

**Lab Manual**

**of**

**Programming Skill II (Python)**

**Thakur College of Engineering and Technology**

**As per REVISED AUTONOMY SYLLABUS**

**OF INFORMATION TECHNOLOGY ENGINEERING**

**SYLLABUS**

**S.E. Semester –IV**

**Choice Based Credit Grading System with Holistic Development Scheme**

**(CBCGS-H 2019)**

**Proposed Syllabus under Autonomy Scheme**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BE ( Information Technology )** | | | | | | | **SEM :** IV | | | | |
| **Course Name :** Python Programming | | | | | | | **Course Code :** ITC 405 | | | | |
| **Contact Hours Per Week :** 04 | | | | | | | **Credits** : 03 | | | | |
| **Teaching Scheme (Program Specific)** | | | | | **Examination Scheme (Formative/ Summative)** | | | | | | |
| **Modes of Teaching / Learning / Weightage** | | | | | **Modes of Continuous Assessment / Evaluation** | | | | | | |
| **Hours Per Week** | | | | | **Theory**  **(100)** | | **Practical/Oral (25)** | | **Term Work (25)** | | **Total** |
| **Theory** | **Tutorial** | **Practical** | **Contact Hours** | **Credits** | **MSA** | **ESE** | **MSA** | **ESE** | **MSA** | **ESE** | **100** |
| - | - | 2+2# | 4 | 2 | - | - | 25 | 25 | 25 | 25 |
|
| **MSA:** Mid Semester Assessment- Practical Duration – **2 Hours** | | | | | | | | | | | |
| **ESE :** End Semester Evaluation- Paper Duration - **2 Hours** | | | | | | | | | | | |
| Mid Semester Assessment for Term work will be on continues basis | | | | | | | | | | | |
| **Prerequisite:** Computer Basics, Procedural Programming Languages  **RBT :** Revised Bloom’s Taxonomy | | | | | | | | | | | |

**Course Objectives:**

|  |  |  |
| --- | --- | --- |
| **SN** | **Course Objectives** | **RBT level** |
| 1 | Understand Basics of Python programming | Understand (U) |
| 2 | Express control statements and Functions | Understand (U) |
| 3 | Implement object Oriented Programming concept using Python | Apply (A) |
| 4 | Understand Errors and Exceptions in Python | Apply (A) |
| 5 | Apply Files Handling in Python | Understand (U), Apply (A) |
| 6 | Analyze and Implement GUI application using Database | Understand (U), Apply (A) |

**Course Outcomes:**

|  |  |  |
| --- | --- | --- |
| **SN** | **Course Objectives** | **RBT Levels** |
| 1 | Describe the Numbers, Math functions, Strings, List, Tuples and Dictionaries in Python | Understand (U) |
| 2 | Implement different Decision Making statements and Functions | Understand (U) |
| 3 | Interpret and apply Object oriented programming concept | Apply (A) |
| 4 | Create need based exceptions | Apply (A) |
| 5 | Understand and summarize different File handling operations | Understand (U), Apply (A) |
| 6 | Create GUI Applications in Python and evaluate different database | Understand (U), Apply (A) |

**Detailed Syllabus:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module No.** | **Topics** | **Hrs.** | **RBT Levels** | **POs** | **PSOs** | **PEOs** |
| **1** | **Basics of Python** | **6** | Understand (U) Apply (A) |  |  |  |
| Theory: Numbers in Python, Basic & Built-in Math functions, Number Formats, Strings, Quotes, print() Function, Assigning Values to Names & Changing Data Through Names, Copying Data, Tuples — Unchanging Sequences of Data, Lists — Changeable Sequences of Data, Dictionaries — Groupings of Data Indexed by Name, Special String Substitution Using Dictionaries , Arrays, Treating a String Like a List, Special Types, Ranges of Sequences, Working with Sets, Arrays | 1 to 5 | 1,2,3 | 1,2 |
| **2** | **Decision Making and Functions** | **6** | Understand (U) Apply (A) |  |  |  |
| Theory: If statement, if-elif-else, Repetition using while loop, for loop, break statement, Handling Errors- try: statement, except: statement, Functions-Grouping Code under a Name, defining a Function, describing a function in the function, Checking & Setting Your Parameters, Calling Functions from within Other Functions, Functions Inside of Functions, Layers of Functions | 1 to 5 | 1,2,3 | 1,2 |
| **3** | **OOP’s Using Python** | **6** | Understand (U) Apply (A) | 1 to 5 | 1,2,3 | 1,2 |
| Theory: Creating a Class, Self-Variables, Constructors, Types of Methods, Inner Classes, Constructors in Inheritance, Polymorphism,, The super() Method, Method Resolution Order (MRO), Operator Overloading, Method Overloading & Overriding, Interfaces in Python. | 1 to 5 | 1,2,3 | 1,2 |
| **4** | **Exception Handling and Packages** | **10** | Understand (U) Apply (A) | 1 to 5 | 1,2,3 | 1,2 |
| Exceptions Handling: Errors in a Python Program, Exceptions, Exception Handling, Types of Exceptions, The Except Block, The assert Statement. Modules and Packages: Creating Modules and Packages, Documenting & Viewing Module, Basics of Testing Your Modules and Packages, Importing & exporting Modules | 1 to 5 | 1,2,3 | 1,2 |
| **5** | **Files Handling** | **8** | Understand (U) Apply (A) | 1 to 5 | 1,2,3 | 1,2 |
| Theory: Types of Files in Python, Opening a File, Closing a File. Writing Text Files, Knowing Whether a File Exists or Not, Working with Binary Files, Appending Text to a File, Reading Text Files, File Exceptions, The with Statement Pickle in Python, Lambda and Filter, Map & range functions | 1 to 5 | 1,2,3 | 1,2 |
| **6** | **GUI Programming and Databases** | **9** | Understand (U) Apply (A) | 1 to 5 | 1,2,3 | 1,2 |
| Theory: GUI Programming - Writing a GUI with Python: GUI Programming Toolkits, Creating GUI Widgets with Tkinter, Creating Layouts, Radio Buttons and Checkboxes, Dialog Boxes. Database Access - Python’s Database Connectivity, Types of Databases Used with Python, Mysql database Connectivity with Python, Performing Insert, Deleting & Update operations on database |  | 1,2,3 | 1,2 |

**Prerequisite Subjects:** Structured Programming Approach & Java Programming

**LAB EXPERIMENT**

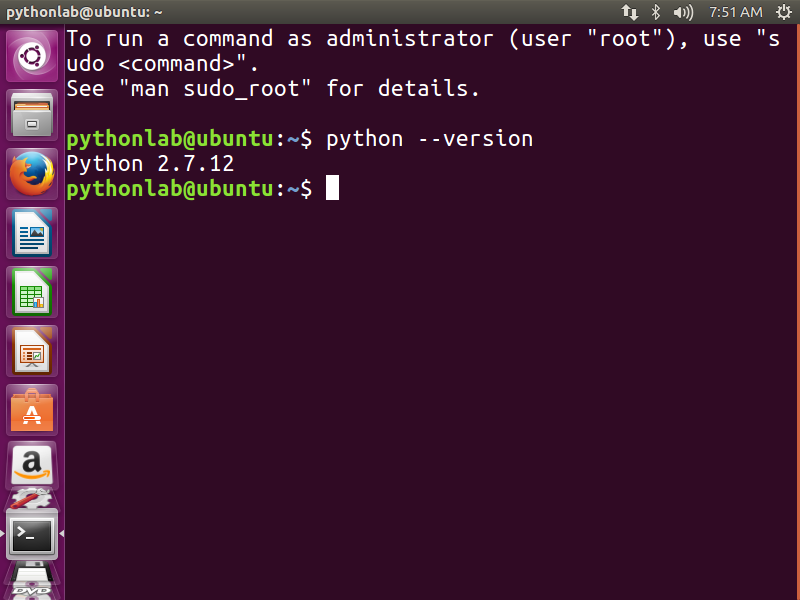
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Name of the experiment** | **Lab objective** | **Lab outcome** | **PO attained** | **PEO achieved** | **Type of Experiment** |
| 01 | a. Write a python program to implement Comments, Datatypes, Expressions, Input and Output Functions  b. Write a python program to implement Byte Array, Range, Set and STRING Functions  c. Write a python program to implement List, Tuple, Dictionaries and Arrays | LOB 1 | LO 1 | 1,11 | 1 | Problem Solving |
| 02 | Write python programs to understand Byte array, Range, Set and STRING Functions | LOB 1,4 | LO 1,4 | 1,3,4,5,10 | 1,3,4,5,6,7 | Group learning |
| 03 | Write python programs to understand List Tuple, Dictionaries and Arrays | LOB 1, 4 | LO 1, 4 | 1,3,11 | 1,5 | Problem Solving |
| 04 | WAP to understand the Control Structures  II. A Write python programs to understand Functions | LOB 1, 4 | LO 1, 4 | 1,3 | 1,5 | Problem Solving /Implementation |
| 05 | WAP to understand Classes, object, Static method and inner class | LOB 2, 5 | LO 2, 5 | 1,3 | 1,5 | Problem Solving/ Design based |
| 06 | WAP to understand different types of Exceptions | LOB 2 | LO 2 | 1,3,5 | 1,2,3,5,6 | Problem Solving /Implementation |
| 07 | WAP to understand different file handling operations with pickle | LOB 2, 4 | LO 2, 4 | 1,3, 5 | 1,2,3,5,6 | Problem Solving/ Design based |
| 08 | WAP to understand Lambda, map, reduce, filter and range functions | LOB 3 | LO 3 | 1,3,5 | 1,2,3,5,6 | Problem Solving /Implementation |
| 09 | Write python programs to understand GUI Canvas Application using Tkinter | LOB 5 | LO 5 | 1,3,5 | 1,2,3,4,5,6 | Problem Solving /Implementation |
| 10 | WAP to understand CRUD Operations using Mysql Python Database Connectivity | LOB 5 | LO 5 | 1,3,5 | 1,2,3,5,6 | Problem Solving /Implementation |
| 11 | Mini project in Python | LOB 1,2,3,4,5,6 | LO 1,2,3,4,5,6 | 1,3,4,5,10,11,12 | 1,2,3,4,5,6,7 | Mini Project |

**Steps to install python runtime environment and ides**

**--------------------------------------------------------------------------**

Ubuntu 14.04 and above versions has built-in Python but every new version of python comes with many new features and as we are going to use IDEs like Pydev and Pycharm, they require latest versions of it.

To check current Version of python run following command.



As we can see Ubuntu has preinstalled python version 2.7.12 but we want its latest version 3.6.4.so download it from python.org. 

The python 3.6.4 for Ubuntu comes with compressed file tgz. So uncompressed it using tar command

$tar –xzvf Python-3.6.4.tgz

Then go to python directory

$cd Python-3.6.4

Execute following commands in a sequence to complete the installation

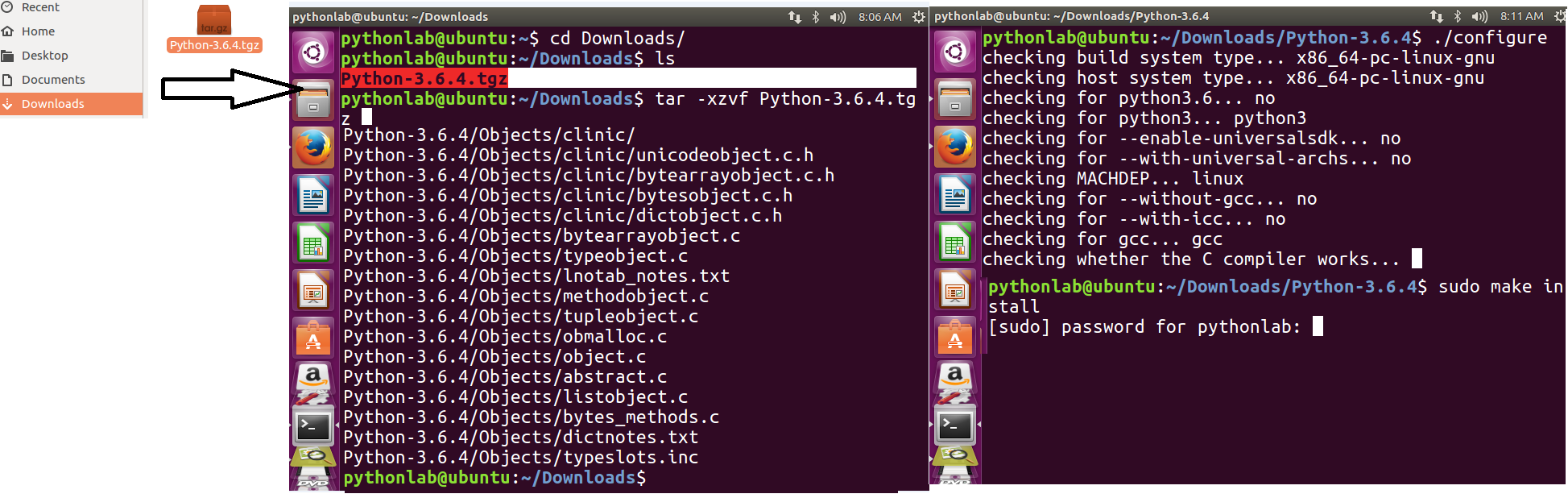
$apt-get install zlib1g-dev

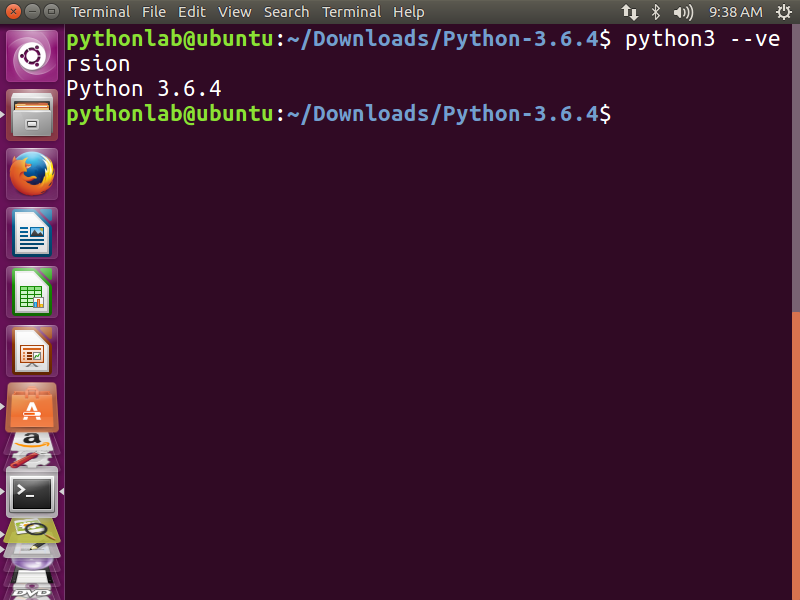
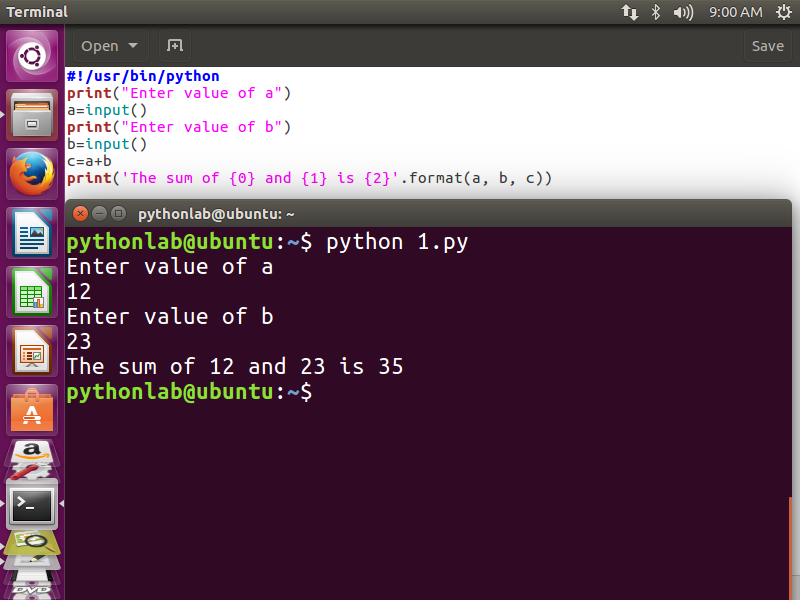
$./configure

$make

$make install

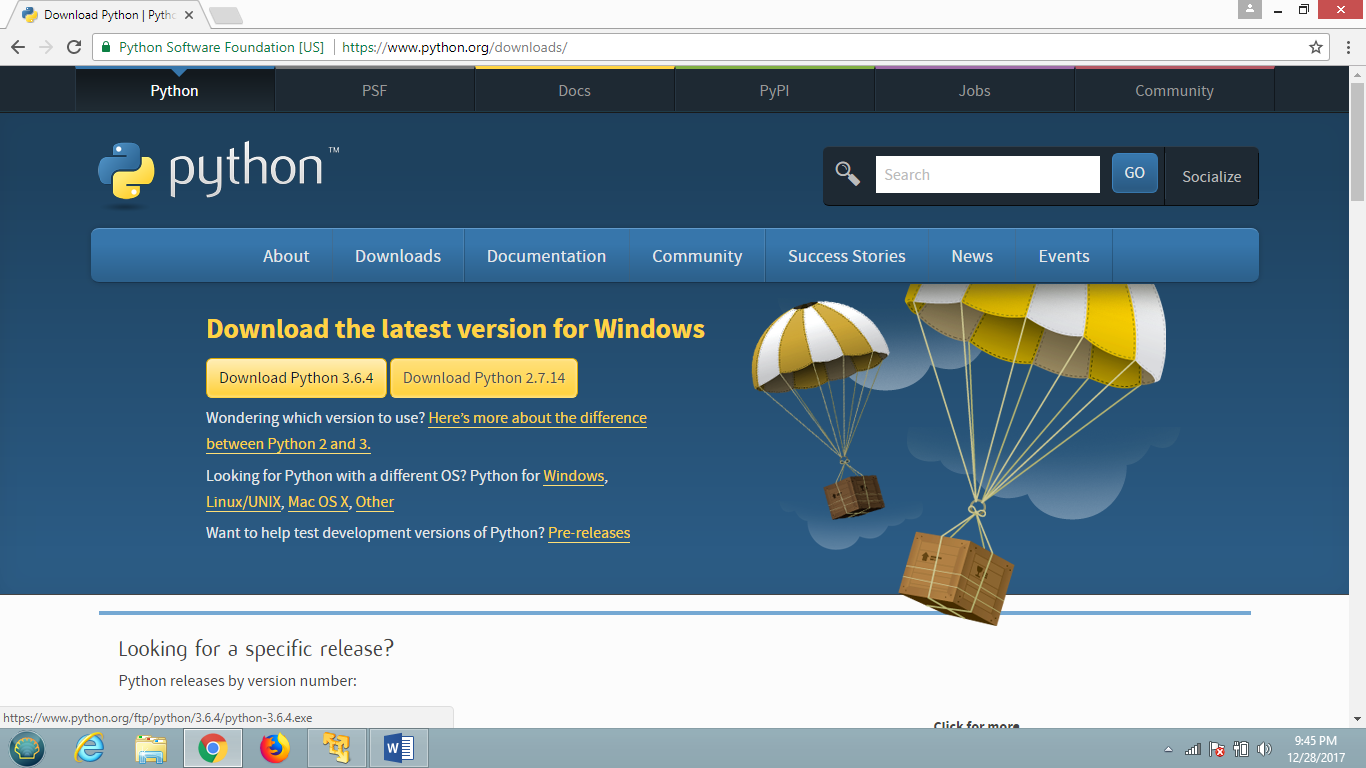
Perform the installation, check the version and run the program 1.py.



