1. **1 ABOUT THE ORGANIZATION**

* + 1. **Introduction about EME Technologies**

EME Technologies a software company is managed by a team of dedicated, committed and highly qualified software & hardware professionalThe company is fortunate enough to have been associated with expert and experienced faculty in the fields of Software, Hardware and Web Technologies, Embedded System, PCB Designing, Networking, Autocad, automation and manymore.

 One of the areas of our specializations is Project study, analysis, development and its live implementationThe computer faculty in our computer institute has hands-on experience in software development and has a proven track record in training and guiding the students.

They take intensive care from the very first step of selecting the Project title and encouraging the student to venture a distinct project, involve themselves in the Project by putting their own effort, time and subject to write programs to execute the same

In case of any problems, the faculty in-charge willingly guides the student and helps them in successfully executing the ProjectFaculty give their support 24/7 toclient.  
  
The institute’s objectives is to empower the future computer Professionals by providing them decent work atmosphere, individual attention, creating confidence in them by encouraging them take-up the Project on their own, right from selection of topic until its implementation, facilitating its submission, under the supervision and guidance of experienced and expert faculty EME delivers an integrated portfolio of solutions and services reflecting a broad range of technology and business practices.

**1.1.2 EME-Overview**

The **“**EME Technologies**”**, is located in Mohali **i**n 7 Phase and in ChandigarhEMETECHNOLOGIES are Offshore Outsourcing Consultants with a leading edge technology focus on delivering the best and most cost-effective solutions to their clients in various areas of web development services and solutions.

The team at EME Technologies consists of over 30 highly skilled professionals associated with Information TechnologyEME Technologies delivers total solutions for software development and maintenance needs, serving companies from the smallest of start-ups to the largest of the Global 2000We specialize in offshore software development and web applications.

At EME Technologies a talented group of designers and interface engineers are masters at effectively conveying a consistent corporate message and brand while concentrating on ensuring a pleasant and useful user experienceThey help in effectively market the company by utilizing their skills in web strategy, creative interface design, corporate branding and logo design, online marketing strategy and copywriting.

**1.1.3 Vertical practices**

**1.1.3.1 Service Practices**

The various types of services that are offered by this company are as follows

* + - * 1. [Custom web applications](http://nztechnologies.com/custom_development.htm)
        2. [Web designing](http://nztechnologies.com/webdesigning.htm)
        3. [Complete e-business solutions](http://nztechnologies.com/ebusiness.htm)
        4. [Ecommerce business](http://nztechnologies.com/development.htm)
        5. [SEO Service](http://nztechnologies.com/seo.htm)
        6. [Web Data Mining](http://nztechnologies.com/data_mining.htm)
        7. Training and Consulting Services for the Development of Embedded Systems

**1.3.1.1.1 Custom application development**

It is meant for designing a software that has been designed and programmed for a specific function/need**Custom application development** is capable of producing practically any feature you may desire for your site.

EME TECHNOLOGIES provide total flexibility in terms of **Custom application development** - the process is essentially "Client Driven"It is important to remember that a well-designed database should provide the end product that is tailored to meet both your professional and practical business needs and therefore serve its intended purpose

**1.3.11.2 Web Designing**

EMETECHNOLOGIES has a team of experienced multi media and web designers for professional web site designing who work closely with our programming team to integrate the various components with a consistent look and feel that represents your corporate brand imageWe know the importance of a "first web impression" in web site design and our experienced team will advise you on how best to take advantage of changing trends and expectations

**1.3.11.3** [**Complete e-business solutions**](http://nztechnologies.com/ebusiness.htm)

EME Technologies ' **Complete e-business solutions** offer a powerful **combination of design and technology**This comprehensive set of online e-commerce technology is designed to help you make the most of your new or existing businessTheir complete e-business solutions will help you increase your sales and improve your bottom line.

**1.3.11.4 E-commerce business development**

Their shopping cart solutions are all you need to be a successful online storeMeet any growing e- commerce requirements with our shopping cart solutions for all sectors of the marketWith the advent of technology, business has increased manifold

If you are not updating your business in accordance to the changing scenario then a realization factor might follow showing your performance during the past timeThere cannot be any short cut to success for your retail businessAll you can do is to provide best offer to your customers by using Ecommerce solutions.

**1.3.11.5 Search Engine Optimization Firm**

It offers the facility of Optimizing and Positioning of your website in the Major Search EnginesThey initiate the **SEO Service** process by determining the Keyword/Phrase that best describe your Website/BusinessThen they build META Tags, for the few search engines that still use these

The placement and maintenance of your website is monitored through out the yearThis becomes even more important whenever the search engines changes their specificationsMaintenance is an extremely important aspect of a quality web site, not just for the benefit of search engines, but also for overall accuracy of your company's information

**1.3.11.6 Web Data mining**

It is a proven technology for advanced analysis that detects key patterns and trendsBut the time-consuming complexity of preparing Web data with the business context necessary for data mining has hampered its use in Web analysis—until now

EME Technologies creates, maintains, and runs Internet robots that retrieve data from the WebThe robots feed extractors that pick out useful information and can deliver it to you in a format for processing and analysis.

