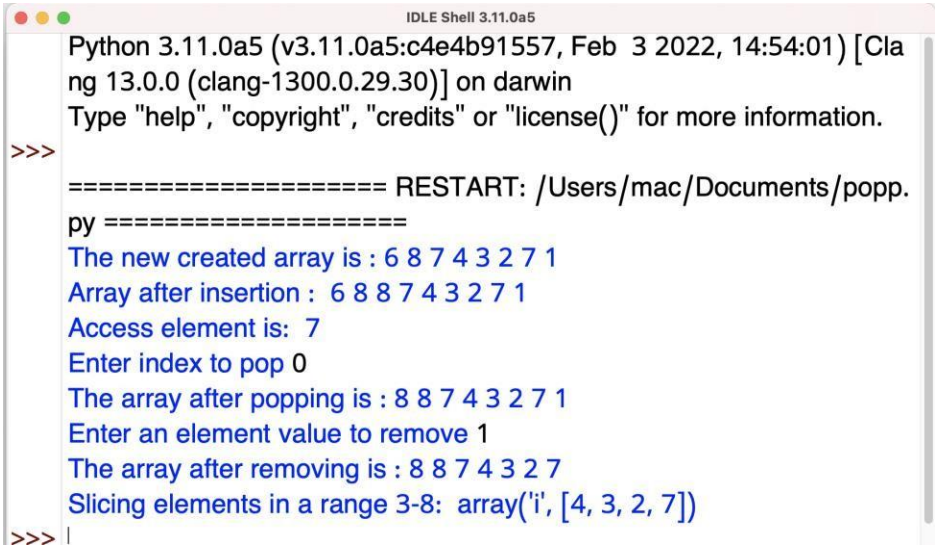
Code:

popp.py - /Users/mac/Documents/popp.py (3.11.0a5)

```
1 import array as arr
2 a = arr.array('i', [6, 8, 7, 4, 3, 2, 7, 1])
3 print ("The new created array is : ", end ="")
4 for i in range (0, 8):
5               print (a[i], end =" ")
6 a.insert(1, 8)
7 print ("\nArray after insertion : ", end =" ")
8 for i in (a):
9               print (i, end =" ")
10 print("\nAccess element is: ", a[3])
11 n=int(input("Enter index to pop"))
12 a.pop(n)
13 print ("The array after popping is : ", end ="")
14 for i in range (0, 8):
15              print (a[i], end =" ")
16 m=int(input("\nEnter an element value to remove"))
17 a.remove(m)
18 print ("The array after removing is : ", end ="")
19 for i in range (0, 7):
20              print (a[i], end=" ")
21 print("\nSlicing elements in a range 3-8: ", a[3:8])
```

Ln: 13 Col: 6

Output:


IDLE Shell 3.11.0a5
Python 3.11.0a5 (v3.11.0a5:c4e4b91557, Feb 3 2022, 14:54:01) [Cla
ng 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /Users/mac/Documents/popp.
py ==========
The new created array is : 6 8 7 4 3 2 7 1
Array after insertion : 6 8 8 7 4 3 2 7 1
Access element is: 7
Enter index to pop 0
The array after popping is : 8 8 7 4 3 2 7 1
Enter an element value to remove 1
The array after removing is : 8 8 7 4 3 2 7
Slicing elements in a range 3-8: array('i', [4, 3, 2, 7])
>>>

Week 9- Functions

Code, execute and debug programs to solve the given problem using built in functions

Write a python code to find factorial of a number using built-in function

Input: A number
Output: Factorial
Algorithm:
Step1: Start
Step 2: Enter a value
Step 3: Initialize fact to 1
Step 4: using range in for loop, multiply fact by i
Step 5: Print fact
Step 6: Stop

```
*fact.py - /Users/mac/Documents/fact.py (3.11.0a5)*
1 n = int(input("enter a value:"))
2 fact = 1
3 for i in range(1,n+1):
4             fact = fact * i
5 print ("The factorial is : ",fact)
6
```

```
========== RE
enter a value: 5
The factorial is :  120
>>>
```

Code, execute and debug programs to solve the given problem by defining a function