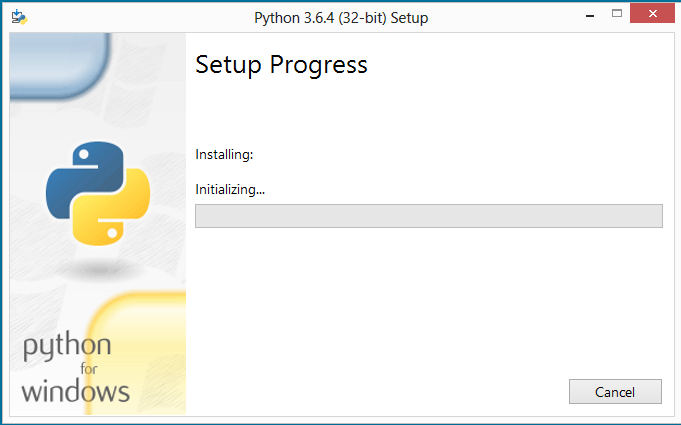
 

**Installation of Python on windows**

1. Download python 3.6.4 from python.org



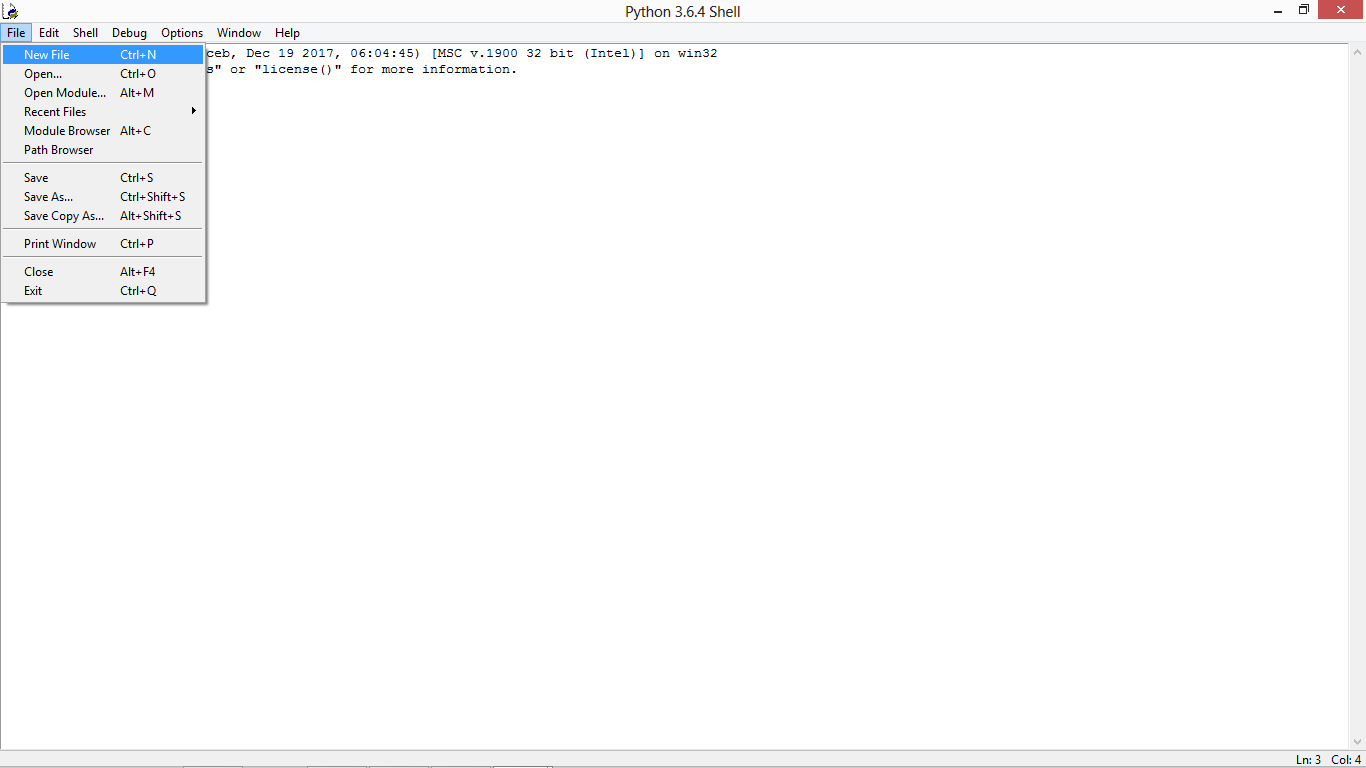
1. Install it by answering simple questions



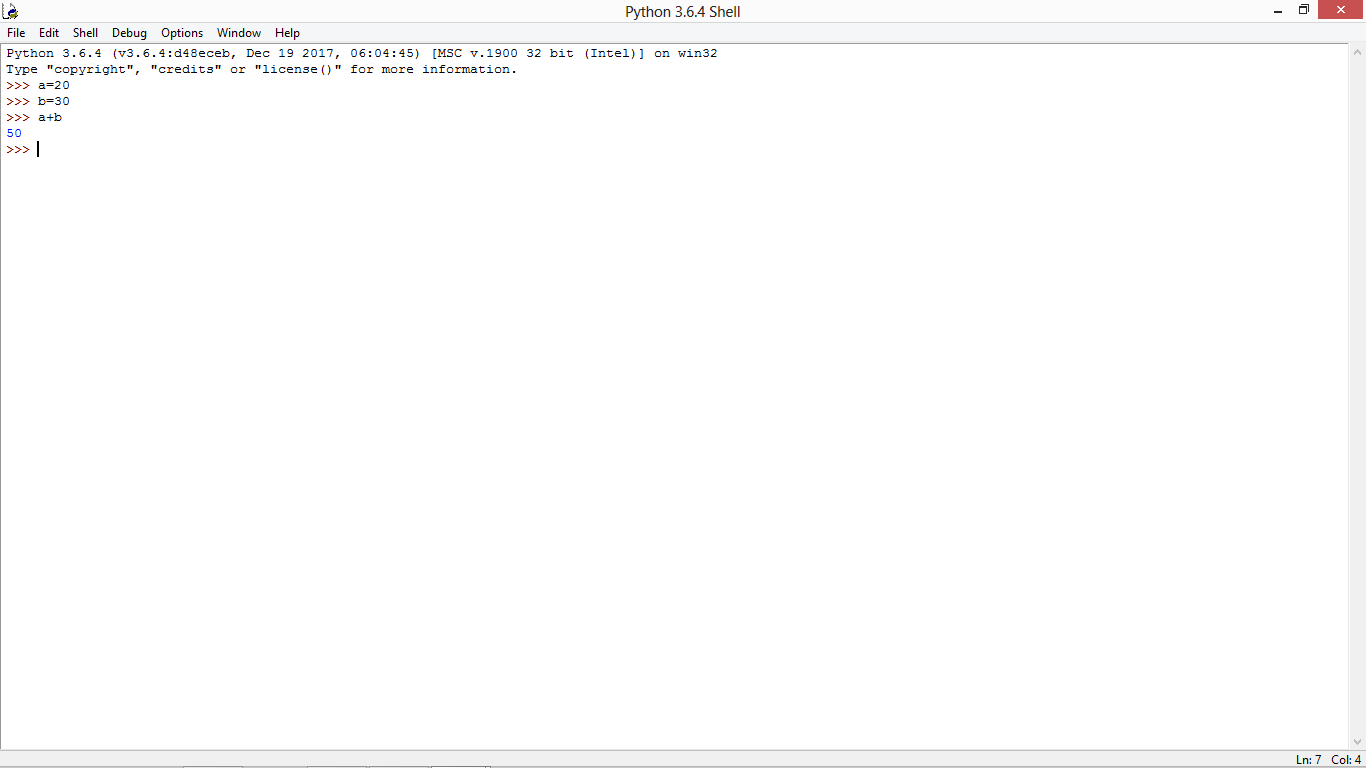
1. Once installation is done then open Python shell or IDLE (python 3.6.4 32 bit) from start menu

|  |  |
| --- | --- |
|  |  |

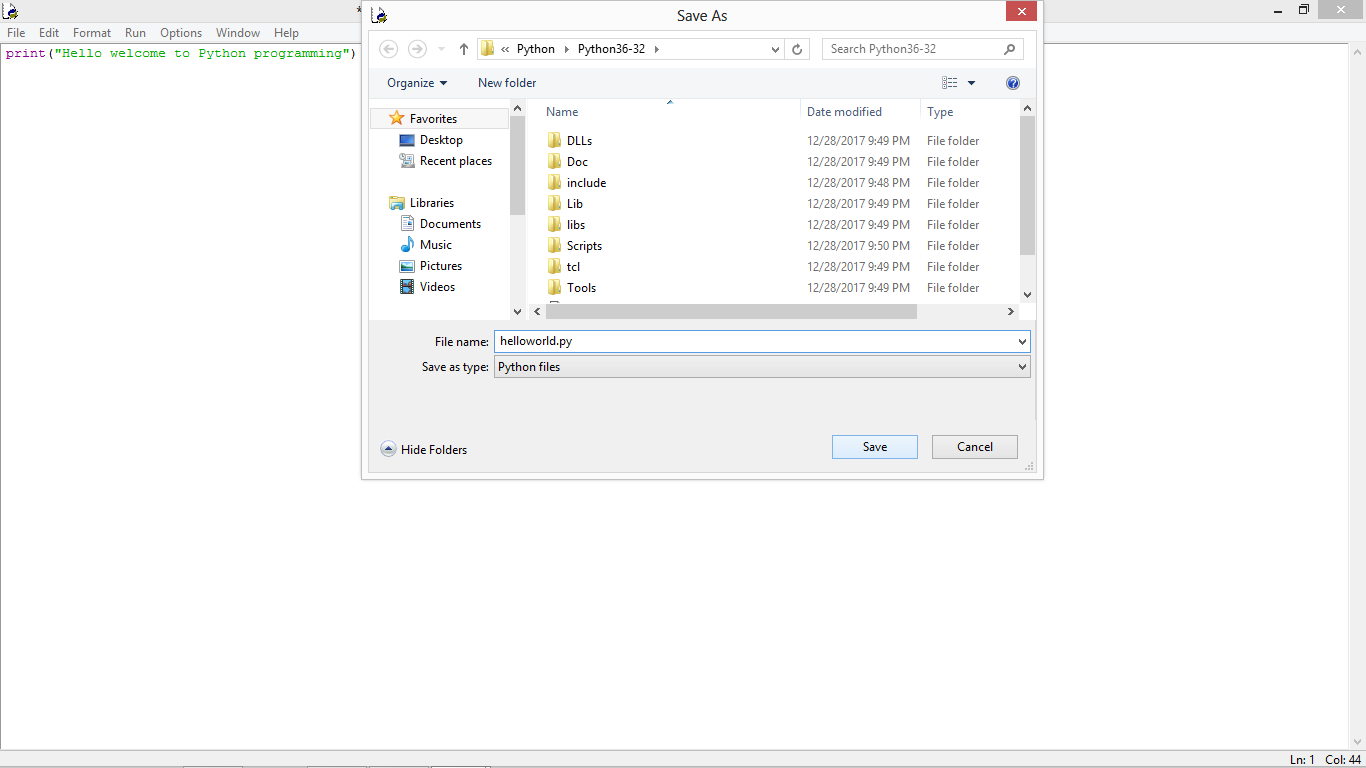
**IDLE**

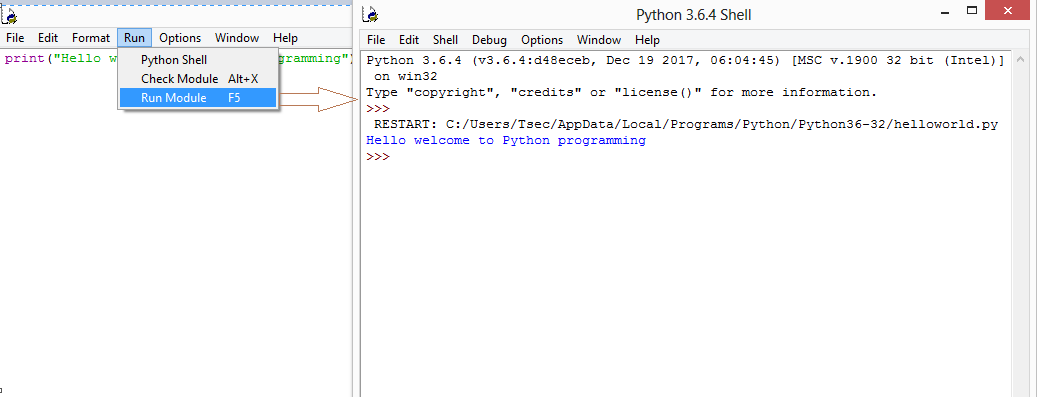


1. Test the application on python shell



On IDLE



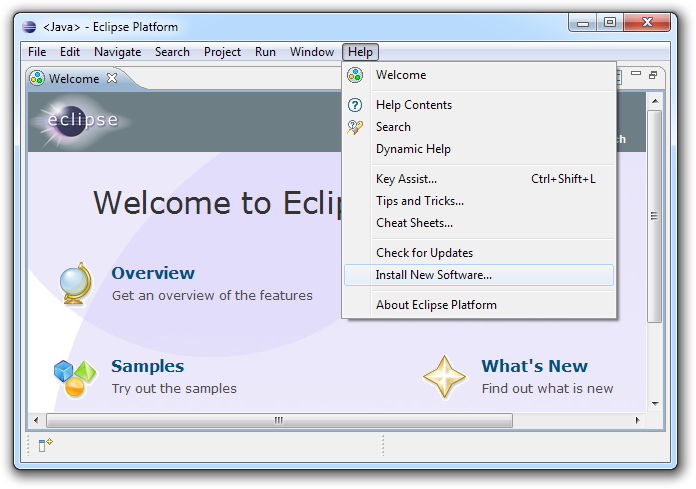


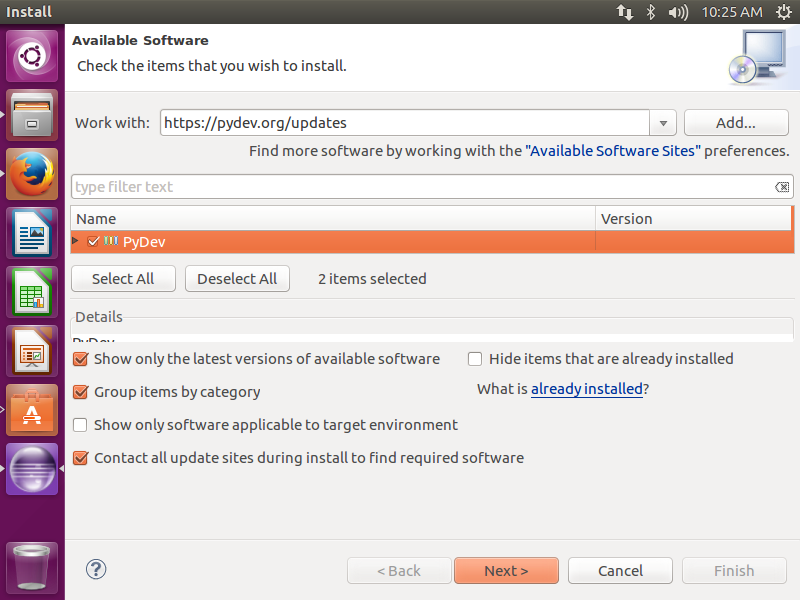
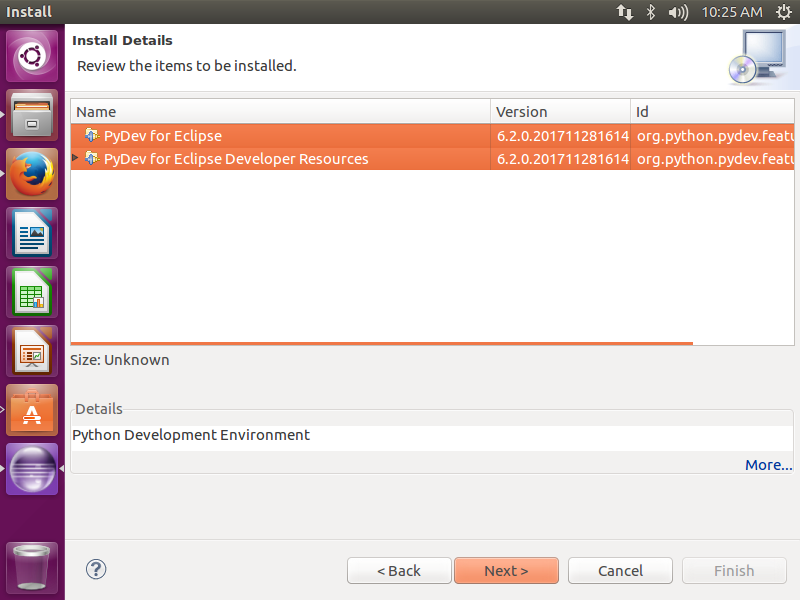
**--------------------------------------------------------------------------**

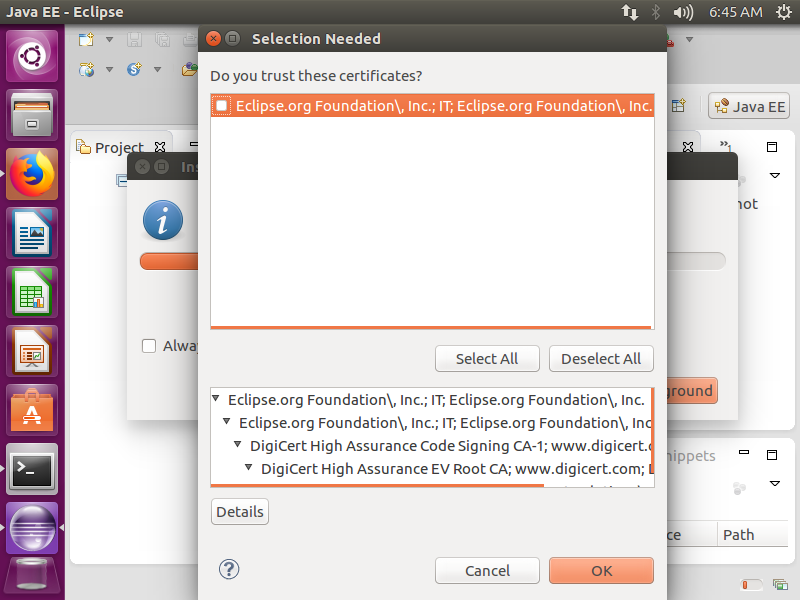
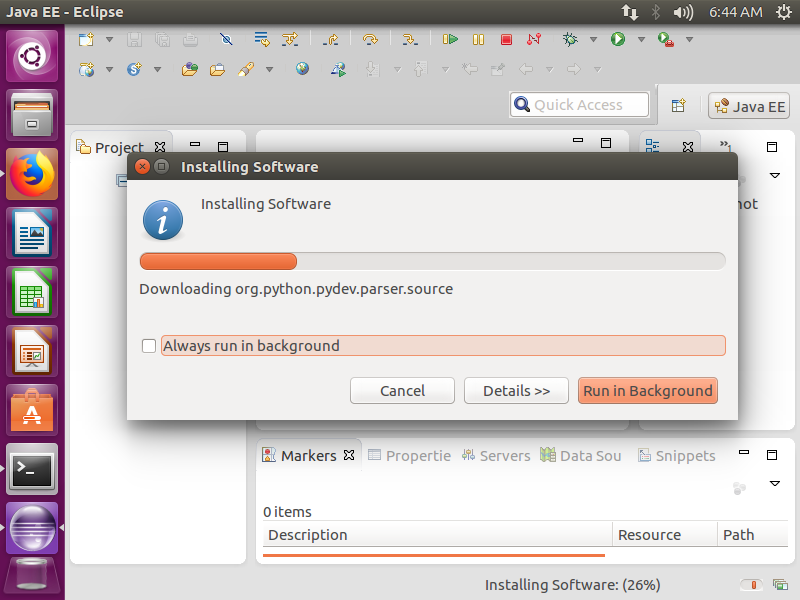
**Steps to install Pydev in eclipse**

Step 1-: Install Eclipse in Ubuntu through Software center and in windows by downloading it through google

Step 2 -: Open Eclipse 🡪 Go to install new software under help menu and type <https://pydev.org/updates> in the search box 🡪 click on add button 🡪specify name as Pydev and select Pydev