**1.3.11.7 Training and Consulting Services for the Development of Embedded Systems**

EME Technologies is pioneer in Establishing Embedded systems in ChandigarhOur expertise covers several microcontroller architectures and their development tool chainsIn addition we focus on topics such as time-to-market, quality improvement , complete PCB Designing and embedded applications using Atmel MCS51,Atmel AVR,Microchip PICcontroller, NXP ARM,Arduino and Raspberry pi Our training and consulting services include prototyping and customized software and hardware developments.Training, consulting or prototyping services include applications on technologies from Atmel , microchip , NXP , Maxim etc.

Embedded Systems Development involve Microcontroller families, these Microcontroller families require programming in Embedded C language.

EME is providing embedded system Development services and Training in Embedded systems, Robotics, VLSI,AUTOCAD since 2010.

1. **Python ─ Overview**

Python is a powerful modern computer programming language. It bears some similarities to Fortran, one of the earliest programming languages, but it is much more powerful than Fortran.

Python allows you to use variables without declaring them (i.e., it determines types implicitly), and it relies on indentation as a control structure. You are not forced to define classes in Python (unlike Java) but you are free to do so when convenient.

Python was developed by Guido van Rossum, and it is free software. Free as in “free beer,” in that you can obtain Python without spending any money. But Python is also free in other important ways, for example you are free to copy it as many times as you like, and free to study the source code, and make changes to it. There is a worldwide movement behind the idea of free software, initiated in 1983 by Richard Stallman.1

This document focuses on learning Python for the purpose of doing mathematical calculations.

We assume the reader has some knowledge of basic mathematics, but we try not to assume any previous exposure to computer programming, although some such exposure would certainly be helpful. Python is a good choice for mathematical calculations, since we can write code quickly, test it easily, and its syntax is similar to the way mathematical ideas are expressed in the mathematical literature. By learning Python you will also be learning a major tool used by many web developers.

* **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language:** Python is a great language for the beginner level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

1. **History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

* Python is derived from many other languages, including ABC, Modula-3, C, C++,

Algol-68, SmallTalk, and Unix shell and other scripting languages.

* Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
* Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.
* Python 1.0 was released in November 1994. In 2000, Python 2.0 was released.

Python 2.7.11 is the latest edition of Python 2.

* Meanwhile, Python 3.0 was released in 2008. Python 3 is not backward compatible with Python 2. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules so that "There should be one – and preferably only one -- obvious way to do it." Python 3.5.1 is the latest version of Python 3.

1. **Python Features**

**Python's features include:**

• Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntaxThis allows the student to pick up the language quickly.

• Easy-to-read: Python code is more clearly defined and visible to the eyes.

• Easy-to-maintain: Python's source code is fairly easy-to-maintain.

• A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh

• Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

• Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

• Extendable: You can add low-level modules to the Python interpreterThese modules enable programmers to add to or customize their tools to be more efficient.

• Databases: Python provides interfaces to all major commercial databases.

• GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

• Scalable: Python provides a better structure and support for large programs than shell scriptingApart from the above-mentioned features, Python has a big list of good features, few are listed below:.

• It supports functional and structured programming methods as well as OOP.

• It can be used as a scripting language or can be compiled to byte-code for building large applications.

• It provides very high-level dynamic data types and supports dynamic type checking.

• It supports automatic garbage collection.

• It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

1. **Python ─ Environment**

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

• Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)

• Win 9x/NT/2000 • Macintosh (Intel, PPC, 68K)

• OS/2

• DOS (multiple versions)

• PalmOS .

• Nokia mobile phones

• Windows CE

• Acorn/RISC OS

• BeOS

• Amiga

• VMS/OpenVMS

• QNX

• VxWorks

• Psion

• Python has also been ported to the Java and .NET virtual machines.

1. **Getting Python**

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python: http://www.python.org/. You can download Python documentation from www.python.org/doc/. The documentation is available in HTML, PDF, and PostScript formats.

1. **InstallingPython**

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python. If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation. Here is a quick overview of installing Python on various platforms: Unix and Linux Installation Here are the simple steps to install Python on Unix/Linux machine.

• Open a Web browser and go to <http://www.python.org/download/>.

• Follow the link to download zipped source code available for Unix/Linux.

• Download and extract files.

• Editing the Modules/Setup file if you want to customize some options.

• run ./configure script make install This installs Python at standard location /usr/local/bin and its libraries at /usr/local/lib/pythonXX where XX is the version of Python. Windows Installation Here are the steps to install Python on Windows machine.

• Open a Web browser and go to <http://www.python.org/download/>

• Follow the link for the Windows installer python-XYZ.msi file where XYZ is the version you need to install.

• To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.

• Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Macintosh Installation Recent Macs come with Python installed, but it may be several years out of date. See http://www.python.org/download/mac/ for instructions on getting the current version along with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available. Jack Jansen maintains it and you can have full access to the entire documentation at his website - http://www.cwi.nl/~jack/macpython.html. You can find complete installation details for Mac OS installation.