Program to find whether a given number is even or odd

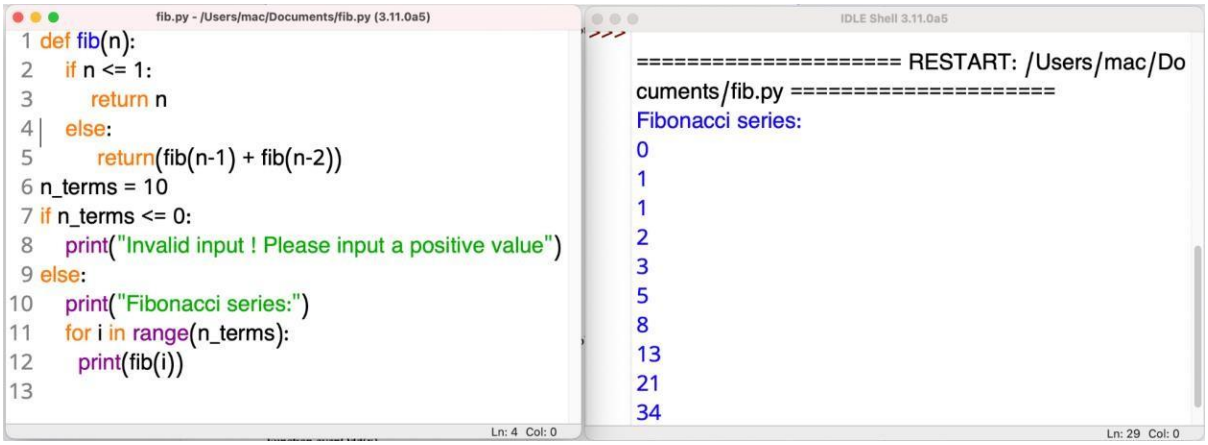
Input: A number
Output: even or odd using defined function
Algorithm:
Step1: Start
Step 2: Define a function evenodd
Check if remainder equals to 0, using mod, if true goto step 2.1 else goto step 2.2
print even
Print odd
Step 3: read a number
Step 4: call function evenOdd
Step 6: Stop

```
evenodd.py - /Users/mac/Documents/evenodd.py (3.11.0a5)
1 def evenOdd(x):
2             if (x % 2 == 0):
3                         print("even")
4             else:
5                         print("odd")
6 n=int(input("Enter number"))
7 evenOdd(n)
8
Ln: 3  Col: 8
Enter number 5
odd
```

Code, execute and debug programs to solve the given problem using recursion

Program to print the Fibonacci series upto n_terms

Input: A number
Output: even or odd using defined function
Algorithm:
Step1: Start
Step 2: Define a function fib
Check if n is less than or equal to 1, if true goto step 2.1 else goto step 2.2
return n
call recursion fib, return fib(n-1) + fib(ni2)
Step 3: initialise n_terms = 10
Step 4: check if n_terms is less than or equal to 0, if true goto step 4.1 else goto step 4.2
print invalid input
print fibonacci series
Step 6: Stop



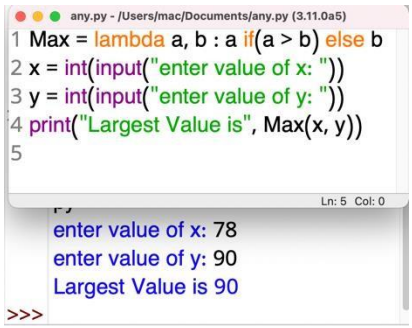
Define anonymous function and code to solve the given problem.

A lambda function is a small anonymous function.
A lambda function can take any number of arguments, but can only have one expression.

Syntax:
```
lambda arguments : expression
```
Example:
Add 10 to argument a, and return the result:
```
x = lambda a : a + 10
print(x(5))
>>> 15
```

Python to find largest of two numbers using lambda function

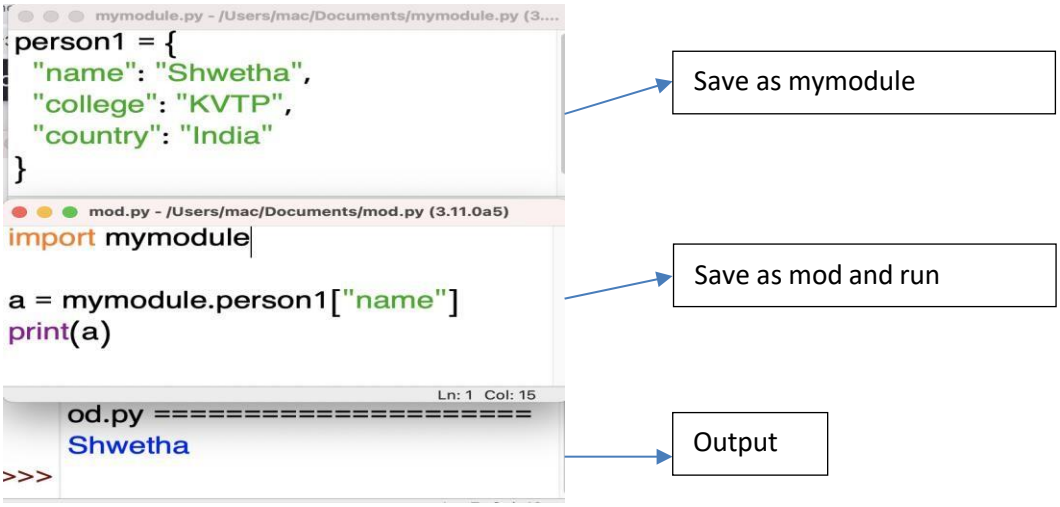
Input: Two values
Output: Largest value using lambda
Algorithm:
Step 1: Start
Step 2: Assign max = lambda a,b : a if(a>b) else b
Step 3: read x
Step 4: read y
Step 5: print Max
Step 6: Stop