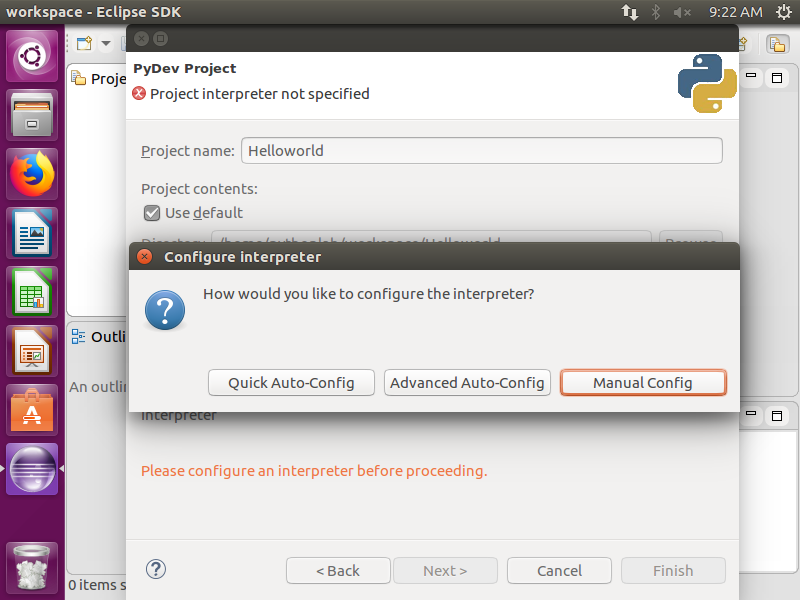
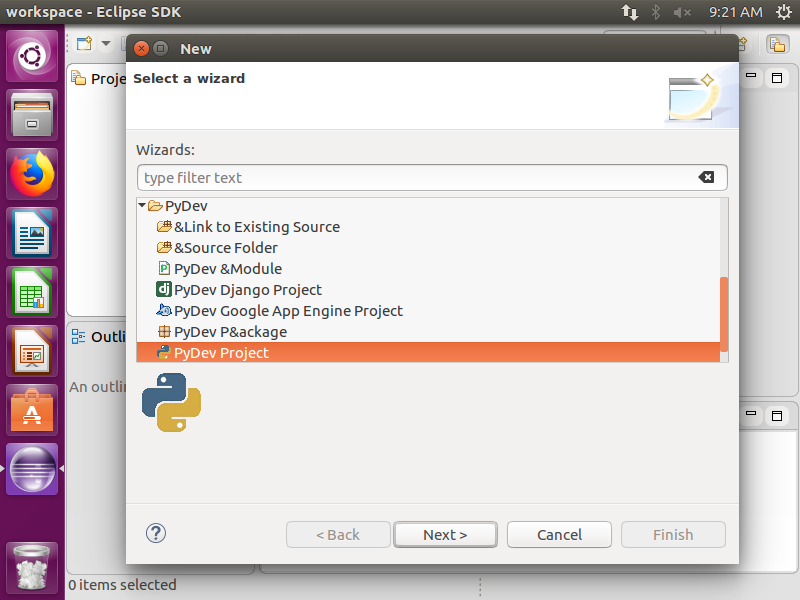


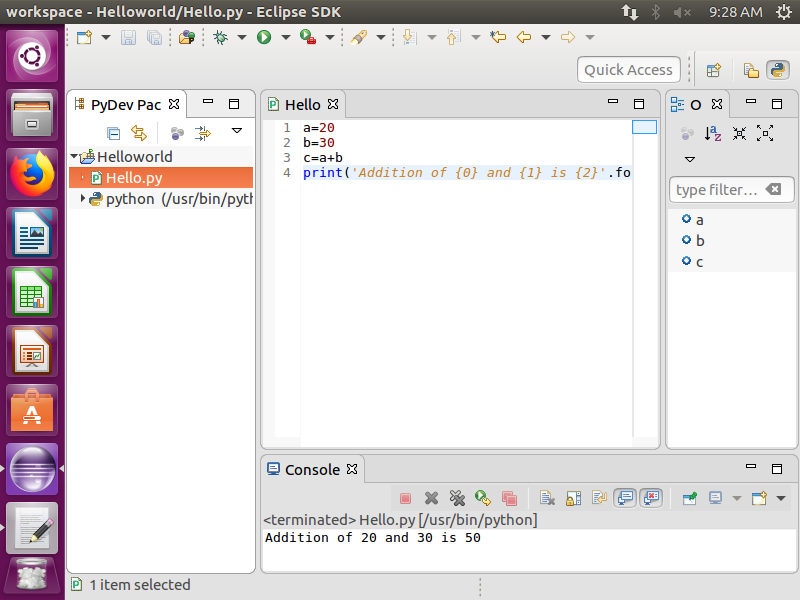
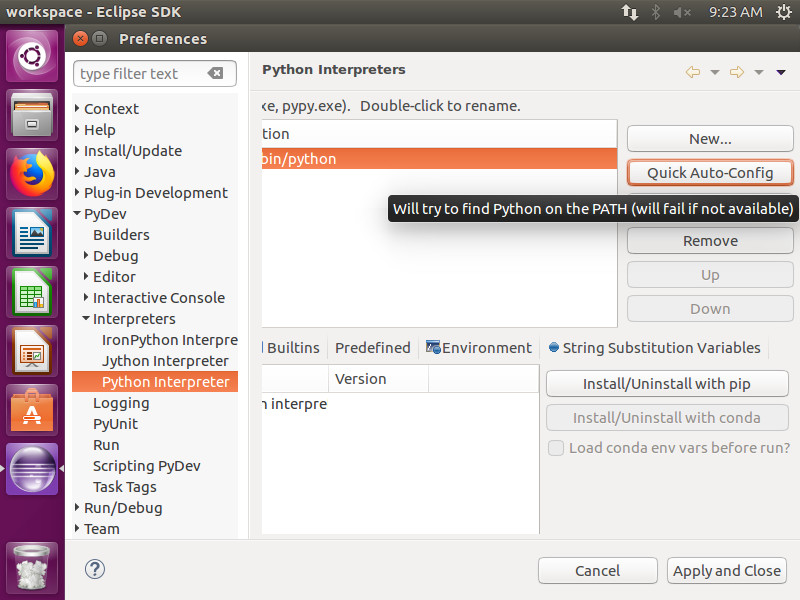
Step 3-: select pydev packages and click on next to finish the installation 



Restart the eclipse when installation is done.

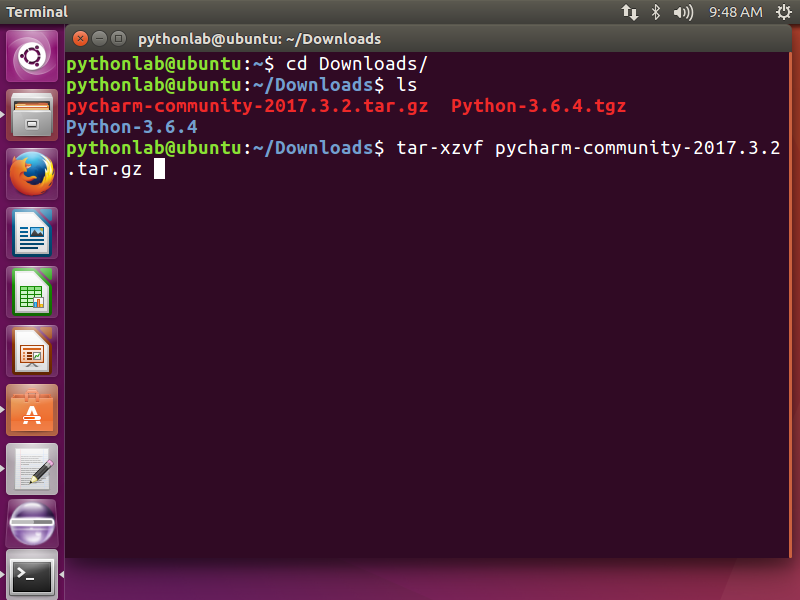
Step 4 -: Set the path to the interpreter and test the application.





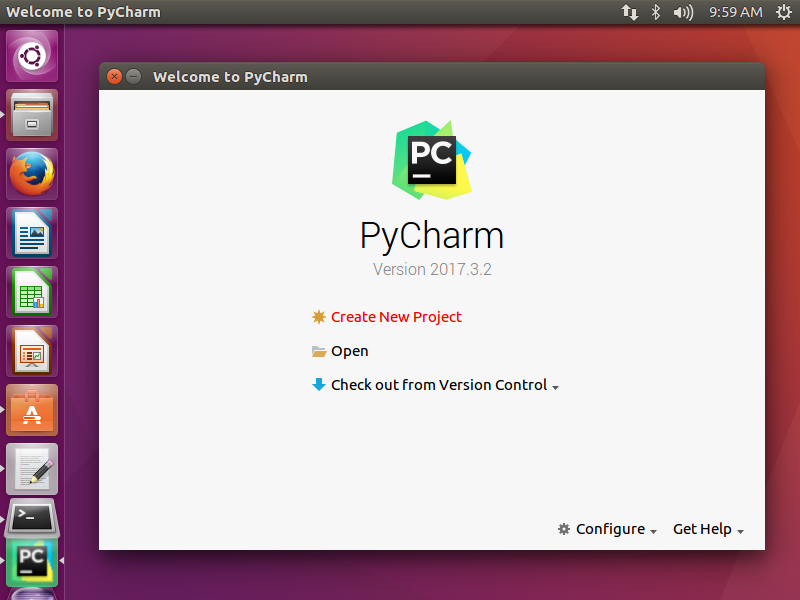
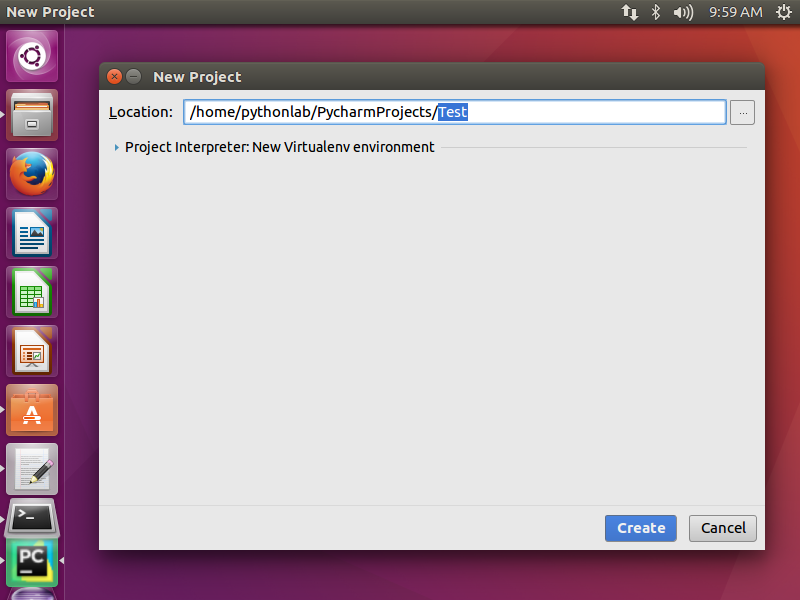
**Steps to install Pycharm**

Step 1-: Download and Install Pycharm from <https://www.jetbrains.com/pycharm/download> during download please select community edition (installation is same as python in Ubuntu)

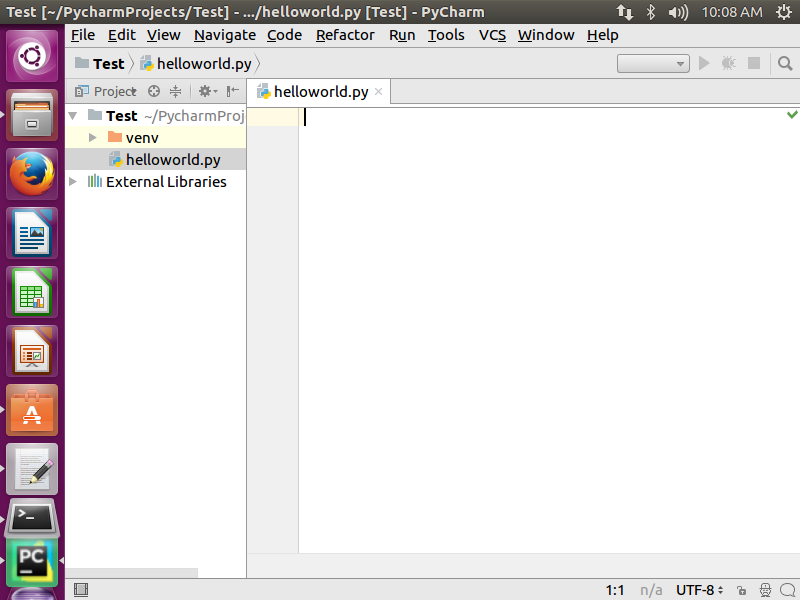




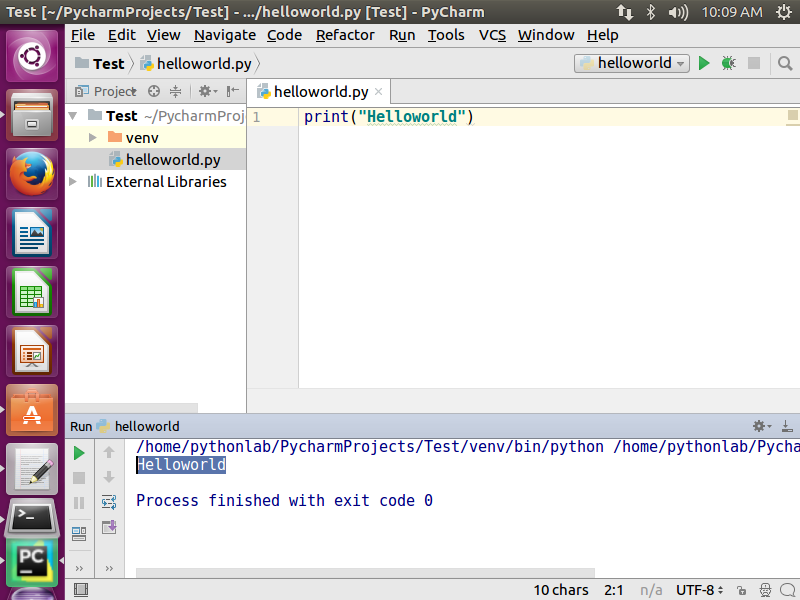
Step 2-: Once Pycharm is ready then Click on Create new project and Specify a project name like test in this example and click on Create.

Step 3-: Rt click on Test project and select New Python file, specify its name to it like helloworld.py in this example.



Step 4-: Write and test application in Pycharm



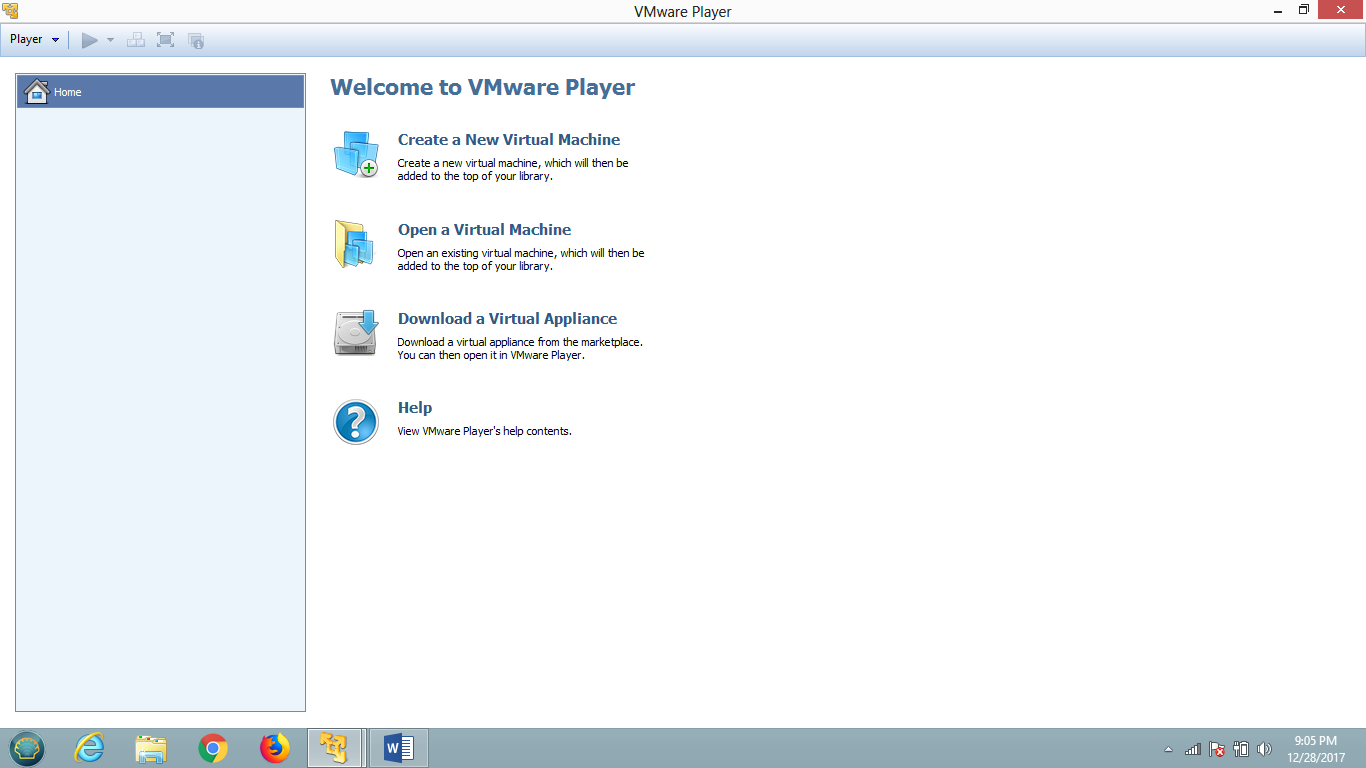
Steps to open python Virtual Machine

Step 1 -: Download VMware Player from following Link or search it on Google for alternate links.

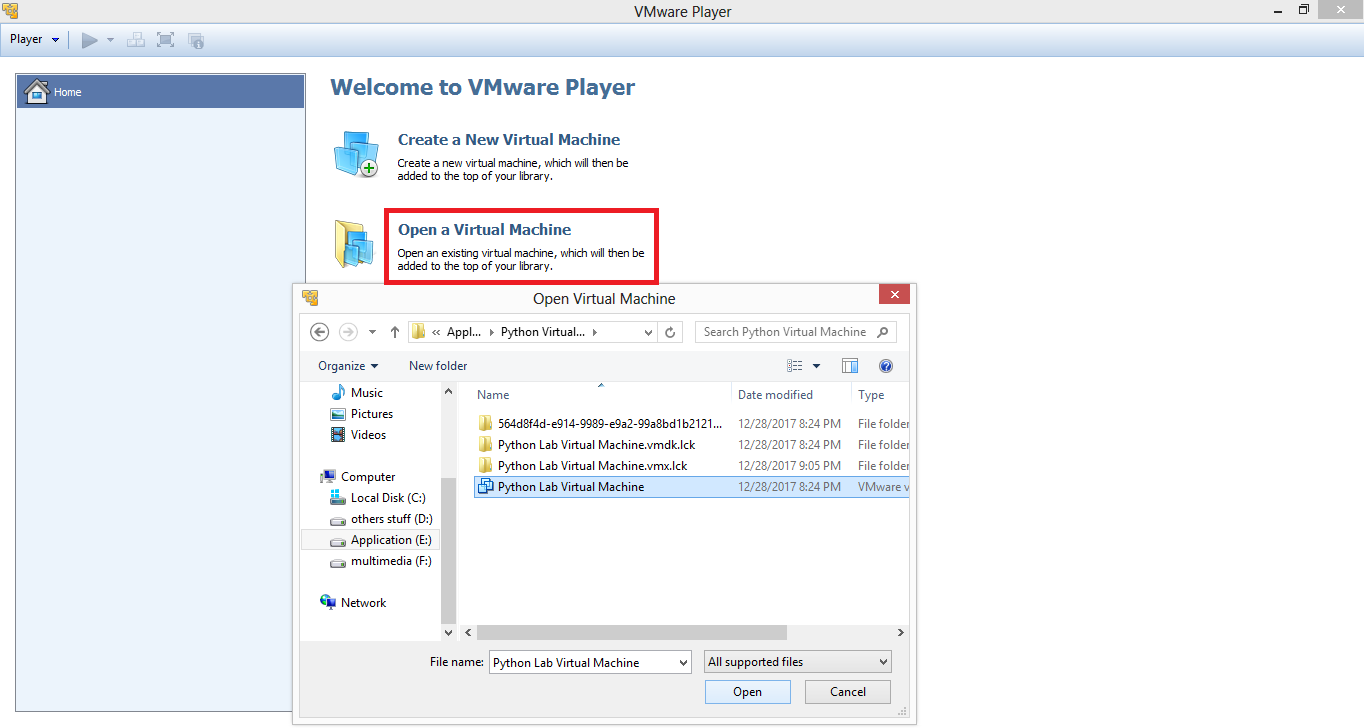
<https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/12_0>

Once package is downloaded then install it.