1. **Setting up PATH**

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables. The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs. The path variable is named as PATH in Unix or Path in Windows (Unix is casesensitive; Windows is not). In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

**Setting path at Unix/Linux**

To add the Python directory to the path for a particular session in Unix:

• In the csh shell: type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.

• In the bash shell (Linux): type export ATH="$PATH:/usr/local/bin/python" and press Enter.

• In the sh or ksh shell: type PATH="$PATH:/usr/local/bin/python" and press Enter. Note: /usr/local/bin/python is the path of the Python directory

**Setting path at Windows**

To add the Python directory to the path for a particular session in Windows: At the command prompt: type path %path%;C:\Python and press Enter. Note: C:\Python is the path of the Python directory

**Python Environment Variables**

Here are important environment variables, which can be recognized by Python:

|  |  |
| --- | --- |
| **Variable** | **Description** |
| PYTHONPATH | It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by the Python installer. |
| PYTHONSTARTUP | It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH. |
| PYTHONCASEOK | It is used in Windows to instruct Python to find the first caseinsensitive match in an import statement. Set this variable to any value to activate it. |
| PYTHONHOME | It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy. |

1. **Running Python**

**There are three different ways to start Python:**

1. **Interactive Interpreter**

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter python the command line. Start coding right away in the interactive interpreter..

$python # Unix/Linux

or

python% # Unix/Linux

or

C:>python # Windows/DOS

Here is the list of all the available command line options:

|  |  |
| --- | --- |
| **Option** | **Description** |
| -d | It provides debug output. |
| -O | It generates optimized bytecode (resulting in .pyo files). |
| -S | Do not run import site to look for Python paths on startup. |
| -v | verbose output (detailed trace on import statements). |
| -X | disable class-based built-in exceptions (just use strings); obsolete starting with version 1.6. |
| -c cmd | run Python script sent in as cmd string |
| File | run Python script from given file |

1. **Script from the Command-line**

A Python script can be executed at command line by invoking the interpreter on your application, as in the following:

$python script.py # Unix/Linuxor

python% script.py # Unix/Linuxor C:>python script.py #

Windows/DOS

**Note:** Be sure the file permission mode allows execution.

1. **Integrated Development Environment**

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

• Unix: IDLE is the very first Unix IDE for Python.

• Windows: PythonWin is the first Windows interface for Python and is an IDE with a GUI.

• Macintosh: The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files. If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine. Note: All the examples given in subsequent chapters are executed with Python 2.4.3 version available on CentOS flavor of Linux. We already have set up Python Programming environment online, so that you can execute all the available examples online at the same time when you are learning theory. Feel free to modify any example and execute it online.

1. **Python ─ Basic Syntax**

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

**First Python Program**

Let us execute programs in different modes of programming. Interactive Mode Programming: Invoking the interpreter without passing a script file as a parameter brings up the following prompt:

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>

Type the following text at the Python prompt and press the Enter:

|  |
| --- |
| >>> print "Hello, Python!"; |

If you are running new version of Python, then you need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result:

|  |
| --- |
| print "Hello, Python!"; |

Defining functions

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function. For a simple example,

the compound command

>>> def f(x):

... return x\*x

...

defines the squaring function f(x) = x2, a popular example used in elementary math courses. In the definition, the first line is the function header where the name, f, of the function is specified.

Subsequent lines give the body of the function, where the output value is calculated. Note that the final step is to return the answer; without it we would never see any results. Continuing the example, we can use the function to calculate the square of any given input:

>>> f(2)

4

>>> f (2.5)

6.25

The name of a function is purely arbitrary. We could have defined the same function as above,

but with the name square instead of f; then to use it we use the new function name instead of

the old:

>>> def square (x):

... return x\*x

...

>>> square (3)

9

>>> square (2.5)

6.25

Actually, a function name is not completely arbitrary, since we are not allowed to use a reserved word as a function name. Python’s reserved words are: and, def, del, for, is, raise, assert, elif, from, lambda, return, break, else, global, not, try, class, except, if, or, while, continue, exec, import, pass, yield.

By the way, Python also allows us to define functions using a format similar to the Lambda

Calculus in mathematical logic. For instance, the above function could alternatively be defined in

the following way:

>>> square = lambda x: x\*x

Here lambda x: x\*x is known as a lambda expression. Lambda expressions are useful when you need to define a function in just one line; they are also useful in situations where you need a function but don’t want to name it.

Usually function definitions will be stored in a module (file) for later use. These are indistinguish- able from Python’s Library modules from the user’s perspective. 7

2.5 Files

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library. For example, to store our squaring function example in a file, we can use any text editor3 to type the code into a file, such as

def square (x):

return x\*x

Notice that we omit the prompt symbols >>>, ... when typing the code into a file, but the indentation is still important. Let’s save this file under the name “SquaringFunction.py” and then open a terminal in order to run it:

doty@brauer:~% python

Python 2.5.2 ( r252 :60911 , Apr 21 2008 , 11:12:42)

[GCC 4.2.3 ( Ubuntu 4.2.3 -2ubuntu7)] on linux2

Type " help", " copyright", " credits" or " license"

for more information.