Week 10 -Modules and Packages

10. Create Modules

Input: A value
Output: value from Module
Algorithm:
Step 1: Start
Step 2: create a module and save it.
Step 3: Import module
Step 4: call module value
Step 5: print value from module
Step 6: Stop



Week 11- NumPy

11 a) Code, execute and debug programs using NumPy module. (Run in online compiler)

Input: An array
Output: Array using numpy
Algorithm:
Step 1: Start
Step 2: import numpy
Step 3: Declare an array
Step 4: Print array
Step 5:Stop

```
main.py
1 import numpy as np
2 arr = np.array([1, 2, 3, 4, 5])
3 print(arr)
4 print(type(arr))
5
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Pandas

11 b) Code, execute and debug programs using series.

What is a Series?
A Pandas Series is like a column in a table.
It is a one-dimensional array holding data of any type.

Input: An array
Output: Array using pandas
Algorithm:
Step 1: Start
Step 2: import pandas
Step 3: Declare an array
Step 4: Print array
Step 5: Stop

```
main.py
1 import pandas as pd
2 a = [1, 7, 2]
3 myvar = pd.Series(a)
4 print(myvar)
5
6
0    1
1    7
2    2
dtype: int64
```

11 c) Code, execute and debug programs using dataframes.

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

Input: Two dimensional data
Output: Dataframe
Algorithm
Step 1: Start
Step 2: import pandas
Step 3: Declare a 2 dimensional array
Step 4: assign df using dataframe
Step 4: Print df
Step 5: Stop

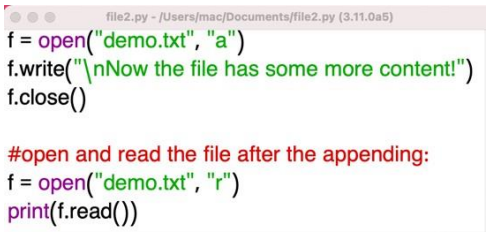
```
main.py
1 import pandas as pd
2 data = {
3   "calories": [420, 380, 390],
4   "duration": [50, 40, 45]
5 }
6 df = pd.DataFrame(data)
7 print(df)
8
   calories  duration
0       420        50
1       380        40
2       390        45
```

Week 12- Files

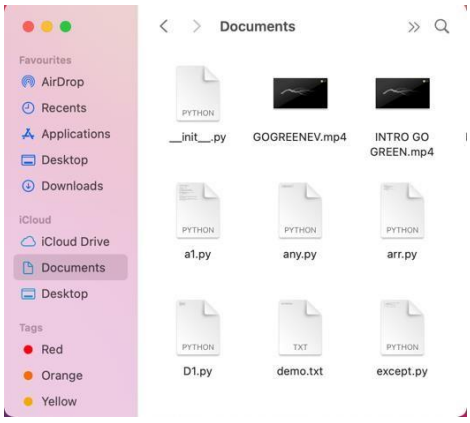
12 a) write code snippet to perform following operations on different types of files

- read file
- write to file.

"x" - Create - will create a file, returns an error if the file exist
"a" - Append - will create a file if the specified file does not exist
"w" - Write - will create a file if the specified file does not exist
“r”- read- read the content of the file
Input: Text
Output: A file is created and write data into it
Algorithm:
Step 1: Start
Step 2:open a file
Step 3: Write some text into file
Step 4: close opened file
Step 5: open file in reading mode
Step 6:read file
Step 7: Stop



A file is created where python file stored.



Week 13 Exceptions

13) Write code snippet to raise exceptions

Input: an excepted error
Output: an exception
Algorithm:
Step 1: Start
Step 2: try block to except error, print error
Step 3: except block, print error
Step 4: finally block, print it
Step 5: Stop