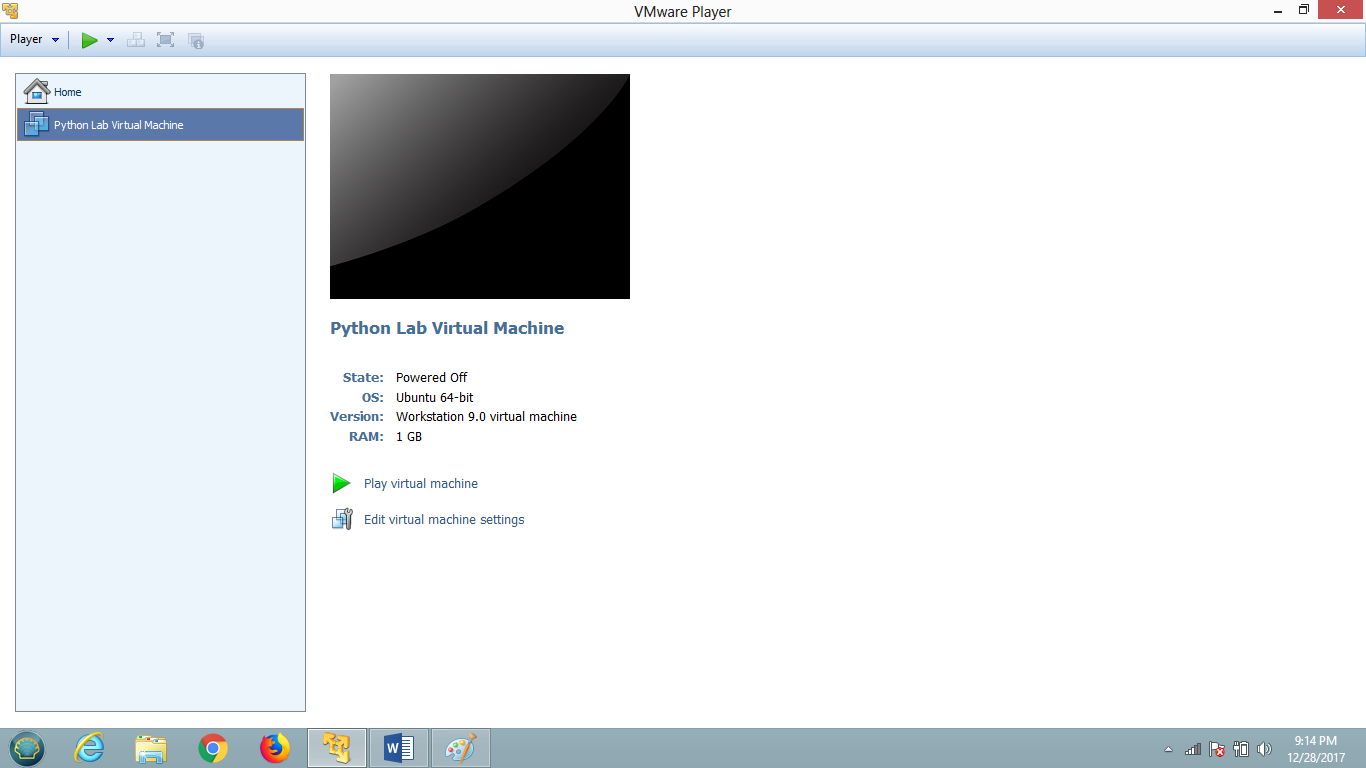
Step 2 -: Once Installation is done then Open it.



Step 3 -: Click on Open Virtual machine and Locate VMDK file (Copy Python Virtual machine folder in C drive and locate VMDK file in Given Folder)



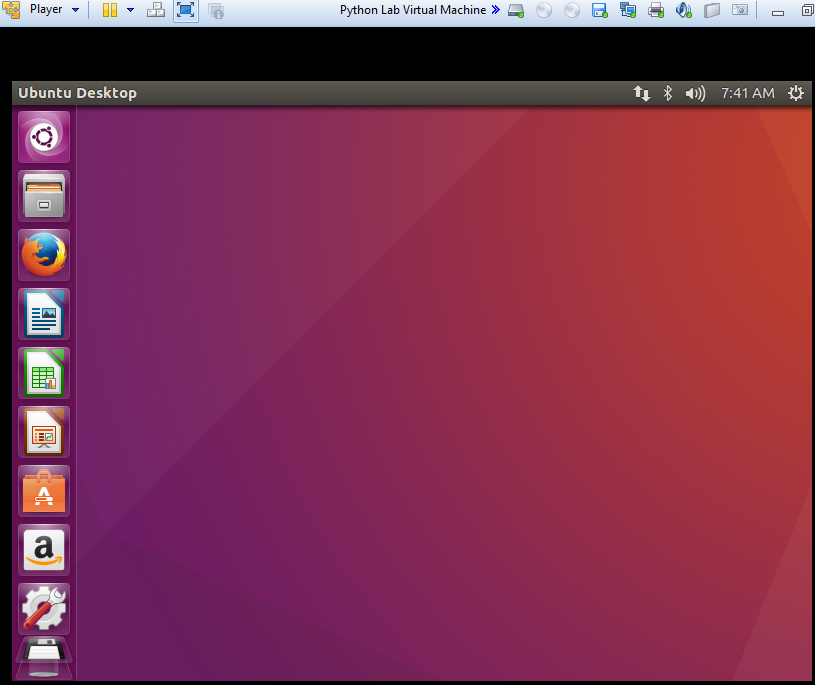
Step 4 -: Select Python Lab Virtual Machine and Click on play Virtual machine.