>>> from SquaringFunction import square

>>> square (1.5)

2.25

Notice that I had to import the function from the file before I could use it. Importing a command from a file works exactly the same as for library modules. (In fact, some people refer to Python files as “modules” because of this analogy.) Also notice that the file’s extension (.py) is omitted in the import command.

**Testing code**

As indicated above, code is usually developed in a file using an editor. To test the code, import it into a Python session and try to run it. Usually there is an error, so you go back to the file, make a correction, and test again. This process is repeated until you are satisfied that the code works.

The entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid. This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

**Scripts**

If you use Mac OS X or some other variant of Unix (such as Linux) then you may be interested in running Python commands as a script. Here’s an example. Use an editor to create a file name

SayHi containing the following lines

#! /usr/bin/ python

print " Hello World!"

print "- From your friendly Python program"

3Most developers rely on emacs for editing code. Other possible choices are Notepad for Windows, gedit for

Linux/Gnome, and TextEdit for OS X. IDLE comes with its own editor, by the way.

The first line tells Python that this is a script. After saving the file, make it executable by typing

chmod 755 SayHi in the terminal. To run the script, type ./SayHi in the terminal. Note that if you move the script someplace in your search path, then you can run it simply by typing SayHi.

Type echo $PATH to see what folders are in your search path, and type which python to see where your python program is — this should match the first line in your script.

As far as I know, it is impossible to run Python scripts in a similar way on a Windows machine.

**Interactive Python**

If you execute Python from the command line with no script (no arguments), Python gives you an interactive prompt. This is an excellent facility for learning Python and for trying small snippets of code. Many of the examples that follow were developed using the Python interactive prompt.

Start the Python interactive interpreter by typing python with no arguments at the command line. For example:

$ python

Python 2.6.1 (r261:67515, Jan 11 2009, 15:19:23)

[GCC 4.3.2] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> print 'hello'

hello

>>>

You may also want to consider using IDLE. IDLE is a graphical integrated development environment for Python; it contains a Python shell. It is likely that Idle was installed for you when you installed Python. You will find a script to start up IDLE in the Tools/scripts directory of your Python distribution. IDLE requires Tkinter.

In addition, there are tools that will give you a more powerful and fancy Python interactive interpreter. One example is IPython, which is available at <http://ipython.scipy.org/>.

***Lexical matters***

**1.2.1 Lines**

● Python does what you want it to do *most* of the time so that you only have to add extra characters *some* of the time.

● Statement separator is a semicolon, but is only needed when there is more than one statement on a line. And, writing more than one statement on the same line is considered bad form.

● Continuation lines A

Backslash as last character of the line makes the following line a continuation of the current line. But, note that an opening "context" (parenthesis, square bracket, or curly bracket) makes the backslash unnecessary.

**1.2.2 Comments**

Everything after "#" on a line is ignored. No block comments, but doc strings are a comment in quotes at the beginning of a module, class, method or function. Also, editors with support for Python often provide the ability to comment out selected blocks of code, usually with "##".

**1.2.3 Names and tokens**

● Allowed characters: az

AZ

09

underscore, and must begin with a letter or underscore.

● Names and identifiers are case sensitive.

● Identifiers can be of unlimited length.

● Special names, customizing, etc. Usually begin and end in double underscores.

● Special name classes Single and double underscores.

○ Single leading single underscore Suggests

a "private" method or variable name. Not imported by "from module import \*".

○ Single trailing underscore Use it to avoid conflicts with Python keywords.

○ Double leading underscores Used in a class definition to cause name mangling (weak hiding). But, not often used.

● Naming conventions Not rigid, but:

○ Modules and packages all lower case.

○ Globals and constants Upper case.

○ Classes Bumpy caps with initial upper.

○ Methods and functions All lower case with words separated by underscores.

○ Local variables Lower case (with underscore between words) or bumpy caps with initial lower or your choice.

○ Good advice Follow the conventions used in the code on which you are working.

● Names/variables in Python do not have a type. Values have types.

**1.2.4 Blocks and indentation**

Python represents block structure and nested block structure with indentation, not with begin and end brackets. The empty block Use the pass noop statement.

Benefits of the use of indentation to indicate structure:

* Reduces the need for a coding standard. Only need to specify that indentation is 4 spaces and no hard tabs.

● Reduces inconsistency. Code from different sources follow the same indentation style. It has to.

● Reduces work. Only need to get the indentation correct, not *both* indentation and brackets.

● Reduces clutter. Eliminates all the curly brackets.

● If it looks correct, it is correct. Indentation cannot fool the reader.

Editor considerations The standard is 4 spaces (no hard tabs) for each indentation level.

You will need a text editor that helps you respect that.

**Operators**

● See: http://docs.python.org/ref/operators.html. Python defines the following

operators:

+ \*

\*\* / // %

<< >> & | ^ ~

< > <= >= == != <>

The comparison operators <> and != are alternate spellings of the same operator.