Step 5 -: Specify the credentials as below

Username -: pythonlab

Password -: python



**EXPERIMENT NO 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Aim: Write python programs to understand Comments, Datatypes, Expressions, Input and Output Functions.**    **Theory:** [A] Python Comments: Comments are for developers. They describe parts of the code where necessary to facilitate the understanding of programmers, including yourself. **To write a comment in Python, simply put the hash mark # before your desired comment:**  # This is a comment  Python ignores everything after the hash mark and up to the end of the line. You can insert them anywhere in your code, even in line with other code:  print("This will run.") # This won't run  When you run the above code, you will only see the output This will run. Everything else is ignored.  Comments should be short, sweet, and to the point. While PEP 8 advises keeping code at 79 characters or fewer per line, it suggests a max of 72 characters for inline comments and docstrings. If your comment is approaching or exceeding that length, then you’ll want to spread it out over multiple lines.  **Python Multiline Comments**  Unfortunately, Python doesn’t have a way to write multiline comments as you can in languages such as C, Java, and Go:  # So you can't  just do this  in python  In the above example, the first line will be ignored by the program, but the other lines will raise a Syntax Error.  **In contrast, a language like Java will allow you to spread a comment out over multiple lines quite easily:**  /\* You can easily  write multiline  comments in Java \*/  Everything between /\* and \*/ is ignored by the program.  While Python doesn’t have native multiline commenting functionality, you can create multiline comments in Python. There are two simple ways to do so.  **The first way is simply by pressing the return key after each line, adding a new hash mark and continuing your comment from there:**  def multiline\_example():  # This is a pretty good example  # of how you can spread comments  # over multiple lines in Python  Each line that starts with a hash mark will be ignored by the program.  **Another thing you can do is use multiline strings by wrapping your comment inside a set of triple quotes:**  """  If I really hate pressing `enter` and  typing all those hash marks, I could  just do this instead  """  **[B]Datatypes:**  **Assigning Values to Variables**  Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.  The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable. For example −  #!/usr/bin/python3  counter = 100 # An integer assignment  miles = 1000.0 # A floating point  name = "John" # A string  print (counter)  print (miles)  print (name)  Here, 100, 1000.0 and "John" are the values assigned to counter, miles, and name variables, respectively. This produces the following result −  100  1000.0  John  **Multiple Assignment**  Python allows you to assign a single value to several variables simultaneously.  For example −  a = b = c = 1  Here, an integer object is created with the value 1, and all the three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables. For example −  a, b, c = 1, 2, "john"  Here, two integer objects with values 1 and 2 are assigned to the variables a and b respectively, and one string object with the value "john" is assigned to the variable c.  **Standard Data Types**  The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.  **Python has five standard data types −**   * Numbers * String * List * Tuple * Dictionary   **Python Numbers**  Number data types store numeric values. Number objects are created when you assign a value to them. For example −  var1 = 1  var2 = 10  You can also delete the reference to a number object by using the **del** statement. The syntax of the **del** statement is −  del var1[,var2[,var3[....,varN]]]]  You can delete a single object or multiple objects by using the **del** statement.  **For example −**  del var  del var\_a, var\_b  Python supports three different numerical types −   * int (signed integers) * float (floating point real values) * complex (complex numbers)   All integers in Python3 are represented as long integers. Hence, there is no separate number type as long.  **Examples**  Here are some examples of numbers −   |  |  |  | | --- | --- | --- | | **int** | **float** | **complex** | | 10 | 0.0 | 3.14j | | 100 | 15.20 | 45.j | | -786 | -21.9 | 9.322e-36j | | 080 | 32.3+e18 | .876j |   A complex number consists of an ordered pair of real floating-point numbers denoted by x + yj, where x and y are real numbers and j is the imaginary unit.  **Taking input in Python**  Developers often have a need to interact with users, either to get data or to provide some sort of result. Most programs today use a dialog box as a way of asking the user to provide some type of input. While Python provides us with two inbuilt functions to read the input from the keyboard.  raw\_input ( prompt )  input ( prompt )  raw\_input ( ) : This function works in older version (like Python 2.x). This function takes exactly what is typed from the keyboard, convert it to string and then return it to the variable in which we want to store. For example –  # Python program showing  # a use of raw\_input()  g = raw\_input("Enter your name : ")  print g  Here, g is a variable which will get the string value, typed by user during the execution of program. Typing of data for the raw\_input() function is terminated by enter key. We can use raw\_input() to enter numeric data also. In that case we use typecasting.For more details on typecasting refer this.  input ( ) : This function first takes the input from the user and then evaluates the expression, which means Python automatically identifies whether user entered a string or a number or list. If the input provided is not correct then either syntax error or exception is raised by python. For example –  # Python program showing  # a use of input()  val = input("Enter your value: ")  print(val)  How the input function works in Python :  When input() function executes program flow will be stopped until the user has given an input.  The text or message display on the output screen to ask a user to enter input value is optional i.e. the prompt, will be printed on the screen is optional.  Whatever you enter as input, input function convert it into a string. if you enter an integer value still input() function convert it into a string. You need to explicitly convert it into an integer in your code using typecasting.  **Output using print() function**  The simplest way to produce output is using the print() function where you can pass zero or more expressions separated by commas. This function converts the expressions you pass into a string before writing to the screen.  Syntax: print(value(s), sep= ‘ ‘, end = ‘\n’, file=file, flush=flush)  Parameters:  value(s) : Any value, 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 :’ ‘  end=’end’: (Optional) Specify what to print at the end.Default : ‘\n’  file : (Optional) An object with a write method. Default :sys.stdout  flush : (Optional) A Boolean, specifying if the output is flushed (True) or buffered (False). Default: False  **Code:**  print "---------------COMMENTS-----------------"  print 'Single Line "#"'  print 'Multi Line ("""---""")'  print "Multi Line ('''---''')"  print ""  print "--Datatypes, Expressions, Output Statement & Conversions-"  a=16 #int datatype  a1="abc" #string datatype  b=23.32 #Float datatype  c=2.3+3.2j #Complex datatype  d=1.2-2.8j #Complex datatype  e=c+d #expression  e1=a\*b #expression  print "Sum is",e #Output statement  print 'Multiplication is ', e1 #Output statement  f=float(a) #Converting int to float  print 'Int to Float', f  g=complex(b) #Converting float to complex  print 'Float to Complex', g  h=oct(a) #Decimal to octal  print 'Decimal to Octal', h  i=bin(a) #Decimal to binary  print 'Decimal to Binary', i  j=hex(a) #Decimal to hexadecimal  print 'Decimal to Hexadecimal',j  print ""  print "------INPUT STATEMENT---------"  k=raw\_input('What is your name? using raw input ') #ask & returns string of data  print 'Your name is ' + k  l=input('Enter value of x ')  print 'Entered value of x using String Function is ' + str(l) #Printing Output in String Format  print 'Entered value of x using format function is {0}'.format(l) #Printing Formatted Output  print "accept integer input",int(input()) #Accept Integer Input  print ""  print "-------Knowing the datatype------"  a=12  print type(a)  b="str"  print type(b)  c=1.2+4j  print type(c)    **OUTPUT OF PROGRAM**  ---------------------------------------------------------------------------------------------------------------- |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Experiment 2 :**  **Aim : Write python programs to understand Byte array, Range, Set and STRING**  **Functions**  **Aim:**  **[A] bytearray() function**  bytearray() method returns a bytearray object which is an array of given bytes. It gives a mutable sequence of integers in the range 0 <= x < 256.  Syntax:  bytearray(source, encoding, errors)  **Parameters:**  source[optional]: Initializes the array of bytes  encoding[optional]: Encoding of the string  errors[optional]: Takes action when encoding fails  Returns: Returns an array of bytes of the given size.  source parameter can be used to initialize the array in few different ways.  **[B] Python range() Function**  Example  Create a sequence of numbers from 0 to 5, and print each item in the sequence:  x = range(6)  for n in x:  print(n)  Definition and Usage : The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.  **Syntax:**  **range(start, stop, step)** **Parameter Values**  |  |  | | --- | --- | | **Parameter** | **Description** | | *start* | Optional. An integer number specifying at which position to start. Default is 0 | | *stop* | Required. An integer number specifying at which position to end. | | *step* | Optional. An integer number specifying the incrementation. Default is 1 |   **Example**  Create a sequence of numbers from 3 to 5, and print each item in the sequence:  x = range(3, 6)  for n in x:  print(n)  Example  Create a sequence of numbers from 3 to 19, but increment by 2 instead of 1:  x = range(3, 20, 2)  for n in x:  print(n)  **[C] Set :** A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.  **Example**  Create a Set:  thisset = {"apple", "banana", "cherry"}  print(thisset)  **Access Items :** You cannot access items in a set by referring to an index, since sets are unordered the items has no index. But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.  **Example**  Loop through the set, and print the values:  thisset = {"apple", "banana", "cherry"}  for x in thisset:  print(x)  **Python String Methods**  Python has a set of built-in methods that you can use on strings.   |  |  | | --- | --- | | **Method** | **Description** | | [capitalize()](https://www.w3schools.com/python/ref_string_capitalize.asp) | Converts the first character to upper case | | [casefold()](https://www.w3schools.com/python/ref_string_casefold.asp) | Converts string into lower case | | [center()](https://www.w3schools.com/python/ref_string_center.asp) | Returns a centered string | | [count()](https://www.w3schools.com/python/ref_string_count.asp) | Returns the number of times a specified value occurs in a string | | [encode()](https://www.w3schools.com/python/ref_string_encode.asp) | Returns an encoded version of the string | | [endswith()](https://www.w3schools.com/python/ref_string_endswith.asp) | Returns true if the string ends with the specified value | | [expandtabs()](https://www.w3schools.com/python/ref_string_expandtabs.asp) | Sets the tab size of the string | | [find()](https://www.w3schools.com/python/ref_string_find.asp) | Searches the string for a specified value and returns the position of where it was found | | [format()](https://www.w3schools.com/python/ref_string_format.asp) | Formats specified values in a string | | format\_map() | Formats specified values in a string | | [index()](https://www.w3schools.com/python/ref_string_index.asp) | Searches the string for a specified value and returns the position of where it was found | | [isalnum()](https://www.w3schools.com/python/ref_string_isalnum.asp) | Returns True if all characters in the string are alphanumeric | | [isalpha()](https://www.w3schools.com/python/ref_string_isalpha.asp) | Returns True if all characters in the string are in the alphabet | | [isdecimal()](https://www.w3schools.com/python/ref_string_isdecimal.asp) | Returns True if all characters in the string are decimals | | [isdigit()](https://www.w3schools.com/python/ref_string_isdigit.asp) | Returns True if all characters in the string are digits | | [isidentifier()](https://www.w3schools.com/python/ref_string_isidentifier.asp) | Returns True if the string is an identifier | | [islower()](https://www.w3schools.com/python/ref_string_islower.asp) | Returns True if all characters in the string are lower case | | [isnumeric()](https://www.w3schools.com/python/ref_string_isnumeric.asp) | Returns True if all characters in the string are numeric | | [isprintable()](https://www.w3schools.com/python/ref_string_isprintable.asp) | Returns True if all characters in the string are printable | | [isspace()](https://www.w3schools.com/python/ref_string_isspace.asp) | Returns True if all characters in the string are whitespaces | | [istitle()](https://www.w3schools.com/python/ref_string_istitle.asp) | Returns True if the string follows the rules of a title | | [isupper()](https://www.w3schools.com/python/ref_string_isupper.asp) | Returns True if all characters in the string are upper case | | [join()](https://www.w3schools.com/python/ref_string_join.asp) | Joins the elements of an iterable to the end of the string | | [ljust()](https://www.w3schools.com/python/ref_string_ljust.asp) | Returns a left justified version of the string | | [lower()](https://www.w3schools.com/python/ref_string_lower.asp) | Converts a string into lower case | | [lstrip()](https://www.w3schools.com/python/ref_string_lstrip.asp) | Returns a left trim version of the string | | maketrans() | Returns a translation table to be used in translations | | [partition()](https://www.w3schools.com/python/ref_string_partition.asp) | Returns a tuple where the string is parted into three parts | | [replace()](https://www.w3schools.com/python/ref_string_replace.asp) | Returns a string where a specified value is replaced with a specified value | | [rfind()](https://www.w3schools.com/python/ref_string_rfind.asp) | Searches the string for a specified value and returns the last position of where it was found | | [rindex()](https://www.w3schools.com/python/ref_string_rindex.asp) | Searches the string for a specified value and returns the last position of where it was found | | [rjust()](https://www.w3schools.com/python/ref_string_rjust.asp) | Returns a right justified version of the string | | [rpartition()](https://www.w3schools.com/python/ref_string_rpartition.asp) | Returns a tuple where the string is parted into three parts | | [rsplit()](https://www.w3schools.com/python/ref_string_rsplit.asp) | Splits the string at the specified separator, and returns a list | | [rstrip()](https://www.w3schools.com/python/ref_string_rstrip.asp) | Returns a right trim version of the string | | [split()](https://www.w3schools.com/python/ref_string_split.asp) | Splits the string at the specified separator, and returns a list | | [splitlines()](https://www.w3schools.com/python/ref_string_splitlines.asp) | Splits the string at line breaks and returns a list | | [startswith()](https://www.w3schools.com/python/ref_string_startswith.asp) | Returns true if the string starts with the specified value | | [strip()](https://www.w3schools.com/python/ref_string_strip.asp) | Returns a trimmed version of the string | | [swapcase()](https://www.w3schools.com/python/ref_string_swapcase.asp) | Swaps cases, lower case becomes upper case and vice versa | | [title()](https://www.w3schools.com/python/ref_string_title.asp) | Converts the first character of each word to upper case | | translate() | Returns a translated string | | [upper()](https://www.w3schools.com/python/ref_string_upper.asp) | Converts a string into upper case | | [zfill()](https://www.w3schools.com/python/ref_string_zfill.asp) | Fills the string with a specified number of 0 values at the beginning |   #Demonstrating  print "---------------BYTEARRAY FUNCTIONS-----------------"  print ""  x=[10,20,30,40,50] #Creating integer List  print "Creating a bytearray using bytearray function"  x1=bytearray(x) #Converting List in to Bytearray  print "printing bytearray:"  print(x[0],x[1],x[2],x[3],x[4]) #printing Bytearray  print ""  print "--------RANGE DATATYPE ------------"  print ""  r=range(1,9) #Creating a range of 1 to 9 elements  print "range of 1 to 9 is :",r  r=range(10) #Creating a range first 10 elements  print "range of 10 is :",r  print ""  print "--SET DATATYPE (Unordered elements with no duplicate)--"  print ""  s={1,4,6,3} #Creating a set  print s #Printing set in ascending order  c1=set("BHUSHANA") #Creating a character set  print c1 #Printing set alphabetically and avoiding duplication of Character  s.update([4,5]) #Adding element 5 after 4 in set s  print s #Printing set in ascending order  fs=frozenset(s) #Creating a frozen of set s i.e. Fixed Set cant be modified  print fs  print ""  print "---------------STRING -----------------"  print ""  str="""WELCOME TO 'CORE PYTHON PROGRAMMING' COURSE""" #STRING Datatype  print "printing String array:"  print str  print "Printing first four element of String Array:" #Printing first four element of String Array  #FIRST POSITION WILL BE A SPACE SO PRINT ACCORDINGLY  print str[0:4]  print "Printing first character from end:" #Printing first character from end  print str[-1]  print "Printing nineth character of String Array:" #Printing nineth character  print str[8]  print ""  print "---------------STRING FUNCTIONS-----------------"  print ""  s1="CORE PYTHON PROGRAMMING"  s2="IS BEST"  print "Length:", len(s1) #Printing length  print "Repeate twice:",(s1)\*2 #Repeat string  print "Concat : specified Character in the strings:",s1.count('M') #Counting  a1='CORE'  a2='BEST'  print "Replacing CORE with BEST:", s1.replace(a1,a2)  print "Changing case from uppercase to Lower:", s1.lower()  print "SORTING STRINGS:", sorted(s2)  s4=u'\u0915\u0947\u0964\u0930 \u092a\u0948\u0925\u0964\u0928' #String with unicode  print "UNICODE FOR CORE PYTHON", s4  **OUTPUT OF PROGRAM**  -------------------------------------------------------------------- |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Aim 3 : Write python programs to understand** **List Tuple, Dictionaries and Arrays**  **Theory:**  **[A ]List:**  The list is the most versatile datatype available in Python, which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that the items in a list need not be of the same type.  Creating a list is as simple as putting different comma-separated values between square brackets. For example −  list1 = ['physics', 'chemistry', 1997, 2000];  list2 = [1, 2, 3, 4, 5 ];  list3 = ["a", "b", "c", "d"];  Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on. **Accessing Values in Lists** To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −  #!/usr/bin/python3  list1 = ['physics', 'chemistry', 1997, 2000]  list2 = [1, 2, 3, 4, 5, 6, 7 ]  print ("list1[0]: ", list1[0])  print ("list2[1:5]: ", list2[1:5])  When the above code is executed, it produces the following result −  list1[0]: physics  list2[1:5]: [2, 3, 4, 5] **Updating Lists** You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method. For example −  #!/usr/bin/python3  list = ['physics', 'chemistry', 1997, 2000]  print ("Value available at index 2 : ", list[2])  list[2] = 2001  print ("New value available at index 2 : ", list[2])  **Note** − The append() method is discussed in the subsequent section.  When the above code is executed, it produces the following result −  Value available at index 2 : 1997  New value available at index 2 : 2001 **Delete List Elements** To remove a list element, you can use either the **del** statement if you know exactly which element(s) you are deleting. You can use the remove() method if you do not know exactly which items to delete. For example −  #!/usr/bin/python3  list = ['physics', 'chemistry', 1997, 2000]  print (list)  del list[2]  print ("After deleting value at index 2 : ", list)  When the above code is executed, it produces the following result −  ['physics', 'chemistry', 1997, 2000]  After deleting value at index 2 : ['physics', 'chemistry', 2000]  Python includes the following list functions −   |  |  | | --- | --- | | **Sr.No.** | **Function & Description** | | 1 | [cmp(list1, list2)](https://www.tutorialspoint.com/python3/list_cmp.htm) : No longer available in Python 3. | | 2 | [len(list)](https://www.tutorialspoint.com/python3/list_len.htm) : Gives the total length of the list. | | 3 | [max(list)](https://www.tutorialspoint.com/python3/list_max.htm) : Returns item from the list with max value. | | 4 | [min(list)](https://www.tutorialspoint.com/python3/list_min.htm) : Returns item from the list with min value. | | 5 | [list(seq)](https://www.tutorialspoint.com/python3/list_list.htm) : Converts a tuple into list. |   Python includes the following list methods −   |  |  | | --- | --- | | **Sr** | **Methods & Description** | | 1 | [list.append(obj)](https://www.tutorialspoint.com/python3/list_append.htm) : Appends object obj to list | | 2 | [list.count(obj)](https://www.tutorialspoint.com/python3/list_count.htm) : Returns count of how many times obj occurs in list | | 3 | [list.extend(seq)](https://www.tutorialspoint.com/python3/list_extend.htm) : Appends the contents of seq to list | | 4 | [list.index(obj)](https://www.tutorialspoint.com/python3/list_index.htm) : Returns the lowest index in list that obj appears | | 5 | [list.insert(index, obj)](https://www.tutorialspoint.com/python3/list_insert.htm) : Inserts object obj into list at offset index | | 6 | [list.pop(obj = list[-1])](https://www.tutorialspoint.com/python3/list_pop.htm) : Removes and returns last object or obj from list | | 7 | [list.remove(obj)](https://www.tutorialspoint.com/python3/list_remove.htm) : Removes object obj from list | | 8 | [list.reverse()](https://www.tutorialspoint.com/python3/list_reverse.htm) : Reverses objects of list in place | | 9 | [list.sort([func])](https://www.tutorialspoint.com/python3/list_sort.htm) : Sorts objects of list, use compare func if given |   **[B] Tuple:**  A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The main difference between the tuples and the lists is that the tuples cannot be changed unlike lists. Tuples use parentheses, whereas lists use square brackets.  Creating a tuple is as simple as putting different comma-separated values. Optionally, you can put these comma-separated values between parentheses also. For example −  tup1 = ('physics', 'chemistry', 1997, 2000)  tup2 = (1, 2, 3, 4, 5 )  tup3 = "a", "b", "c", "d"  The empty tuple is written as two parentheses containing nothing −  tup1 = ();  To write a tuple containing a single value you have to include a comma, even though there is only one value −  tup1 = (50,)  Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.  **Accessing Values in Tuples**  To access values in tuple, use the square brackets for slicing along with the index or indices to obtain the value available at that index. For example −  #!/usr/bin/python3  tup1 = ('physics', 'chemistry', 1997, 2000)  tup2 = (1, 2, 3, 4, 5, 6, 7 )  print ("tup1[0]: ", tup1[0])  print ("tup2[1:5]: ", tup2[1:5])  When the above code is executed, it produces the following result −  tup1[0]: physics  tup2[1:5]: (2, 3, 4, 5)  **Updating Tuples**  Tuples are immutable, which means you cannot update or change the values of tuple elements. You are able to take portions of the existing tuples to create new tuples as the following example demonstrates −    #!/usr/bin/python3  tup1 = (12, 34.56)  tup2 = ('abc', 'xyz')  # Following action is not valid for tuples  # tup1[0] = 100;  # So let's create a new tuple as follows  tup3 = tup1 + tup2  print (tup3)  When the above code is executed, it produces the following result −  (12, 34.56, 'abc', 'xyz')  **Delete Tuple Elements**  Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.  **To explicitly remove an entire tuple, just use the del statement. For example −**  #!/usr/bin/python3  tup = ('physics', 'chemistry', 1997, 2000);  print (tup)  del tup;  print ("After deleting tup : ")  print (tup)  This produces the following result.  **Note − An exception is raised. This is because after del tup, tuple does not exist anymore.**  ('physics', 'chemistry', 1997, 2000)  After deleting tup :  Traceback (most recent call last):  File "test.py", line 9, in <module>  print tup;  NameError: name 'tup' is not defined **Basic Tuples Operations** Tuples respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string.  In fact, tuples respond to all of the general sequence operations we used on strings in the previous chapter.   |  |  |  | | --- | --- | --- | | **Python Expression** | **Results** | **Description** | | len((1, 2, 3)) | 3 | Length | | (1, 2, 3) + (4, 5, 6) | (1, 2, 3, 4, 5, 6) | Concatenation | | ('Hi!',) \* 4 | ('Hi!', 'Hi!', 'Hi!', 'Hi!') | Repetition | | 3 in (1, 2, 3) | True | Membership | | for x in (1,2,3) : print (x, end = ' ') | 1 2 3 | Iteration |  **Indexing, Slicing, and Matrixes** Since tuples are sequences, indexing and slicing work the same way for tuples as they do for strings, assuming the following input −  T=('C++', 'Java', 'Python')   |  |  |  | | --- | --- | --- | | **Python Expression** | **Results** | **Description** | | T[2] | 'Python' | Offsets start at zero | | T[-2] | 'Java' | Negative: count from the right | | T[1:] | ('Java', 'Python') | Slicing fetches sections |  **No Enclosing Delimiters** No enclosing Delimiters is any set of multiple objects, comma-separated, written without identifying symbols, i.e., brackets for lists, parentheses for tuples, etc., default to tuples, as indicated in these short examples. **Built-in Tuple Functions** Python includes the following tuple functions −   |  |  | | --- | --- | | **Sr.** | **Function & Description** | | 1 | [cmp(tuple1, tuple2)](https://www.tutorialspoint.com/python3/tuple_cmp.htm) : Compares elements of both tuples. | | 2 | [len(tuple)](https://www.tutorialspoint.com/python3/tuple_len.htm) : Gives the total length of the tuple. | | 3 | [max(tuple)](https://www.tutorialspoint.com/python3/tuple_max.htm) : Returns item from the tuple with max value. | | 4 | [min(tuple)](https://www.tutorialspoint.com/python3/tuple_min.htm) : Returns item from the tuple with min value. | | 5 | [tuple(seq)](https://www.tutorialspoint.com/python3/tuple_tuple.htm) : Converts a list into tuple. |   **[C] Dictionary**  Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a ‘comma’.  **A Dictionary** in Python works similar to the Dictionary in a real world. Keys of a Dictionary must be unique and of immutable data type such as Strings, Integers and tuples, but the key-values can be repeated and be of any type.  Note – Keys in a dictionary doesn’t allows Polymorphism.  In Python, a Dictionary can be created by placing sequence of elements within curly {} braces, separated by ‘comma’. Dictionary holds a pair of values, one being the Key and the other corresponding pair element being its Key:value. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can’t be repeated and must be immutable.  **Dictionary** can also be created by the built-in function dict(). An empty dictionary can be created by just placing to curly braces{}.  Note – Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.  **Example :**  # Creating an empty Dictionary  Dict = {}  print("Empty Dictionary: ")  print(Dict)  # Creating a Dictionary  # with Integer Keys  Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}  print("\nDictionary with the use of Integer Keys: ")  print(Dict)  # Creating a Dictionary  # with Mixed keys  Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}  print("\nDictionary with the use of Mixed Keys: ")  print(Dict)  # Creating a Dictionary  # with dict() method  Dict = dict({1: 'Geeks', 2: 'For', 3:'Geeks'})  print("\nDictionary with the use of dict(): ")  print(Dict)  # Creating a Dictionary  # with each item as a Pair  Dict = dict([(1, 'Geeks'), (2, 'For')])  print("\nDictionary with each item as a pair: ")  print(Dict)  **Output:**  Empty Dictionary:  {}  Dictionary with the use of Integer Keys:  {1: 'Geeks', 2: 'For', 3: 'Geeks'}  Dictionary with the use of Mixed Keys:  {1: [1, 2, 3, 4], 'Name': 'Geeks'}  Dictionary with the use of dict():  {1: 'Geeks', 2: 'For', 3: 'Geeks'}  Dictionary with each item as a pair:  {1: 'Geeks', 2: 'For'}  **[D] Array :** Arrays are used to store multiple values in one single variable:  Example : Create an array containing car names:  cars = ["Ford", "Volvo", "BMW"]  **What is an Array?**  An array is a special variable, which can hold more than one value at a time.  If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:  car1 = "Ford"  car2 = "Volvo"  car3 = "BMW"  However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?  The solution is an array!  An array can hold many values under a single name, and you can access the values by referring to an index number.  **Access the Elements of an Array**  You refer to an array element by referring to the index number.  **Example : Get the value of the first array item:**  x = cars[0]  Example  Modify the value of the first array item:  cars[0] = "Toyota"  The Length of an Array  Use the len() method to return the length of an array (the number of elements in an array).  **Example : Return the number of elements in the cars array:**  x = len(cars)  Note: The length of an array is always one more than the highest array index.  Looping Array Elements : You can use the for in loop to loop through all the elements of an array.  **Example : Print each item in the cars array:**  for x in cars:  print(x)  Adding Array Elements  You can use the append() method to add an element to an array.  **Example : Add one more element to the cars array:**  cars.append("Honda")  Removing Array Elements  You can use the pop() method to remove an element from the array.  **Example : Delete the second element of the cars array:**  cars.pop(1)  You can also use the remove() method to remove an element from the array.  **Example : Delete the element that has the value "Volvo":**  cars.remove("Volvo") **Array Methods :** Python has a set of built-in methods that you can use on lists/arrays.  |  |  | | --- | --- | | **Method** | **Description** | | [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list | | [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list | | [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list | | [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value | | [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list | | [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value | | [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position | | [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position | | [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value | | [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list | | [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |   **Code:**  from array import \* #Import an array package  print "---------------LIST -----------------"  print ""  l=[12,23,-5,0.8,'python',"BJ"] #Creating a list of variable datatypes  print "printing original list",l #printing list  print "printing first three elements of a list", l[0:3] #printing first three elements of a list  print "printing Last elements of a list",l[-1] #printing Last elements of a list  print "printing first three elements of a list twice",l[0:3] \* 2 #printing first three elements of a list twice  print ""  print "---------------LIST FUNCTIONS-----------------"  print ""  l1=list(range(1,8)) #Creating a list of range 1 to 7  print "List of range 1 to 7", l1  l1.append(9) #append 9  print "Append 9 :",l1  l1.sort(reverse=True)  print "Descending Sort:",l1  l1.sort()  print "Ascending Sort:",l1  l1.reverse()  print "Reverse:",l1  l1.sort()  l1.remove(9)  print "Remove 9:",l1  print "count:",l1.count(5)  print "max:",max(l1)  print "min:",min(l1)  print "Index of 6:",l1.index(6)  print "--------MATRICES using LIST---------------"  print ""  m1=[1,2,3],[4,5,6],[7,8,9]  for r in m1:  print(r)  print ""  print "--------TOUPLE DATATYPE (READ ONLY LIST)------------"  print ""  tpl=(-5,0.8,'python',"BJ") #Creating a Touple  print "Created Touple is ", tpl  print "Touple elements 0 to 2 is", tpl[0:2] #printing first two elements of a touple  print "Touple element -2 is ", tpl[-2] #printing Second last elements of a touple  print ""  print "--------TOUPLE FUNCTIONS -----------"  tpl2=(10,20,30,40,10,20,10) #Creating a Touple2  print "Created Touple2 is ", tpl2  print "Length : ", len(tpl2)  print "Min : ", min(tpl2)  print "Max : ", max(tpl2)  print "Count no. of 10's: ", tpl2.count(10)  print "Reverse Sort : ", sorted(tpl2,reverse=True)  print ""  print "--------DICTIONARIES i.e. key:value pair------------"  print ""  d1={'Name':"bhushan",'gender':"M",'age':32,'college':"TCET"} #create dictionary  print "print dictionary: ",d1  print ""  print "-->Keys :",d1.keys()  print "-->Values :",d1.values()  print "-->Keys and values :",d1.items()  d1.update({'country':"india"})  print "-->Print updated dictionary: ",d1  c=sorted(d1.items(),key=lambda t:t[1])  print "-->Sort By values :",c  print "---------------ARRAYS -----------------"  print ""  arr=array('i',[10,20,30,40,50]) #Create array with integer values  print "The Array Elements are"  for i in arr: #i is element and arr in array name  print i #Requires indentation  print "length of array is",len(arr)  arr1=array('c',['a','b','c','d']) #Create array with character values  print "The Character Array Elements are"  for ch in arr1: #i is element and arr1 in array name  print ch #Requires indentation  --------------------------------------------------------------------------------------------------------------  **OUTPUT OF PROGRAM**  ---------------LIST -----------------  printing original list [12, 23, -5, 0.8, 'python', 'BJ']  printing first three elements of a list [12, 23, -5]  printing Last elements of a list BJ  printing first three elements of a list twice [12, 23, -5, 12, 23, -5]  ---------------LIST FUNCTIONS-----------------  List of range 1 to 7 [1, 2, 3, 4, 5, 6, 7]  Append 9 : [1, 2, 3, 4, 5, 6, 7, 9]  Descending Sort: [9, 7, 6, 5, 4, 3, 2, 1]  Ascending Sort: [1, 2, 3, 4, 5, 6, 7, 9]  Reverse: [9, 7, 6, 5, 4, 3, 2, 1]  Remove 9: [1, 2, 3, 4, 5, 6, 7]  count: 1  max: 7  min: 1  Index of 6: 5  --------MATRICES using LIST---------------  [1, 2, 3]  [4, 5, 6]  [7, 8, 9]  --------TOUPLE DATATYPE (READ ONLY LIST)------------  Created Touple is (-5, 0.8, 'python', 'BJ')  Touple elements 0 to 2 is (-5, 0.8)  Touple element -2 is python  --------TOUPLE FUNCTIONS -----------  Created Touple2 is (10, 20, 30, 40, 10, 20, 10)  Length : 7  Min : 10  Max : 40  Count no. of 10's: 3  Reverse Sort : [40, 30, 20, 20, 10, 10, 10]  --------DICTIONARIES i.e. key:value pair------------  print dictionary: {'gender': 'M', 'age': 32, 'college': 'TCET', 'Name': 'bhushan'}  -->Keys : ['gender', 'age', 'college', 'Name']  -->Values : ['M', 32, 'TCET', 'bhushan']  -->Keys and values : [('gender', 'M'), ('age', 32), ('college', 'TCET'), ('Name', 'bhushan')]  -->Print updated dictionary: {'gender': 'M', 'age': 32, 'college': 'TCET', 'Name': 'bhushan', 'country': 'india'}  -->Sort By values : [('age', 32), ('gender', 'M'), ('Name', 'bhushan'), ('country', 'india'), ('college', 'TCET')]  ---------------ARRAYS -----------------  The Array Elements are  10  20  30  40  50  length of array is 5  The Character Array Elements are  a  b  c  d |

**Experiment 4**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a. Aim : Write python programs to understand the Control Structures**  **Theory :**  In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.  Programming languages provide various control structures that allow for more complicated execution paths.  A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement −  Loop Architecture  Python programming language provides following types of loops to handle looping requirements.   |  |  | | --- | --- | | **Sr.No.** | **Loop Type & Description** | | 1 | [while loop](https://www.tutorialspoint.com/python/python_while_loop.htm) : Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. | | 2 | [for loop](https://www.tutorialspoint.com/python/python_for_loop.htm) : Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. | | 3 | [nested loops](https://www.tutorialspoint.com/python/python_nested_loops.htm) :You can use one or more loop inside any another while, for or do..while loop. |  **Loop Control Statements** Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.  Python supports the following control statements. Click the following links to check their detail.  Let us go through the loop control statements briefly   |  |  | | --- | --- | | **Sr.No.** | **Control Statement & Description** | | 1 | [break statement](https://www.tutorialspoint.com/python/python_break_statement.htm) : Terminates the loop statement and transfers execution to the statement immediately following the loop. | | 2 | [continue statement](https://www.tutorialspoint.com/python/python_continue_statement.htm) : Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating. | | 3 | [pass statement](https://www.tutorialspoint.com/python/python_pass_statement.htm) : The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute. |   **[I] for loop**  Syntax:  I)  for variablename in range(endlimit): # print from 0 to endlimit-1  #code  II)  for variablename in range(startvalue,endlimit): # print from startvalue to endlimit-1  #code  III)  # print from startvalue to endlimit-1 with specified updation  for variablename in range(startvalue,endlimit,updation):  #code  **Example 1:**  primes = [2, 3, 5, 7]  for i in primes:  print(i)  **Example2:**  # Prints out the numbers 0,1,2,3,4  for x in range(5):  print(x)  **Example3:**  # Prints out 3,4,5  for x in range(3, 6):  print(x)  **Example4:**  # Prints out 3,5,7  for x in range(3, 8, 2):  print(x)  **Example5:**  fruits = ["apple", "banana", "cherry"]  for x in fruits:  print(x)  **#Code: WAP to enter the n numbers and display the same on the output screen**  a= list()  num = int(input("Enter how many elements you want:"))  print ("Enter numbers in array: ")  for i in range(num):  n = int(input("Enter a num :"))  a.append(n)    print("array elements",a)  #second way  print("Array elements are: ")  for i in range(num):  print(a[i])  **#code: WAP to find the average of n numbers**  a= list()  num = int(input("Enter how many elements you want:"))  print ("Enter numbers in array: ")  for i in range(num):  n = int(input("Enter a num :"))  a.append(n)  sum=0  for i in range(num):  sum=sum+a[i]  avg=sum/num  print("Sum of numbers=",sum)  print()    **#code for factorial**  n=int(input("Enter a number"))  f=1  for i in range(1,n+1):  f=f\*i    print("factorial=",f)  **[II] while loop**  while condition:  statement  **Example1:**  i = 1  while i < 6:  print(i)  i += 1  **Example2:**  i = 1  while i < 6:  print(i)  if i == 3:  break  i += 1  **Example3:**  i = 0  while i < 6:  i += 1  if i == 3:  continue  print(i)  **Code:**  #Control Structures remember the indentations  print "--Finding MAX BETWEEN 3 Nos using IF-Else-Elif STATEMENT--"  a=input("Enter value of First element:")  b=input("Enter value of Second element:")  c=input("Enter value of Third element:")  if a>b and a>c:  print a,"is Greater"  elif b>c:  print b,"is Greater"  else:  print c,"is Greater"  print ""  print "-----Calculate Factorial using WHILE LOOP-----------"  num=int(input("Enter the number : "))  fact=1  i=1  while i <= num:  fact=fact\*i  i=i+1  print"Factorial of",num," is :",fact  print "-----Generate Fibonacci series using If statement & WHILE LOOP-----------"  n=int(input("How many numbers??? : "))  f1=0  f2=1  c=2  if n==1:  print f1  elif n==2:  print f1,'\n',f2  else:  print f1,'\n',f2  while c<n:  f=f1+f2  print f  f1,f2=f2,f  c+=1  **-------------------------OUTPUT----------------------------------------**  --Finding MAX BETWEEN 3 Nos using IF-Else-Elif STATEMENT--  Enter value of First element:43  Enter value of Second element:43  Enter value of Third element:2  43 is Greater  -----Calculate Factorial using WHILE LOOP-----------  Enter the number : 2  Factorial of 2 is : 2  -----Generate Fibonacci series using If statement & WHILE LOOP-----------  How many numbers??? : 10  0  1  1  2  3  5  8  13  21  34 |
|  |