!= is the preferred spelling; <> is obsolescent.

● Logical operators:

and or is not in

● There are also (1) the dot operator, (2) the subscript operator [], and the function/method call operator ().

● For information on the precedences of operators, see the table at

<http://docs.python.org/2/reference/expressions.html#operatorprecedence>, which is reproduced below.

● For information on what the different operators *do*, the section in the "Python Language Reference" titled "Special method names" may be of help:

<http://docs.python.org/2/reference/datamodel.html#specialmethodnames>

The following table summarizes the operator precedences in Python, from lowest precedence (least binding) to highest precedence (most binding). Operators on the same line have the same precedence. Unless the syntax is explicitly given, operators are binary. Operators on the same line group left to right (except for comparisons, including tests, which all have the same precedence and chain from left to right see section 5.9 and exponentiation, which groups from right to left):

Operator Description

======================== ==================

lambda Lambda expression

or Boolean OR

and Boolean AND

not x Boolean NOT

in, not in Membership tests

is, is not Identity tests

<, <=, >, >=, <>, !=, == Comparisons

| Bitwise OR

^ Bitwise XOR

& Bitwise AND

<<, >> Shifts

Addition

and subtraction

\*, /, % Multiplication, division,

remainder

+x, x

Positive, negative

~x Bitwise not

\*\* Exponentiation

x.attribute Attribute reference

x[index] Subscription

x[index:index] Slicing

f(arguments...) Function call

(expressions...) Binding or tuple display

[expressions...] List display

{key:datum...} Dictionary display

`expressions...` String conversion

● Note that most operators result in calls to methods with special names, for

example \_\_add\_\_, \_\_sub\_\_, \_\_mul\_\_, etc. See Special method names

http://docs.python.org/2/reference/datamodel.html#specialmethodnames

Later, we will see how these operators can be emulated in classes that you define yourself, through the use of these special names.

**Code evaluation**

Understanding the Python execution model How Python evaluates and executes your code.

Evaluating expressions.

Creating names/variables Binding

The following all create names (variables) and bind values (objects) to them: (1) assignment, (2) function definition, (3) class definition,

(4) function and method call, (5) importing a module, ...

First class objects Almost all objects in Python are first class. Definition: An object is first class if: (1) we can put it in a structured object; (2) we can pass it to a function; (3) we can return it from a function.

References Objects (or references to them) can be shared. What does this mean?

● The object(s) satisfy the identity test operator is.

● The built in function id() returns the same value.

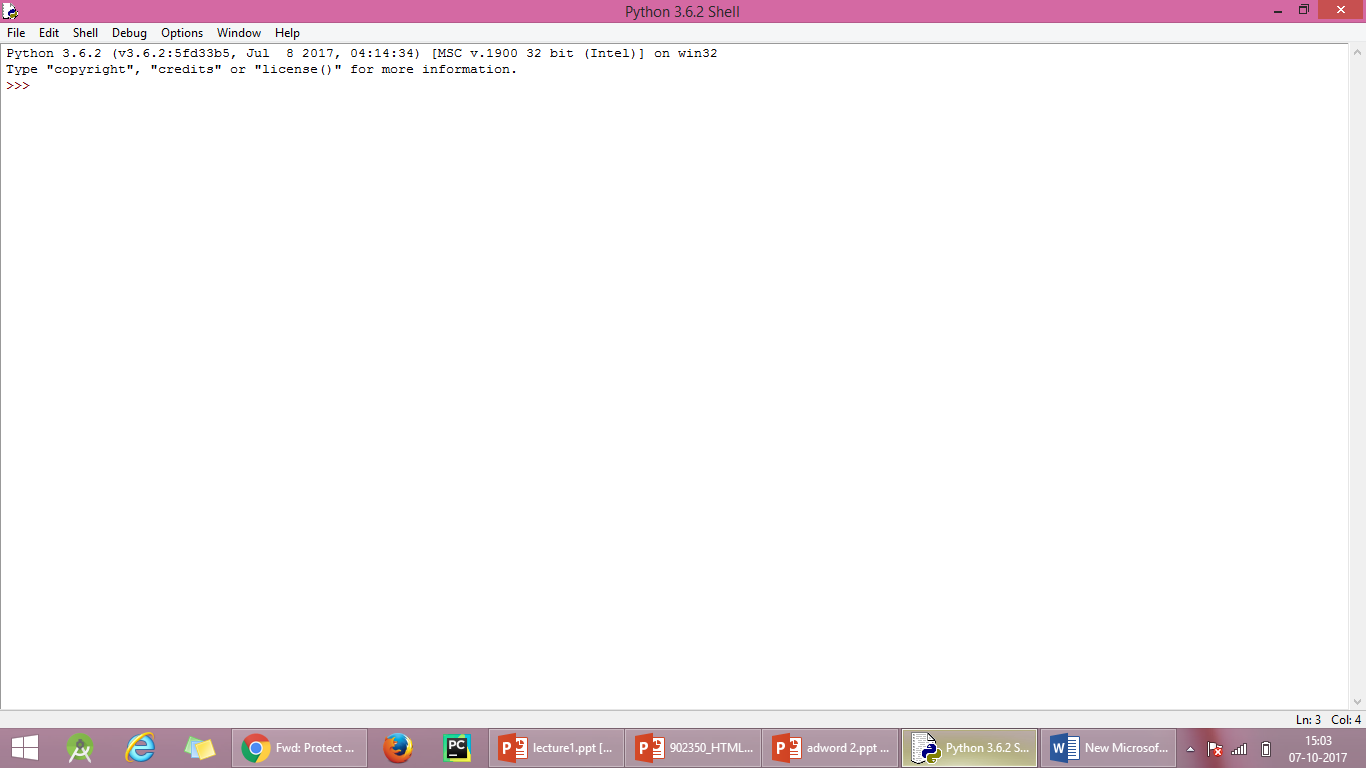
● The consequences for mutable objects are different from those for immutable objects.

● Changing (updating) a mutable object referenced through one variable or container also changes that object referenced through other variables or containers, because *it is the same object*.

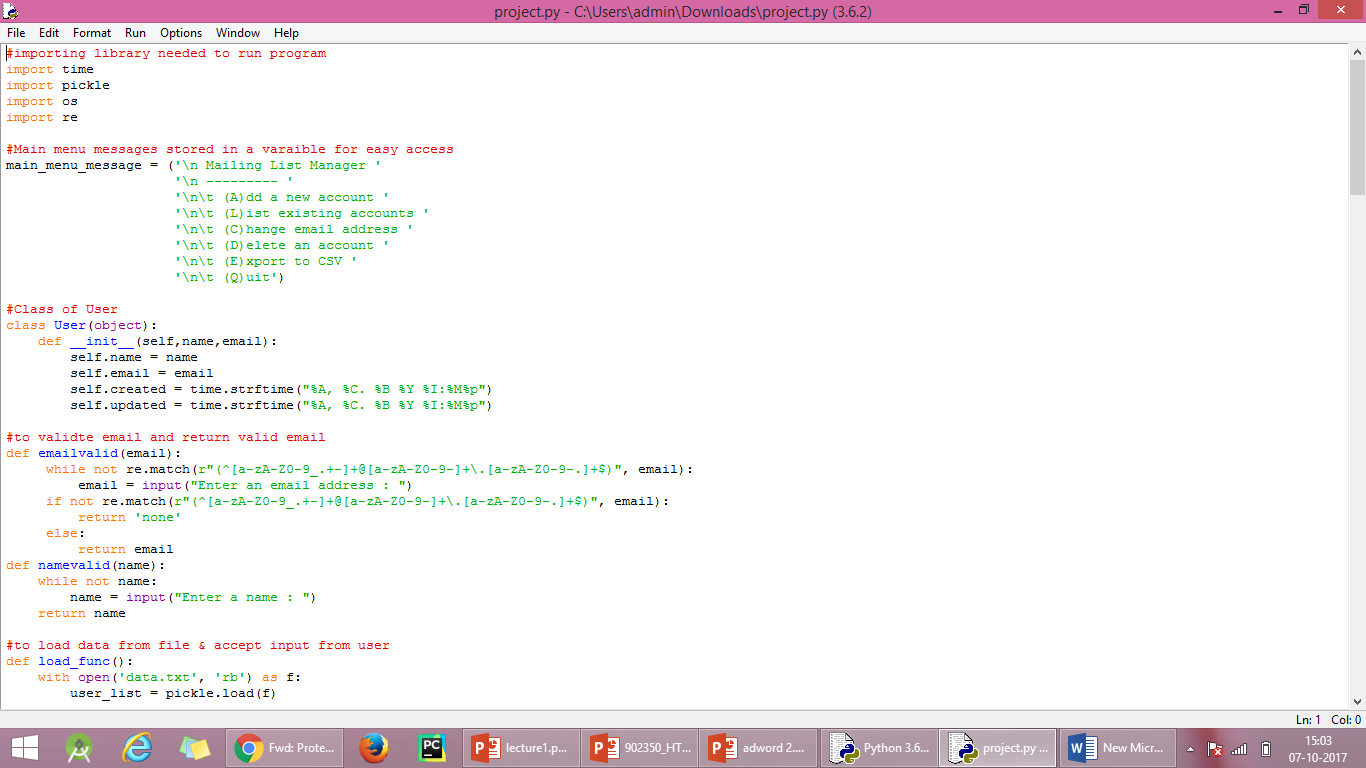
● del() The builtin function del() removes a reference, not (necessarily) the object itself.