|  |
| --- |
| **Code : Write python programs to understand the For Loop**  #FOR LOOP  print "--Multiplication table using FOR loop--"  n=int(input("Enter the number:"))  for i in range(1,11):  print n,"X",i,"=",n\*i  print ""  print "--Pascal triangles--"  n=int(input("Enter the rows:"))  trow=[1]  y=[0]  for x in range(max(n,0)):  print(trow)  trow=[l+r for l,r in zip(trow+y, y+trow)]  n>=1  **-------------------------------------------OUTPUT------------------------------------------------------**  --Multiplication table using FOR loop--  Enter the number:3  3 X 1 = 3  3 X 2 = 6  3 X 3 = 9  3 X 4 = 12  3 X 5 = 15  3 X 6 = 18  3 X 7 = 21  3 X 8 = 24  3 X 9 = 27  3 X 10 = 30 |
| **II. A Write python programs to understand Functions**  **Theory :**  **FUNCTIONS:**  A function is a block of organized, reusable code that is used to perform a single, related action. Once a function is written, it can be reused as and when required. So, functions are also called reusable code.  Functions provide modularity for programming. A module represents a part of the program. Usually, a programmer divides the main task into smaller sub tasks called modules. Code maintenance will become easy because of functions. When a new feature has to be added to the existing software, a new function can be written and integrated into the software. When there is an error in the software, the corresponding function can be modified without disturbing the other functions in the software.  The use of functions in a program will reduce the length of the program.  As you already know, Python gives you many built-in functions like sqrt( ), etc. but you can also create your own functions. These functions are called user-defined functions.  **Difference between a function and a method:**  A function can be written individually in a python program. A function is called using its name. When a function is written inside a class, it becomes a „method‟. A method is called using object name or class name. A method is called using one of the following ways:  **Objectname.methodname()**  **Classname.methodname()**  **Defining a Function**  You can define functions to provide the required functionality. Here are simple rules to define a function in Python.  • Function blocks begin with the keyword def followed by the function name and parentheses ( ).  • Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.  • The first statement of a function can be an optional statement - the documentation string of the function or docstring.  • The code block within every function starts with a colon (:) and is indented.  • The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return none.  **Syntax:**    By default, parameters have a positional behavior and you need to inform them in the same order that they were defined.  **Example:**  def add(a,b):  """This function sum the numbers""" c=a+b  print c return  Here, „def’ represents starting of function. „add’ is function name. After this name, parentheses ( ) are compulsory as they denote that it is a function and not a variable or something else. In the parentheses we wrote two variables „a‟ and „b‟ these variables are called „parameters‟. A parameter is a variable that receives data from outside a function. So, this function receives two values from outside and those are stored in the variables „a‟ and  „b‟. After parentheses, we put colon (:) that represents the beginning of the function body. The function body contains a group of statements called „suite‟.  **Calling Function:**  A function cannot run by its own. It runs only when we call it. So, the next step is to call function using its name. While calling the function, we should pass the necessary values to the function in the parentheses as:    Here, we are calling „add‟ function and passing two values 5 and 12 to that function.  When this statement is executed, the python interpreter jumps to the function definition and copies the values 5 and 12 into the parameters „a‟ and „b‟ respectively.  Example:  **Returning Results from a function:**  We can return the result or output from the function using a „return‟ statement in the function body. When a function does not return any result, we need not write the return statement in the body of the function.    **Returning multiple values from a function:**  A function can returns a single value in the programming languages like C, C++ and JAVA. But, in python, a function can return multiple values. When a function calculates multiple results and wants to return the results, we can use return statement as:  return a, b, c  Here, three values which are in „a‟, „b‟ and „c‟ are returned. These values are returned by the function as a tuple. To grab these values, we can three variables at the time of calling the function as:  x, y, z = functionName( )  Here, „x‟, „y‟ and „z‟ are receiving the three values returned by the function.  Example:  **Functions are First Class Objects:**  In Python, functions are considered as first class objects. It means we can use functions as perfect objects. In fact when we create a function, the Python interpreter internally creates an object. Since functions are objects, we can pass a function to another function just like we pass an object (or value) to a function. The following possibilities are:   * + It is possible to assign a function to a variable.   + It is possible to define one function inside another function.   + It is possible to pass a function as parameter to another function.   + It is possible that a function can return another function. To understand these points, we will take a few simple programs.   **Pass by Value:**  Pass by value represents that a copy of the variable value is passed to the function and any modifications to that value will not reflect outside the function. In python, the values are sent to functions by means of object references. We know everything is considered as an object in python. All numbers, strings, tuples, lists and dictionaries are objects.  **Pass by Reference:**  Pass by reference represents sending the reference or memory address of the variable to the function. The variable value is modified by the function through memory address and hence the modified value will reflect outside the function also.  In python, lists and dictionaries are mutable. That means, when we change their data, the same object gets modified and new object is not created. In the below program, we are passing a list of numbers to modify ( ) function. When we append a new element to the list, the same list is modified and hence the modified list is available outside the function also.  **Formal and Actual Arguments:**  When a function is defined, it may have some parameters. These parameters are useful to receive values from outside of the function. They are called „formal arguments‟. When we call the function, we should pass data or values to the function. These values are called „actual arguments‟. In the following code, „a‟ and „b‟ are formal arguments and „x‟ and  „y‟ are actual arguments.  **Example:**  def add(a,b): # a, b are formal arguments c=a+b  print c x,y=10,15  add(x,y) # x, y are actual arguments  The actual arguments used in a function call are of 4 types:  a) Positional arguments  b) Keyword arguments  c) Default arguments  d) Variable length arguments  a) Positional Arguments:  These are the arguments passed to a function in correct positional order. Here, the number of arguments and their position in the function definition should match exactly with the number and position of argument in the function call.  **Code:**  #Functions to run basic calculator  def menu(): #Defining function Menu  #print what options you have  print ""  print "Welcome to calculator in Python"  print "your options are:"  print " "  print "1) Addition"  print "2) Subtraction"  print "3) Multiplication"  print "4) Division"  print "5) Quit calculator"  print " "  return input ("Choose your option: ")    # this adds two numbers given  def add(a,b): #Defining function add  print a, "+", b, "=", a + b    # this subtracts two numbers given  def sub(a,b): #Defining function sub  print a, "-", b, "=", a - b    # this multiplies two numbers given  def mul(a,b): #Defining function mul  print a, "\*", b, "=", a \* b    # this divides two numbers given  def div(a,b): #Defining function div  print a, "/", b, "=", a / b    # NOW THE PROGRAM STARTS, AS CODE IS RUN  loop = 1  choice = 0  while loop == 1:  choice = menu()  if choice == 1:  num1=int(input("Enter num 1 --: "))  num2=int(input("Enter num 2 --: "))  add(num1,num2)  elif choice == 2:  num1=int(input("Enter num 1 --: "))  num2=int(input("Enter num 2 --: "))  sub(num1,num2)  elif choice == 3:  num1=int(input("Enter num 1 --: "))  num2=int(input("Enter num 2 --: "))  mul(num1,num2)  elif choice == 4:  num1=int(input("Enter num 1 --: "))  num2=int(input("Enter num 2 --: "))  div(num1,num2)  elif choice == 5:  loop = 0    # End of the program  --------------------------------------------------------------------------------------------------------------  **OUTPUT OF PROGRAM**  ----------------------------------------------------------------------------------------------------------------  Welcome to calculator in Python  your options are:    1) Addition  2) Subtraction  3) Multiplication  4) Division  5) Quit calculator    Choose your option: 1  Enter num 1 --: 21  Enter num 2 --: 2  21 + 2 = 23  Welcome to calculator in Python  your options are:    1) Addition  2) Subtraction  3) Multiplication  4) Division  5) Quit calculator    Choose your option: 2  Enter num 1 --: 23  Enter num 2 --: 2  23 - 2 = 21  Welcome to calculator in Python  your options are:    1) Addition  2) Subtraction  3) Multiplication  4) Division  5) Quit calculator    Choose your option: 3  Enter num 1 --: 23  Enter num 2 --: 1  23 \* 1 = 23  Welcome to calculator in Python  your options are:    1) Addition  2) Subtraction  3) Multiplication  4) Division  5) Quit calculator  Choose your option: 23  Welcome to calculator in Python  your options are:  1) Addition  2) Subtraction  3) Multiplication  4) Division  5) Quit calculator    Choose your option: 4  Enter num 1 --: 23  Enter num 2 --: 3  23 / 3 = 7  Welcome to calculator in Python  your options are:  1) Addition  2) Subtraction  3) Multiplication  4) Division  5) Quit calculator  Choose your option: 5 |
|  |

**Experiment No: 5**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Aim: Write python programs to understand Classes, object, Static method and inner class**  **Theory :**  **Overview of OOP Terminology**   * **Class** − A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation. * **Class variable** − A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are. * **Data member** − A class variable or instance variable that holds data associated with a class and its objects. * **Function overloading** − The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved. * **Instance variable** − A variable that is defined inside a method and belongs only to the current instance of a class. * **Inheritance** − The transfer of the characteristics of a class to other classes that are derived from it. * **Instance** − An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle. * **Instantiation** − The creation of an instance of a class. * **Method** − A special kind of function that is defined in a class definition. * **Object** − A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods. * **Operator overloading** − The assignment of more than one function to a particular operator.   **Creating Classes**  The *class* statement creates a new class definition. The name of the class immediately follows the keyword *class* followed by a colon as follows −  class ClassName:  'Optional class documentation string'  class\_suite   * The class has a documentation string, which can be accessed via *ClassName.\_\_doc\_\_*. * The *class\_suite* consists of all the component statements defining class members, data attributes and functions.   **Example**  Following is the example of a simple Python class −  class Employee:  'Common base class for all employees'  empCount = 0  def \_\_init\_\_(self, name, salary):  self.name = name  self.salary = salary  Employee.empCount += 1    def displayCount(self):  print "Total Employee %d" % Employee.empCount  def displayEmployee(self):  print "Name : ", self.name, ", Salary: ", self.salary   * The variable *empCount* is a class variable whose value is shared among all instances of a this class. This can be accessed as *Employee.empCount* from inside the class or outside the class. * The first method *\_\_init\_\_()* is a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class. * You declare other class methods like normal functions with the exception that the first argument to each method is *self*. Python adds the *self* argument to the list for you; you do not need to include it when you call the methods.   **Creating Instance Objects**  To create instances of a class, you call the class using class name and pass in whatever arguments its *\_\_init\_\_* method accepts.  "This would create first object of Employee class"  emp1 = Employee("Zara", 2000)  "This would create second object of Employee class"  emp2 = Employee("Manni", 5000)  **Accessing Attributes**  You access the object's attributes using the dot operator with object. Class variable would be accessed using class name as follows −  emp1.displayEmployee()  emp2.displayEmployee()  print "Total Employee %d" % Employee.empCount  Now, putting all the concepts together −  class Employee:  'Common base class for all employees'  empCount = 0  def \_\_init\_\_(self, name, salary):  self.name = name  self.salary = salary  Employee.empCount += 1    def displayCount(self):  print "Total Employee %d" % Employee.empCount  def displayEmployee(self):  print "Name : ", self.name, ", Salary: ", self.salary  "This would create first object of Employee class"  emp1 = Employee("Zara", 2000)  "This would create second object of Employee class"  emp2 = Employee("Manni", 5000)  emp1.displayEmployee()  emp2.displayEmployee()  print "Total Employee %d" % Employee.empCount  When the above code is executed, it produces the following result −  Name : Zara ,Salary: 2000  Name : Manni ,Salary: 5000  Total Employee 2  You can add, remove, or modify attributes of classes and objects at any time −  emp1.age = 7 # Add an 'age' attribute.  emp1.age = 8 # Modify 'age' attribute.  del emp1.age # Delete 'age' attribute.  Instead of using the normal statements to access attributes, you can use the following functions −   * The **getattr(obj, name[, default])** − to access the attribute of object. * The **hasattr(obj,name)** − to check if an attribute exists or not. * The **setattr(obj,name,value)** − to set an attribute. If attribute does not exist, then it would be created. * The **delattr(obj, name)** − to delete an attribute.   hasattr(emp1, 'age') # Returns true if 'age' attribute exists  getattr(emp1, 'age') # Returns value of 'age' attribute  setattr(emp1, 'age', 8) # Set attribute 'age' at 8  delattr(empl, 'age') # Delete attribute 'age'  **Built-In Class Attributes**  Every Python class keeps following built-in attributes and they can be accessed using dot operator like any other attribute −   * **\_\_dict\_\_** − Dictionary containing the class's namespace. * **\_\_doc\_\_** − Class documentation string or none, if undefined. * **\_\_name\_\_** − Class name. * **\_\_module\_\_** − Module name in which the class is defined. This attribute is "\_\_main\_\_" in interactive mode. * **\_\_bases\_\_** − A possibly empty tuple containing the base classes, in the order of their occurrence in the base class list.   class Employee:  'Common base class for all employees'  empCount = 0  def \_\_init\_\_(self, name, salary):  self.name = name  self.salary = salary  Employee.empCount += 1    def displayCount(self):  print "Total Employee %d" % Employee.empCount  def displayEmployee(self):  print "Name : ", self.name, ", Salary: ", self.salary  print "Employee.\_\_doc\_\_:", Employee.\_\_doc\_\_  print "Employee.\_\_name\_\_:", Employee.\_\_name\_\_  print "Employee.\_\_module\_\_:", Employee.\_\_module\_\_  print "Employee.\_\_bases\_\_:", Employee.\_\_bases\_\_  print "Employee.\_\_dict\_\_:", Employee.\_\_dict\_\_  **static method**  The class method in Python is a method, which is bound to the class but not the object of that class. The static methods are also same but there are some basic differences. For class methods, we need to specify @classmethod decorator, and for static method @staticmethod decorator is used.  **Syntax for Class Method.**  class my\_class:  @classmethod  deffunction\_name(cls, arguments):  #Function Body  return value  **Syntax for Static Method.**  class my\_class:  @staticmethod  deffunction\_name(arguments):  #Function Body  return value **Differences between Classmethod and StaticMehtod**  |  |  | | --- | --- | | **Class Method** | **Static Method** | | The class method takes cls (class) as first argument. | The static method does not take any specific parameter. | | Class method can access and modify the class state. | Static Method cannot access or modify the class state. | | The class method takes the class as parameter to know about the state of that class. | Static methods do not know about class state. These methods are used to do some utility tasks by taking some parameters. | | @classmethod decorator is used here. | @staticmethod decorator is used here. |   **Code 1**  #Demonstrate Class object static method and inner class  class Student:  roll=int(input('Enter your Roll Number \n'))  name=raw\_input('Enter your Name \n')  age=int(input('Enter your Age \n'))  gender=raw\_input('Enter your Gender \n')  print""  class address: #Inner Class  print "-------------Inner Class--------------"  print "This is Inner class"    @staticmethod #Static method  def statmeth():  print "This is Static method"  def display\_record(self):  print "Your Roll number is",self.roll  print "Your Name is",self.name  print "Your Age is",self.age  print "Your Gender is",self.gender  print""  s=Student() #Main class object  print""  print "-------------Static Method--------------"  s.statmeth() #calling static method  print""  print "-------------Details of Student are--------------"  s.display\_record() #Calling class method  **----------------------------------OUTPUT**----------------------------------------------------------------  Enter your Roll Number  1  Enter your Name  BHUSHAN NEMADE  Enter your Age  32  Enter your Gender  MALE  -------------Inner Class--------------  This is Inner class  -------------Static Method--------------  This is Static method  -------------Details of Student are--------------  Your Roll number is 1  Your Name is BHUSHAN NEMADE  Your Age is 32  Your Gender is MALE |

|  |
| --- |
| **Code 2 : Write python programs to understand Constructors**  #Demonstrating Constructor  class Student:  def \_\_init\_\_(self,n='',m1=0,m2=0,m3=0,m4=0,m5=0): #Constructor  self.name=raw\_input('Enter your name \n')  self.hmarks=int(input('Enter your History marks \n'))  self.mmarks=int(input('Enter your Maths marks \n'))  self.smarks=int(input('Enter your Science marks \n'))  self.gmarks=int(input('Enter your Geography marks \n'))  self.amarks=float(self.hmarks +self.mmarks+ self.smarks +self.gmarks) /4    def display\_record(self): #Instance method  print "Hello Mr.",self.name  print "Your Average marks are",self.amarks  if(self.amarks>=75):  print "You got Distinction class"  elif(self.amarks>=60):  print "You got First class"  elif(self.amarks>=50):  print "You got Second class"  elif(self.amarks>=40):  print "You got Pass class"  else:  print "You are failed"  s=Student()  print""  print "----------------RESULT--------------------"  s.display\_record()    ----------------------------------------------**OUTPUT**----------------------------------------------------  Enter your name  BHUSHAN NEMADE  Enter your History marks  87  Enter your Maths marks  83  Enter your Science marks  85  Enter your Geography marks  78  ----------------RESULT--------------------  Hello Mr. BHUSHAN NEMADE  Your Average marks are 83.25  You got Distinction class |

|  |
| --- |
| **Code 3: Write python programs to understand Inheritance and Polymorphism with Method overloading and Method Overriding**  class Student: # define parent class  def \_\_init\_\_(self):  print "Calling parent constructor"  def teacher(self): #Overridden Method (Method Over riding)  print "Overriding Teacher method in Parent Class"  def s\_details(self):  print ".........CALLING PARENT METHOD using Child Object......"  self.roll=raw\_input("Enter Your Roll Number. -: ")  self.name=raw\_input("Enter Your Name -: ")  self.div=raw\_input("Enter Your Division -: ")  self.city=raw\_input("Enter Your City -: ")  print " "  print "Student Details Are....."  print "Your Roll Number is",self.roll  print "Your Name is",self.name  print "Your Division is",self.div  print "Your City is",self.city  def sum(self, a=None,b=None,c=None):  print " "    print ".........METHOD OVERLOADING WITH a=10,b=20 and c=30......"    if a is not None and b is not None and c is not None :  print 'Sum of a,b and c is',a+b+c  elif a is not None and b is not None:  print 'Sum of a and b is',a+b  else:  print 'Sum Not possible..Value of a is',a  class College(Student): # define child class (Inheritance)  def \_\_init\_\_(self):  print "Calling child constructor"  def teacher(self): #Overridden Method (Method Over riding)  print " "  print ".........METHOD OVERRIDING:Overriding Teacher method in Child Class"    def c\_details(self):  print ".........CALLING CHILD METHOD using Child Object......"  self.cname=raw\_input("Enter Your College Name -: ")  self.uni=raw\_input("Enter Your University Name -: ")  self.caddress=raw\_input("Enter Your College City -: ")  print " "  print "Your College Details Are....."  print "Your College name is",self.cname  print "Your University is",self.uni  print "Your College City is",self.caddress  c = College() # instance of child  c.c\_details() # calls parent's method  c.s\_details() # calls parent's method  c.teacher() # calls Overridden method  c.sum(10,20,30) # Method overloading with three parameters  c.sum(10,20) # Method overloading with two parameters  c.sum(10) # Method overloading with one parameters  **-------------------------------------OUTPUT------------------------------------------------------**  Calling child constructor  .........CALLING CHILD METHOD using Child Object......  Enter Your College Name -: TCET  Enter Your University Name -: MUMBAI  Enter Your College City -: KANDIVALI  Your College Details Are.....  Your College name is TCET  Your University is MUMBAI  Your College City is KANDIVALI  .........CALLING PARENT METHOD using Child Object......  Enter Your Roll Number. -: 1  Enter Your Name -: BHUSHAN NEMADE  Enter Your Division -: A  Enter Your City -: THANE  Student Details Are.....  Your Roll Number is 1  Your Name is BHUSHAN NEMADE  Your Division is A  Your City is THANE  .........METHOD OVERRIDING:Overriding Teacher method in Child Class  .........METHOD OVERLOADING WITH a=10,b=20 and c=30......  Sum of a,b and c is 60  .........METHOD OVERLOADING WITH a=10,b=20 and c=30......  Sum of a and b is 30  .........METHOD OVERLOADING WITH a=10,b=20 and c=30......  Sum Not possible..Value of a is 10 |
| **Experiment 6**  **Aim : Write python program to understand different types of Exceptions**  **Theory:**  **What are exceptions in Python?**  Python has many built-in exceptions which forces your program to output an error when something in it goes wrong.  When these exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled. If not handled, our program will crash.  For example, if function A calls function B which in turn calls function C and an exception occurs in function C. If it is not handled in C, the exception passes to B and then to A. If never handled, an error message is spit out and our program come to a sudden, unexpected halt.  **Catching Exceptions in Python**  In Python, exceptions can be handled using a try statement. A critical operation which can raise exception is placed inside the try clause and the code that handles exception is written in except clause.  It is up to us, what operations we perform once we have caught the exception. Here is a simple example.  # import module sys to get the type of exception  import sys  randomList = ['a', 0, 2]  for entry in randomList:  try:  print("The entry is", entry)  r = 1/int(entry)  break  except:  print("Oops!",sys.exc\_info()[0],"occured.")  print("Next entry.")  print()  print("The reciprocal of",entry,"is",r)  **Output:**  The entry is a  Oops! <class 'ValueError'> occured.  Next entry.  The entry is 0  Oops! <class 'ZeroDivisionError' > occured.  Next entry.    The entry is 2  The reciprocal of 2 is 0.5  **I**n this program, we loop until the user enters an integer that has a valid reciprocal. The portion that can cause exception is placed inside try block.If no exception occurs, except block is skipped and normal flow continues. But if any exception occurs, it is caught by the except block.  Here, we print the name of the exception using ex\_info() function inside sys module and ask the user to try again. We can see that the values 'a' and '1.3' causes ValueError and '0' causes ZeroDivisionError.  **Catching Specific Exceptions in Python**  In the above example, we did not mention any exception in the except clause.This is not a good programming practice as it will catch all exceptions and handle every case in the same way. We can specify which exceptions an except clause will catch.A try clause can have any number of except clause to handle them differently but only one will be executed in case an exception occurs. We can use a tuple of values to specify multiple exceptions in an except clause.  **pseudo code:**  try:  # do something  pass  except ValueError:  # handle ValueError exception  pass  except (TypeError, ZeroDivisionError):  # handle multiple exceptions  # TypeError and ZeroDivisionError  pass  except:  # handle all other exceptions  pass  **Raising Exceptions**  In Python programming, exceptions are raised when corresponding errors occur at run time, but we can forcefully raise it using the keyword raise.  We can also optionally pass in value to the exception to clarify why that exception was raised.  >>> raise KeyboardInterrupt  Traceback (most recent call last):  ...  KeyboardInterrupt  >>> raise MemoryError("This is an argument")  Traceback (most recent call last):  ...  MemoryError: This is an argument  >>> try:  ... a = int(input("Enter a positive integer: "))  ... if a <= 0:  ... raise ValueError("That is not a positive number!")  ... except ValueError as ve:  ... print(ve)  ...  Enter a positive integer: -2  That is not a positive number!  **try...finally**  The try statement in Python can have an optional finally clause. This clause is executed no matter what, and is generally used to release external resources.  For example, we may be connected to a remote data center through the network or working with a file or working with a Graphical User Interface (GUI).  In all these circumstances, we must clean up the resource once used, whether it was successful or not. These actions (closing a file, GUI or disconnecting from network) are performed in the finally clause to guarantee execution.  **Code :**  try:  f = open("test.txt",encoding = 'utf-8')  # perform file operations  finally:  f.close()  This type of construct makes sure the file is closed even if an exception occurs.  **Code :**  i=1  while i<=5:  n=int(input("Please enter numbers between 1 to 5 to see different Exceptions : "))  if n==1:  try:  a=int(input("Please enter number a : "))  b=int(input("Please enter number b (put b=0): "))  c=a/b  except ZeroDivisionError:  print("Oops! Number Divisible by Zero Exception Occurs.")  else:  print "Division is",c  elif n==2:  try:  a=int(input("Please enter number a : "))  b=int(input("Please enter number b (put b='a'): "))  c=a/b  except ValueError:  print("Oops! Value Error Exception Occurs.Please enter a valid number.")  else:  print "Division is",c    elif n==3:  try:  a=int(input("Please enter number a : "))  b=int(input("Please enter number b : "))  c=k/b  except NameError:  print("Oops! Name Error Exception Occurs due to c=k/b (k is not defined ).Please enter a valid variable number.")  elif n==4:  try:  r='2'+2  except TypeError:  print("Oops! Type Error Exception Occurs (due to '2'+2).Please Provide Valid data type. ")    elif n==5:  try:  n=int(input("Please enter Numbers between 2 to 3: (Check for other nos) "))  assert n>=2 and n<=3  print("The Number Entered is",n)  except AssertionError:  print("Oops! Assertion Error Occurs..Please enter number between 2 to 5.")    else:  print "Existing The Program"  exit()  i+=1  **--------------------------------------------OUTPUT-----------------------------------------------------**  Please enter numbers between 1 to 5 to see diffrent Exceptions : 1  Please enter number a : 12  Please enter number b (put b=0): 0  Oops! Number Divisible by Zero Exception Occurs.  Please enter numbers between 1 to 5 to see different Exceptions : 2  Please enter number a : 24  Please enter number b (put b='a'): 'a'  Oops! Value Error Exception Occurs.Please enter a valid number.  Please enter numbers between 1 to 5 to see different Exceptions : 3  Please enter number a : 13  Please enter number b : 45  Oops! Name Error Exception Occurs due to c=k/b (k is not defined ).Please enter a valid variable number.  Please enter numbers between 1 to 5 to see different Exceptions : 4  Oops! Type Error Exception Occurs (due to '2'+2).Please Provide Valid data type.  Please enter numbers between 1 to 5 to see different Exceptions : 5  Please enter Numbers between 2 to 3: (Check for other nos) 6  Oops! Assertion Error Occurs. Please enter number between 2 to 5.  Please enter numbers between 1 to 5 to see different Exceptions : 6  Existing The Program |

**Experiment 7**

|  |
| --- |
| **Aim: Write python programs to understand different file handling operations with pickle**  **Theory:**  **File Handling in Python**  Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but alike other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let’s start with Reading and Writing files.  **Working of open() function**  We use open () function in Python to open a file in read or write mode. As explained above, open ( ) will return a file object. To return a file object we use open() function along with two arguments, that accepts file name and the mode, whether to read or write. So, the syntax being: open(filename, mode). There are three kinds of mode, that Python provides and how files can be opened:  “ r “, for reading.  “ w “, for writing.  “ a “, for appending.  “ r+ “, for both reading and writing  One must keep in mind that the mode argument is not mandatory. If not passed, then Python will assume it to be “ r ” by default. Let’s look at this program and try to analyze how the read mode works:  # a file named "geek", will be opened with the reading mode.  file = open('geek.txt', 'r')  # This will print every line one by one in the file  for each in file:  print (each)  The open command will open the file in the read mode and the for loop will print each line present in the file.  **Working of read() mode**  There is more than one way to read a file in Python. If you need to extract a string that contains all characters in the file then we can use file.read(). The full code would work like this:  **# Python code to illustrate read() mode**  file = open("file.text", "r")  print file.read()  Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:  **# Python code to illustrate read() mode character wise**  file = open("file.txt", "r")  print file.read(5)  **Creating a file using write() mode**  Let’s see how to create a file and how write mode works:  To manipulate the file, write the following in your Python environment:  **# Python code to create a file**  file = open('geek.txt','w')  file.write("This is the write command")  file.write("It allows us to write in a particular file")  file.close()  The close() command terminates all the resources in use and frees the system of this particular program.  **Code Example:**  **Filename-: studentpickle.py**  class student:  def \_\_init\_\_(self,roll,name,age):  self.roll=roll  self.name=name  self.age=age  def display(self):  print("{:5d}{:20s}{:10.2f}".format(self.roll,self.name,self.age))  **Code:**  import pickle,studentpickle  print "\*\*\*\*\*\*WRITE file\*\*\*\*\*\*\*\*\*\*\*\*"  f=open('myfile.txt','w') #Create file if not exist and open it for writting  s1=raw\_input('Enter Your Text to be written in to file: ')  f.write(s1) #write text into file  f.close() #Close file  print""  print "\*\*\*\*\*\*Read file \*\*\*\*\*\*\*\*\*\*\*\*"  f1=open('myfile.txt','r') #open file for reading  print "Contents of file are"  s=f1.read()  print s  f1.close()#Close file  print""  print "\*\*\*\*\*\*Append file\*\*\*\*\*\*\*\*\*\*\*\*"  f=open('myfile.txt','a') #open file for Appending  s2=raw\_input('Enter text to be appended in original file..')  f.write(s2)  f.close() #Close file  print""  print ""  print "\*\*\*\*\*\*Read Appended file\*\*\*\*\*\*\*\*\*\*\*\*"  f1=open('myfile.txt','r') #Create file if not exist and open it for reading  s=f1.read()  print s  f.close()  print "\*\*\*\*\*\*WITH OPEN\*\*\*\*\*\*\*\*\*\*\*\*"  with open('sample.txt','w') as f:  s3=raw\_input('Enter text for WITHOPEN write..')  f.write(s3)  with open('sample.txt','r') as f:  for line in f:  print line  print "\*\*\*\*\*\*PICKLE\*\*\*\*\*\*\*\*\*\*\*\*"  f=open('studentpickle.dat','wb')  n=int(input('How many students?? '))  for i in range (n):  roll=int(input("Enter your roll no : "))  name=raw\_input("Enter your name : ")  age=int(input("Enter your age : "))  s=studentpickle.student(roll,name,age)  pickle.dump(s,f)  f.close()  ---------------------------------------------------------------------------------------------------------------  **OUTPUT**  ----------------------------------------------------------------------------------------------------------------  \*\*\*\*\*\*WRITE file\*\*\*\*\*\*\*\*\*\*\*\*  Enter Your Text to be written in to file: BHUSHAN  \*\*\*\*\*\*Read file \*\*\*\*\*\*\*\*\*\*\*\*  Contents of file are  BHUSHAN  \*\*\*\*\*\*Append file\*\*\*\*\*\*\*\*\*\*\*\*  Enter text to be appended in original file.. NEMADE  \*\*\*\*\*\*Read Appended file\*\*\*\*\*\*\*\*\*\*\*\*  BHUSHAN NEMADE  \*\*\*\*\*\*WITH OPEN\*\*\*\*\*\*\*\*\*\*\*\*  Enter text for WITHOPEN write.. PYTHON  PYTHON  \*\*\*\*\*\*PICKLE\*\*\*\*\*\*\*\*\*\*\*\*  How many students?? 1  Enter your roll no : 1  Enter your name : BHUSHAN  Enter your age : 32  C:\Users\Tsec\Desktop\My experiments\Experiment 4\4.1\Screenshot from 2017-12-31 04-30-53.png |

**Experiment 8**

|  |
| --- |
| **Aim: Write python programs to understand Lambda, map, reduce, filter and range functions**  **Theory:**  **Python lambda (Anonymous Functions) | filter, map, reduce**  In Python, anonymous function means that a function is without a name. As we already know that def keyword is used to define the normal functions and the lambda keyword is used to create anonymous functions. It has the following  **Syntax:**  **lambda arguments: expression**  This function can have any number of arguments but only one expression, which is evaluated and returned.  One is free to use lambda functions wherever function objects are required.  You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.  It has various uses in particular fields of programming besides other types of expressions in functions.  Let’s look at this example and try to understand the difference between a normal def defined function and lambda function. This is a program that returns the cube of a given value:  **Code:**  # Python code to illustrate cube of a number  # showing difference between def() and lambda().  def cube(y):  return y\*y\*y;  g = lambda x: x\*x\*x  print(g(7))  print(cube(5))  Output:  343  125  Without using Lambda : Here, both of them returns the cube of a given number. But, while using def, we needed to define a function with a name cube and needed to pass a value to it. After execution, we also needed to return the result from where the function was called using the return keyword.  Using Lambda : Lambda definition does not include a “return” statement, it always contains an expression which is returned. We can also put a lambda definition anywhere a function is expected, and we don’t have to assign it to a variable at all. This is the simplicity of lambda functions.  Lambda functions can be used along with built-in functions like filter(), map() and reduce().  **Use of lambda() with filter()**  The filter() function in Python takes in a function and a list as arguments. This offers an elegant way to filter out all the elements of a sequence “sequence”, for which the function returns True. Here is a small program that returns the odd numbers from an input list:  # Python code to illustrate  # filter() with lambda()  li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]  final\_list = list(filter(lambda x: (x%2 != 0) , li))  print(final\_list)  [5, 7, 97, 77, 23, 73, 61]    **Use of lambda() with map()**  The map() function in Python takes in a function and a list as argument. The function is called with a lambda function and a list and a new list is returned which contains all the lambda modified items returned by that function for each item.  Example:  # Python code to illustrate  # map() with lambda()  # to get double of a list.  li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]  final\_list = list(map(lambda x: x\*2 , li))  print(final\_list)  Output: [10, 14, 44, 194, 108, 124, 154, 46, 146, 122]  **Use of lambda() with reduce()**  The reduce() function in Python takes in a function and a list as argument. The function is called with a lambda function and a list and a new reduced result is returned. This performs a repetitive operation over the pairs of the list. This is a part of functools module.  **Example:**  # Python code to illustrate  # reduce() with lambda()  # to get sum of a list  from functools import reduce  li = [5, 8, 10, 20, 50, 100]  sum = reduce((lambda x, y: x + y), li)  print (sum)  **Output: 193**  Here the results of previous two elements are added to the next element and this goes on till the end of the list like (((((5+8)+10)+20)+50)+100).  **Code:**  "------------------------Lambda function--------------"  print "Lambda Function"  f = lambda x, y : x + y  print f(1,1)  print ""  "------------------------Map function--------------"  print "Map Function"  def fahrenheit(T):  return ((float(9)/5)\*T + 32)  def celsius(T):  return (float(5)/9)\*(T-32)  temp = (36.5, 37, 37.5,39)  F = map(fahrenheit, temp)  C = map(celsius, F)  print F  print C  print ""  "-----------------------Reduce function--------------"  print "Reduce Function"  print reduce(lambda x,y: x+y, [1,2,3,4])  print ""  "------------------------Filter function--------------"  print "Filter Function"  fib = [0,1,1,2,3,5,8,13,21,34,55]  result = filter(lambda x: x % 2, fib)  print result  print ""  "------------------------Range function--------------"  print "Range Function"  print range(10)  ---------------------------------------------------------------------------------------------------------------  **OUTPUT**  ----------------------------------------------------------------------------------------------------------------  Lambda Function  2  Map Function  [97.7, 98.60000000000001, 99.5, 102.2]  [36.5, 37.00000000000001, 37.5, 39.0]  Reduce Function  10  Filter Function  [1, 1, 3, 5, 13, 21, 55]  Range Function  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] |

**Experiment No: 9**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Aim: Write python programs to understand GUI Canvas Application using Tkinter**  **Theory:**  **Python Tkinter Canvas**  The canvas widget is used to add the structured graphics to the python application. It is used to draw the graph and plots to the python application. The syntax to use the canvas is given below.  **Syntax**   1. w = canvas(parent, options)   **A list of possible options is given below**   |  |  |  | | --- | --- | --- | | **SN** | **Option** | **Description** | | 1 | bd | The represents the border width. The default width is 2. | | 2 | bg | It represents the background color of the canvas. | | 3 | confine | It is set to make the canvas unscrollable outside the scroll region. | | 4 | cursor | The cursor is used as the arrow, circle, dot, etc. on the canvas. | | 5 | height | It represents the size of the canvas in the vertical direction. | | 6 | highlightcolor | It represents the highlight color when the widget is focused. | | 7 | relief | It represents the type of the border. The possible values are SUNKEN, RAISED, GROOVE, and RIDGE. | | 8 | scrollregion | It represents the coordinates specified as the tuple containing the area of the canvas. | | 9 | width | It represents the width of the canvas. | | 10 | xscrollincrement | If it is set to a positive value. The canvas is placed only to the multiple of this value. | | 11 | xscrollcommand | If the canvas is scrollable, this attribute should be the .set() method of the horizontal scrollbar. | | 12 | yscrollincrement | Works like xscrollincrement, but governs vertical movement. | | 13 | yscrollcommand | If the canvas is scrollable, this attribute should be the .set() method of the vertical scrollbar. |   **Code 1:**  from Tkinter import \*  root=Tk()  root.title("My First Tkinter Application") #Giving a title to bar  c=Canvas(root,bg="orange",height=700, width=800) #Creating a canvas  id=c.create\_line(80,80,200,80,200,200,width=6,fill="cyan3")#Drawing a Line  id=c.create\_line(80,80,300,80,300,300,width=6,fill="cyan3")#Drawing a Line  id=c.create\_oval(120,120,400,300,width=6,fill="red",outline="cornsilk1",activefill="green")#Drawing a Oval  fnt=('Times',22,'bold italic underline') #Setting Font  id=c.create\_text(250,40,text="My first Tkinter Application on canvas",font=fnt,fill="red",activefill="green") #Drawing a Text  id=c.create\_polygon(320,320,320,420,420,320,width=6,fill="yellow",outline="cornsilk2",activefill="lightblue")#Drawing Polygon  id=c.create\_rectangle(360,460,550,550,width=6,fill="brown",outline="cornsilk3",activefill="chocolate")#Drawing Rectangle  id=c.create\_arc(500,100,600,300,width=6,start=0, extent=180,outline="white",style="arc")#Drawing arc  file1=PhotoImage(file="cat.gif",height=100,width=100)  file2=PhotoImage(file="dog.gif",height=100,width=100)  id=c.create\_image(300,400,anchor=NE,image=file2,activeimage=file1)#Drawing Image  c.pack()  root.mainloop()  **-----------------------------------------------OUTPUT----------------------------------------------------**  **C:\Users\Tsec\Desktop\My experiments\Experiment 5\5.1\5.1.png** |