**Screenshots**

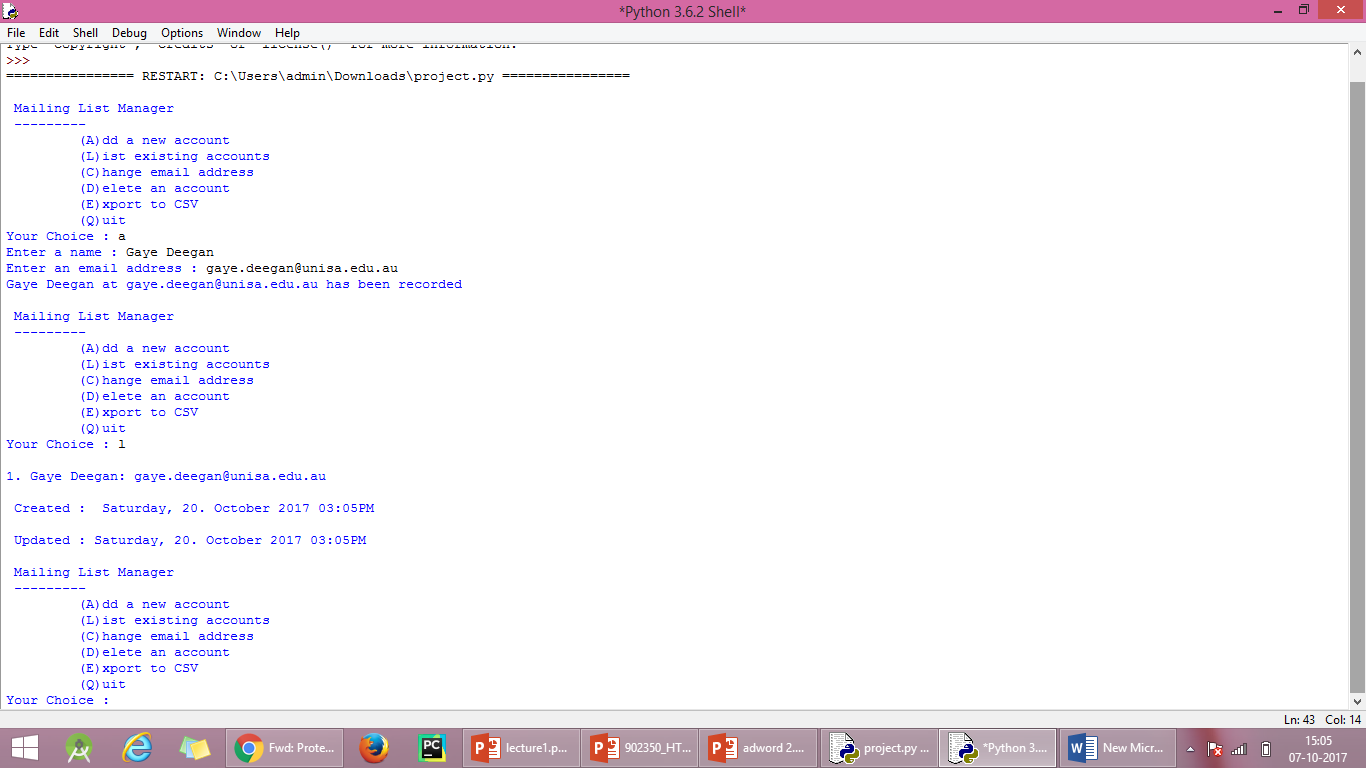
**1.**



**2.**



**3.