|  |  |
| --- | --- |
| **Code 2: Write python programs to understand GUI Frame Application using Tkinter**  from Tkinter import \*  class MyEntry:  def \_\_init\_\_(self,root):  self.f=Frame(root,height=400, width=600)  self.f.propagate(0)  self.f.pack()  self.l1=Label(text='Enter Your Name') #Create Label name  self.l2=Label(text='Enter Your age') #Create Label password  self.l3=Label(text='Select Your City') #Create Label address  self.l4=Label(text='Select Your Gender') #Create Label Gender  self.l5=Label(text='Select Favourite games') #Create Label Gender  self.e1=Entry(self.f,width=18,fg="black",bg="gray", font=('arial',12))#Add text entry widget for name    self.e2=Entry(self.f,width=18,fg="black",bg="gray", font=('arial',12),show="\*") #Add text entry widget for address  self.val1=StringVar()  self.s1=Spinbox(self.f,values=('Mumbai','Delhi','Kolkata','Chennai'),textvariable=self.val1,width=15,fg='black',bg='gray',font=('Arial',12,'bold italic'))  self.var=IntVar()  self.r1=Radiobutton(self.f,text='Male',variable=self.var,value=1,command=self.display) #Add radiobutton  self.r2=Radiobutton(self.f,text='Female',variable=self.var,value=2,command=self.display) #Add radiobutton  self.var1=IntVar()  self.var2=IntVar()  self.var3=IntVar()  self.c1=Checkbutton(self.f,text='Cricket',variable=self.var1,command=self.display) #Add Checkbox  self.c2=Checkbutton(self.f,text='Football',variable=self.var2,command=self.display) #Add Checkbox  self.c3=Checkbutton(self.f,text='Tennis',variable=self.var3,command=self.display) #Add Checkbox  self.b=Button(self.f,text="Print Details",command=self.display)#Add Button  #Place Labels,entry widget,Buttons,Spinbox on the frame  self.l1.place(x=50,y=30)  self.e1.place(x=200,y=30)  self.l2.place(x=50,y=60)  self.e2.place(x=200,y=60)  self.l3.place(x=50,y=90)  self.s1.place(x=200,y=90)  self.l4.place(x=50,y=130)  self.r1.place(x=190,y=130)  self.r2.place(x=260,y=130)  self.l5.place(x=50,y=160)  self.c1.place(x=190,y=160)  self.c2.place(x=270,y=160)  self.c3.place(x=360,y=160)  self.b.place(x=140,y=180)  def display(self):  str1=self.e1.get()  str2=self.e2.get()  str3=self.s1.get()  x1=self.var1.get()  y=self.var2.get()  z=self.var3.get()  x=self.var.get()  str4=''  if x==1:  str4+='Your Gender is Male'  if x==2:  str4+='Your Gender is Female'  str5=''  if x1==1:  str5+='Cricket'  if y==1:  str5+='Football'  if z==1:  str5+='Tennis'  l7=Label(text='Your Name is: '+str1,font=('Verdana',10,'bold'),fg='red').place(x=50,y=220)#label for entry widget name  l8=Label(text='Your Age is: '+str2,font=('Verdana',10,'bold'),fg='red').place(x=50,y=250)#label for entry widget age  l9=Label(text='Your City is: '+str3,font=('Verdana',10,'bold'),fg='red').place(x=50,y=280)#label for Spinbox  l10=Label(text=str4,font=('Verdana',10,'bold'),fg='red').place(x=50,y=310) #label for radio button  l11=Label(text='Your favourite games are '+str5,font=('Verdana',10,'bold'),fg='red').place(x=50,y=340) #label for Checkbox  root=Tk()#create root window  root.title("Frame Tkinter Application") #Giving a title to bar  mb=MyEntry(root)#creating an object to class  root.mainloop()#root window handles the mouse events  C:\Users\Tsec\Desktop\My experiments\Experiment 5\5.1\5.2.1.png  **Code 3: Application 2**  from Tkinter import \*  fields = 'First Name', 'Last Name', 'Gender','City','Job', 'Country','State',  def fetch(entries):  for entry in entries:  field = entry[0]  text = entry[1].get()  print('%s: "%s"' % (field, text))  def makeform(root, fields):  entries = []  for field in fields:  row = Frame(root)  lab = Label(row, width=15, text=field, anchor='w')  ent = Entry(row)  row.pack(side=TOP, fill=X, padx=5, pady=5)  lab.pack(side=LEFT)  ent.pack(side=RIGHT, expand=YES, fill=X)  entries.append((field, ent))  return entries  if \_\_name\_\_ == '\_\_main\_\_':  root = Tk()  root.title("Tkinter FRAME Application")  ents = makeform(root, fields)  root.bind('<Return>', (lambda event, e=ents: fetch(e)))  b1 = Button(root, text='Show',  command=(lambda e=ents: fetch(e)))  b1.pack(side=LEFT, padx=5, pady=5)  b2 = Button(root, text='Quit', command=root.quit)  b2.pack(side=LEFT, padx=5, pady=5)  root.mainloop()  **------------------------------------------------OUTPUT----------------------------------------------------**  **C:\Users\Tsec\Desktop\My experiments\Experiment 5\5.1\5.2.2.png**  **Experiment 10**  **Aim: Write python programs to understand CRUD Operations using Mysql Python Database Connectivity**  **Theory :**  **Steps to Perform CRUD Operations Python Mysql Database Connectivity**  **Step 1: Install Mysql in Ubuntu**  First you must install a MySQL driver, use the specific installation method below  pythonlab@ubuntu:~$ sudo apt-get install mysql-server  pythonlab@ubuntu:~$sudo apt-get install python-mysqldb  pythonlab@ubuntu:~$ sudo apt-get install python-pip python-dev libmysqlclient-dev  **Step 2: Setup the database**  Make sure you have database access, from the command line type:  pythonlab@ubuntu:~$ mysql –u root -p  MySQL will then ask your password put the password ‘python’.  Execute the following commands to create database TESTDB, Create table EMPLOYEE and Insert Records using following commands:   |  | | --- | | mysql> create database TESTDB  -> ;  Query OK, 1 row affected (0.08 sec)  mysql> USE TESTDB  Database changed |   mysql> CREATE TABLE EMPLOYEE (  -> FIRST\_NAME CHAR(20) NOT NULL,  -> LAST\_NAME CHAR(20),  -> AGE INT,  -> SEX CHAR(1),  -> INCOME FLOAT )  -> ;  mysql> INSERT INTO EMPLOYEE(FIRST\_NAME,LAST\_NAME, AGE, SEX, INCOME) VALUES ('RAJESH','GANGULY',28,'M',93776);  Query OK, 1 row affected (0.06 sec)  mysql> INSERT INTO EMPLOYEE(FIRST\_NAME,LAST\_NAME, AGE, SEX, INCOME) VALUES ('SACHIN','TENDULKAR',40,'M',98463);  Query OK, 1 row affected (0.15 sec)  mysql> INSERT INTO EMPLOYEE(FIRST\_NAME,LAST\_NAME, AGE, SEX, INCOME) VALUES ('ANIL','KUMBLE',42,'M',64300);  Query OK, 1 row affected (0.07 sec)  mysql> INSERT INTO EMPLOYEE(FIRST\_NAME,LAST\_NAME, AGE, SEX, INCOME) VALUES ('RAHUL','DRAVID',39,'M',84300);  Query OK, 1 row affected (0.07 sec)  mysql> INSERT INTO EMPLOYEE(FIRST\_NAME,LAST\_NAME, AGE, SEX, INCOME) VALUES ('YUVRAJ','SINGH',28,'M',94300);  Query OK, 1 row affected (0.17 sec)  mysql> INSERT INTO EMPLOYEE(FIRST\_NAME,LAST\_NAME, AGE, SEX, INCOME) VALUES ('VIRAT','KOHLI',26,'M',98343);  Query OK, 1 row affected (0.15 sec)  mysql> INSERT INTO EMPLOYEE(FIRST\_NAME,LAST\_NAME, AGE, SEX, INCOME) VALUES ('BHUSHAN','NEMADE',29,'M',94433);  Query OK, 1 row affected (0.00 sec)  mysql> select \* from EMPLOYEE;  +------------+-----------+------+------+--------+  | FIRST\_NAME | LAST\_NAME | AGE | SEX | INCOME |  +------------+-----------+------+------+--------+  | RAJESH | GANGULY | 28 | M | 93776 |  | SACHIN | TENDULKAR | 40 | M | 98463 |  | ANIL | KUMBLE | 42 | M | 64300 |  | RAHUL | DRAVID | 39 | M | 84300 |  | YUVRAJ | SINGH | 28 | M | 94300 |  | VIRAT | KOHLI | 26 | M | 98343 |  | BHUSHAN | NEMADE | 29 | M | 94433 |  +------------+-----------+------+------+--------+  7 rows in set (0.00 sec)  Screenshot from 2018-01-01 11-30-25  Screenshot from 2018-01-01 12-32-37  **Step 2:Run the Python Program**  **--------------------------------------Python Program----------------------------------------------------**  #Mysql CRUD Operations using Python  import MySQLdb  # Open database connection  db = MySQLdb.connect("localhost","root","python","TESTDB" )  # prepare a cursor object using cursor() method  cursor = db.cursor()  print "----------Insert record----------------"  # Prepare SQL query to INSERT a record into the database.  a=raw\_input('Enter u r First name: ')  b=raw\_input('Enter u r Last name: ')  c=int(input('Enter u r Age: '))  d=raw\_input('Enter u r Sex: ')  e=int(input('Enter u r Income: '))  sql = "INSERT INTO EMPLOYEE(FIRST\_NAME, \  LAST\_NAME, AGE, SEX, INCOME) \  VALUES ('%s', '%s', '%d', '%c', '%d' )" % \  (a, b, c, d, e)  try:  # Execute the SQL command  cursor.execute(sql)  # Commit your changes in the database  db.commit()  except:  # Rollback in case there is any error  db.rollback()  print "----------View records-----------"  sql = "SELECT \* FROM EMPLOYEE"  try:  # Execute the SQL command  cursor.execute(sql)  # Fetch all the rows in a list of lists.  results = cursor.fetchall()  for row in results:  fname = row[0]  lname = row[1]  age = row[2]  sex = row[3]  income = row[4]  # Now print fetched result  print "fname=%s,lname=%s,age=%d,sex=%s,income=%d" % \  (fname, lname, age, sex, income )  except:  print "Error: unable to fecth data"  print "----------Update records with age+1 For males-----------"  # Prepare SQL query to UPDATE required records  sql = "UPDATE EMPLOYEE SET AGE=AGE+1 WHERE SEX = '%c'" % ('M')  try:  # Execute the SQL command  cursor.execute(sql)  # Commit your changes in the database  db.commit()  except:  # Rollback in case there is any error  db.rollback()  print "----------Delete records with age > 30-----------"  sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (30)  try:  # Execute the SQL command  cursor.execute(sql)  # Commit your changes in the database  db.commit()  except:  # Rollback in case there is any error  db.rollback()  print "----------View records-----------"  sql = "SELECT \* FROM EMPLOYEE"  try:  # Execute the SQL command  cursor.execute(sql)  # Fetch all the rows in a list of lists.  results = cursor.fetchall()  for row in results:  fname = row[0]  lname = row[1]  age = row[2]  sex = row[3]  income = row[4]  # Now print fetched result  print "fname=%s,lname=%s,age=%d,sex=%s,income=%d" % \  (fname, lname, age, sex, income )  except:  print "Error: unable to fecth data"  # disconnect from server  db.close()  **--------------------------------------OUTPUT--------------------------------------------------------------**  pythonlab@ubuntu:~$ python 5.3.py  ----------Insert record----------------  Enter u r First name: ROHIT  Enter u r Last name: SHARMA  Enter u r Age: 27  Enter u r Sex: M  Enter u r Income: 35432  ----------View records-----------  fname=RAJESH,lname=GANGULY,age=28,sex=M,income=93776  fname=SACHIN,lname=TENDULKAR,age=40,sex=M,income=98463  fname=ANIL,lname=KUMBLE,age=42,sex=M,income=64300  fname=RAHUL,lname=DRAVID,age=39,sex=M,income=84300  fname=YUVRAJ,lname=SINGH,age=28,sex=M,income=94300  fname=VIRAT,lname=KOHLI,age=26,sex=M,income=98343  fname=BHUSHAN,lname=NEMADE,age=29,sex=M,income=94433  fname=ROHIT,lname=SHARMA,age=27,sex=M,income=35432  ----------Update records with age+1 For males-----------  ----------Delete records with age > 30-----------  ----------View records-----------  fname=RAJESH,lname=GANGULY,age=29,sex=M,income=93776  fname=YUVRAJ,lname=SINGH,age=29,sex=M,income=94300  fname=VIRAT,lname=KOHLI,age=27,sex=M,income=98343  fname=BHUSHAN,lname=NEMADE,age=30,sex=M,income=94433  fname=ROHIT,lname=SHARMA,age=28,sex=M,income=35432  pythonlab@ubuntu:~$  Screenshot from 2018-01-01 12-56-20 |

|  |
| --- |
| POST LAB EXPERIMENTS |
| 1. **Introduction** 2. Python Program to Calculate the Area of a Triangle ,square and rectangle 3. Python Program to Solve Quadratic Equations 4. Python Program to Swap Two Variables 5. Python Program to Generate a Random Number 6. Python Program to Convert Kilometers to Miles, Celsius To Fahrenheit and other conversions 7. **Decision making and Loops** 8. Python Program to Check if a Number is Positive, Negative or 0 9. Python Program to Check if a Number is Odd or Even 10. Python Program to Check Leap Year 11. Python Program to Find the Largest Among Three Numbers 12. Python Program to Check Prime Number 13. Python Program to Print all Prime Numbers in an Interval 14. Python Program to Find the Factorial of a Number 15. Python Program to Display the multiplication Table 16. Python Program to Print the Fibonacci sequence 17. Python Program to Check Armstrong Number 18. Python Program to Find Armstrong Number in an Interval 19. Python Program to Find the Sum of Natural Numbers 20. WAP to count number of vowels and consonants from the given strings 21. Python program that prints all real solutions to the quadratic equation ax2+bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac is negative, display a message stating that there are no real solutions. 22. Python program to demonstrate sorting algorithms using List or Arrays 23. Python program to demonstrate searching algorithms using List, Arrays or touples 24. **Functions** 25. Python Program To Display Powers of 2 Using Anonymous Function 26. Python Program to Find Numbers Divisible by Another Number 27. Python Program to Convert Decimal to Binary, Octal and Hexadecimal 28. Python Program to Find ASCII Value of Character 29. Python Program to Find HCF or GCD 30. Python Program to Find LCM 31. Python Program to Find Factors of Number 32. Python Program to Make a Simple Calculator 33. Python Program to Shuffle Deck of Cards 34. Python Program to Display Calendar 35. Python Program to Display Fibonacci Sequence Using Recursion 36. Python Program to Find Sum of Natural Numbers Using Recursion 37. Python Program to Find Factorial of Number Using Recursion 38. Python Program to Convert Decimal to Binary Using Recursion 39. Write a Menu driven program in python to implement simple banking application. Application should read the customer name, account number, initial balance, rate of interest, contact number and address field etc. Application should have following methods.   1. createAccount()  2. deposit()  3. withdraw()  4. computeInterest()  5. displayBalance()  16) Write a menu driven code in python which will read a number and should implement the following methods  1. factorial()  2. reverse()  3. testArmstrong()  4. testPalindrome()  5. testPrime()  6. fibonacciSeries()   1. **Native Datatypes** 2. Python Program to Add Two Matrices 3. Python Program to Transpose a Matrix 4. Python Program to Multiply Two Matrices 5. Python Program to Check Whether a String is Palindrome or Not 6. Python Program to Remove Punctuations From a String 7. Python Program to Sort Words in Alphabetic Order 8. Python Program to Illustrate Different Set Operations 9. Python Program to Count the Number of Each Vowel 10. **File handling** 11. Python Program to Merge Mails 12. Python Program to or delete file ,copy file or move file from one directory in to other directory 13. Python Program to Find Hash of File 14. **GUI and Database handling using Python**   1) WAP to draw the house on an canvas using Tkinter.  2) On Tkinter Form: Take a Login and Password from the user and display it on the labels which appears only on clicking OK button and clear both the labels on clicking RESET button    3) WAP to create a registration form on an Tkinter Form with Mysql Databases.  4) To display all the employee records in the emp table  5) Update the salary of all the employees in the emp table with an addition of 1000 Rs  6) To get the employee id from the user input and display of the details of employees in the emp table  7) create the following tables   |  |  | | --- | --- | | **EMP**  Eno  Ename  Basic  Hra  Da  Pf  Dno | **DEPT**  Dno  Dname |  1. Add the records to the above tables 2. Display a report which shows all the above listed fields grouped by their DNO 3. Calculate the gross salary of individual and display it. 4. Calculate department wise total, grand total and display it. 5. **Networking** 6. WAP to copy a file from client to server using tcp socket 7. WAP for sending a email 8. WAP to find out which of the first 1024 ports seem to be hosting TCP servers 9. WAP to develop chat application 10. WAP that implements socket server to communicate with socket client. Also include logging capability to access pop server |

|  |  |
| --- | --- |
| LIST OF MINI PROJECT TOPICS | |
| 1. Railway reservation system  2. Inventory Management system.  3 Classroom Management  4 Clinical Trial Initiation and Management  5 Competitive Analysis Web Site  6 Discussion Forum website  7 Disputed Invoice Management  8 Employee Training Scheduling and Materials  9 Equity Research Management  10 Integrated Marketing Campaign Tracking  11 Manufacturing Process Managements  12 Product and Marketing Requirements Planning  13 Request for Proposal Software  14 Sports League Management  15 Absence Request and Vacation Schedule Management  16 Budgeting and Tracking Multiple Projects  17 Bug Database Management  18 Call Center Management Software  19 Change Request Management  20 Compliance Process Support Site  21 Contacts Management Software  22 Document Library and Review  23 Event Planning and Management  24 Expense Reimbursement and Approval  25 Help Desk and Ticket Management  26 Inventory Tracking  27 I T Team Workspace  29 Job Requisition and Interview Management  28 Knowledge Base  29 Lending Library  30 Physical Asset Tracking and Management  31 Project Tracking Workspace  32. Shopping Cart .  33 Knowledge Base  34 Lending Library  35 Physical Asset Tracking and Management  36 Project Tracking Workspace  37 Room and Equipment Reservations  38 Sales Lead Pipeline  39. Yellow Pages & Business Directory  40. Time & Billing  41. Class Room Management  42. Expense Report Database  43. Sales Contact Management Database  44. Inventory Management Database  45. Issue Database  46. Event Management Database  47. Service Call Management Database  48. Accounting Ledger Database  49. Asset Tracking Database  50. Cycle Factory Works Management  51. Sales Corporation Management  52. Business Directory  53. Education Directory  54. Dental Clinic Management  55. Fund Raising Management  56. Clinic/ Health Management  57. Cable Management System  58. Survey Creation and Analytics  59. Museum Management System  60. Multi-Level Marketing System  61. Learning Management System  62. Knowledge Management System  63. Missing Person Site  64. Disaster Management Site  65. Job Management Site  66. Financial Portfolio Management  67. Market Research Management  68. Order Management System  69. Point of Sale  70. Advertisement /Banner Management and Analytics  71. Export Management System  72. Invoice Management  73. Recruitment Management System  74. Articles / Blog / Wiki Web site  75. Online Planner  76. Mock Tests and Examination Management  77. Examination System  78. Practice Test Management.  79. Asset Management System  80. Travel Agency System.  81. Placement Management System.  82. Polls Management  83. Customer Management  84. Project Management System.  85. Network Marketing System  86. Yoga Health Care Management  87. Personal Finance Management System  88. Real Estate Management System  89. Stock Mutual Funds Management  90. Careers and Employment Management System  91. Music Albums Management System  92. Classified Ads Managements  93. Property Management System  94. Sales & Retail Management  95. Dating Site  96. Hotel Management System  97. Search Engine  98. Online News Paper Site  99. Image Gallery  100. Staffing and Human Capital Management  101. Development of a feature-rich, practical Online Survey Tool (OST)  102 Development of a Web/Email based Search Engine  103. Development of a web-based Recruitment Process System for the HR group for a company  104. Development of a Budget Approval System  105. File system simulation  106. Development of a Network Print Spooler  107. Development of a HTTP Caching Proxy Server  108.Development of a Lost Articles and Letters Reconciliation System  109.Student Project Allocation and Management with Online Testing System (SPM)  110.Development of a user friendly ,feature-rich, practical Online Testing System (OTS). | 111.Development of a feature-rich, practical Resource Management System (RMS)  112.Development of a feature rich, practical online Tickets reservation system for Cinema halls.  113.Development of a feature rich, practical Time table generation system for a college.  114.Development of a user friendly ,feature-rich, practical Appraisal Tracker  115.Development of Effort Tracker System  116.Development of a feature-rich, practical ?Web Enabled Estate Agent?(WEEA)  117.Development of a Web Based Mail Client  118.Development of a work flow based Complaint Management System (where the complaints are received through emails)  119.Development of an application for receiving orders for printing digital photographs  120.Development of a work flow based purchase request approval system  121.Development of a Defect Tracking System (DTS)  122.Development of a Product Master Maintenance system  123.Development of a Recipe Management System  124. Development of a feature-rich Employee Separation System (E-Separation System)  125. Development of a Miles Acquisition System (MAS)  126. Development of a Network packet sniffer  127. Development of a Web Based Meeting Scheduler  128. Development of an Employee Cubicle Management System  129. Development of a web based Stationery Management System  130. Development of an Online Course Portal for a campus  131. Development of an Online Auctioning Shop for a campus/organization  132. Solving system of linear equations using parallel processing  133. Design and development of Point Of Sale [ POS ]  134. Design and development of Speed Cash System [ SCS ]  135. Development of a feature-rich, practical online leave management system (LMS)  136. Development of a practical Online Help Desk (OHD) for the facilities in the campus  137. Development of an auto-summarization tool  138. Development of an agent-based information push mechanism  139. Development of a feature-rich, practical online on-request courses coordination system (ORS)  140. Development of an online Library Management System (LiMS)  141. Development of an online Sales and Inventory Management System (SIMS)  142. Development of a feature-rich, Employee Transfer Application  143. Development of a feature-rich, Resume Builder Application  144. Development of a safe and secure Internet banking system( Java based) OR Banking System in Visual Basic( Stand Alone)  145. Development of a feature-rich, practical online intranet knowledge mgmt system for the college (KMS).  146. Development of a feature-rich, practical online application for the Training and Placement Dept. of the college  147. Development of a Repository and Search Engine for Alumni of College (RASE)  148. Development of a split screen application for the data entry of the shipments.  149. Development of a Campaign Information System  150 Development of an e-Post Office System |