**



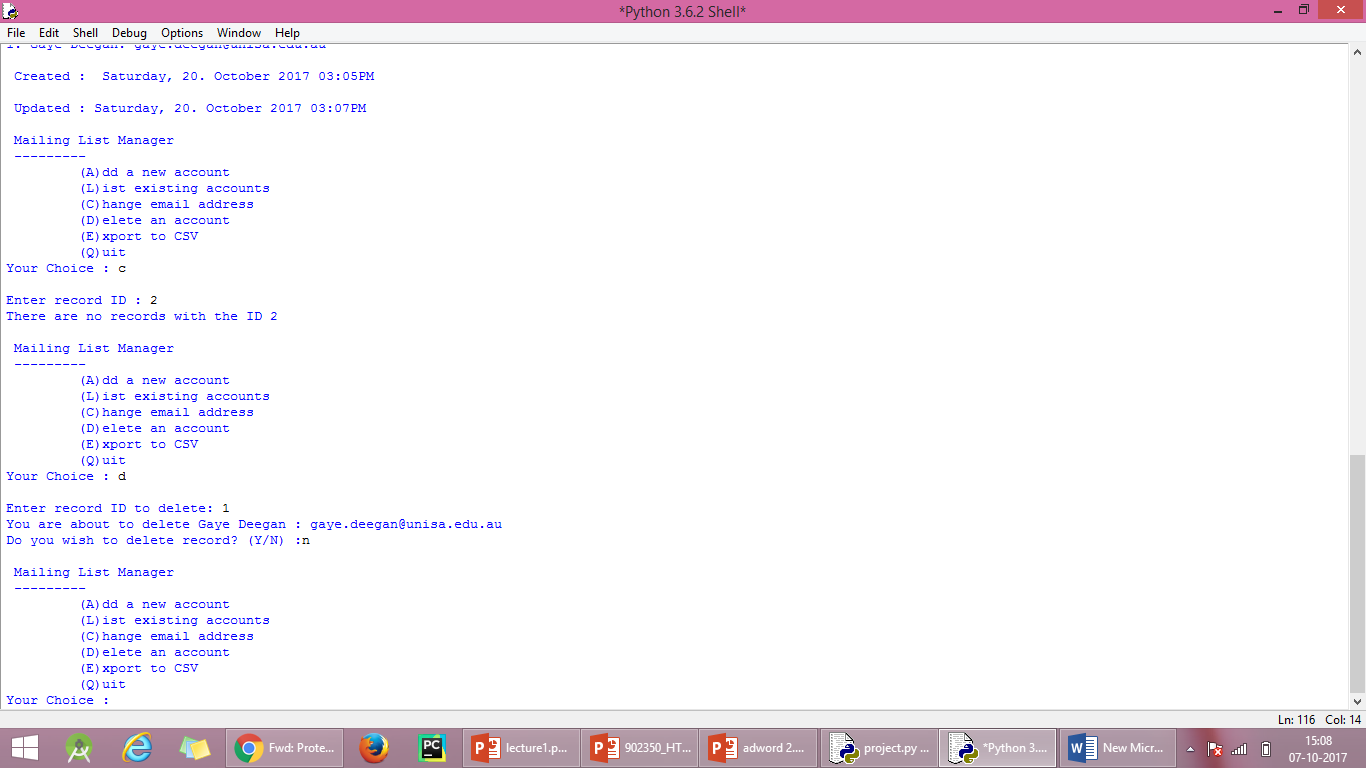
**4.**



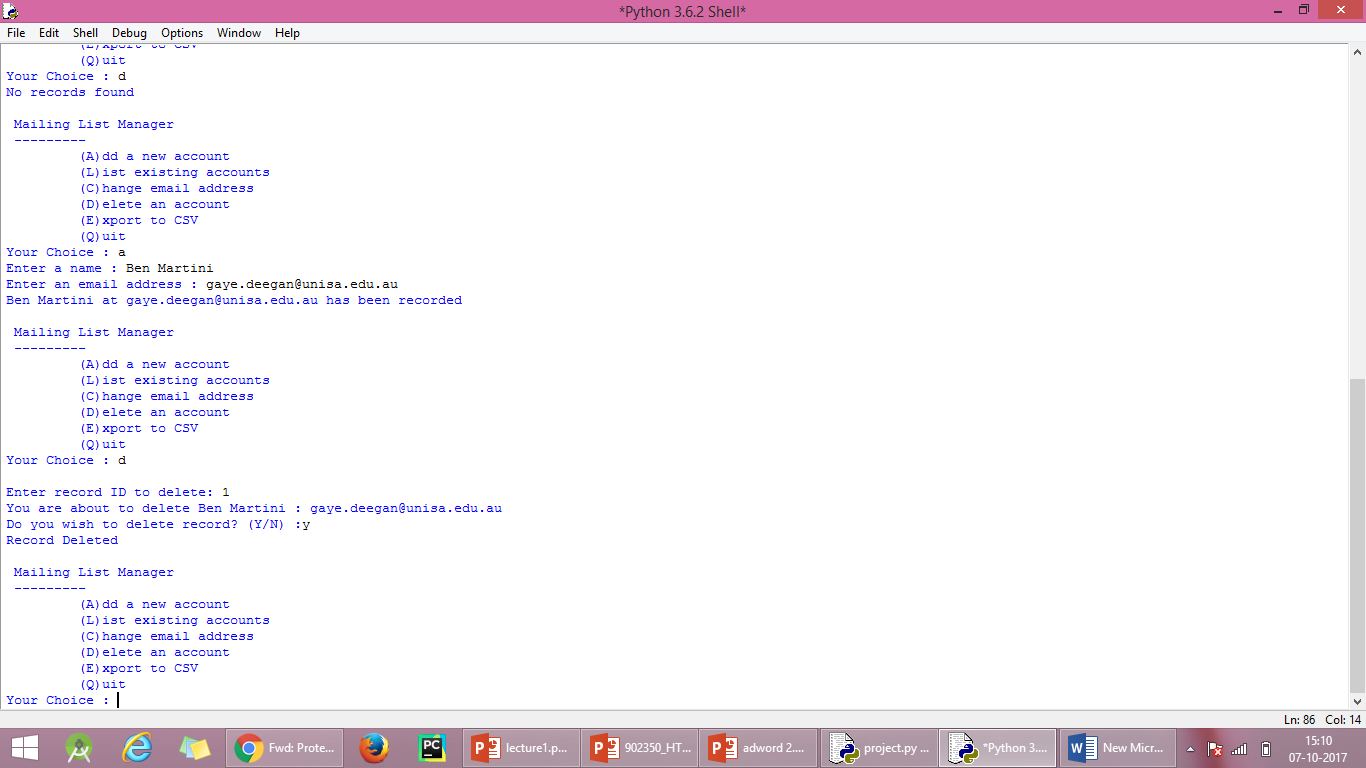
**5.**



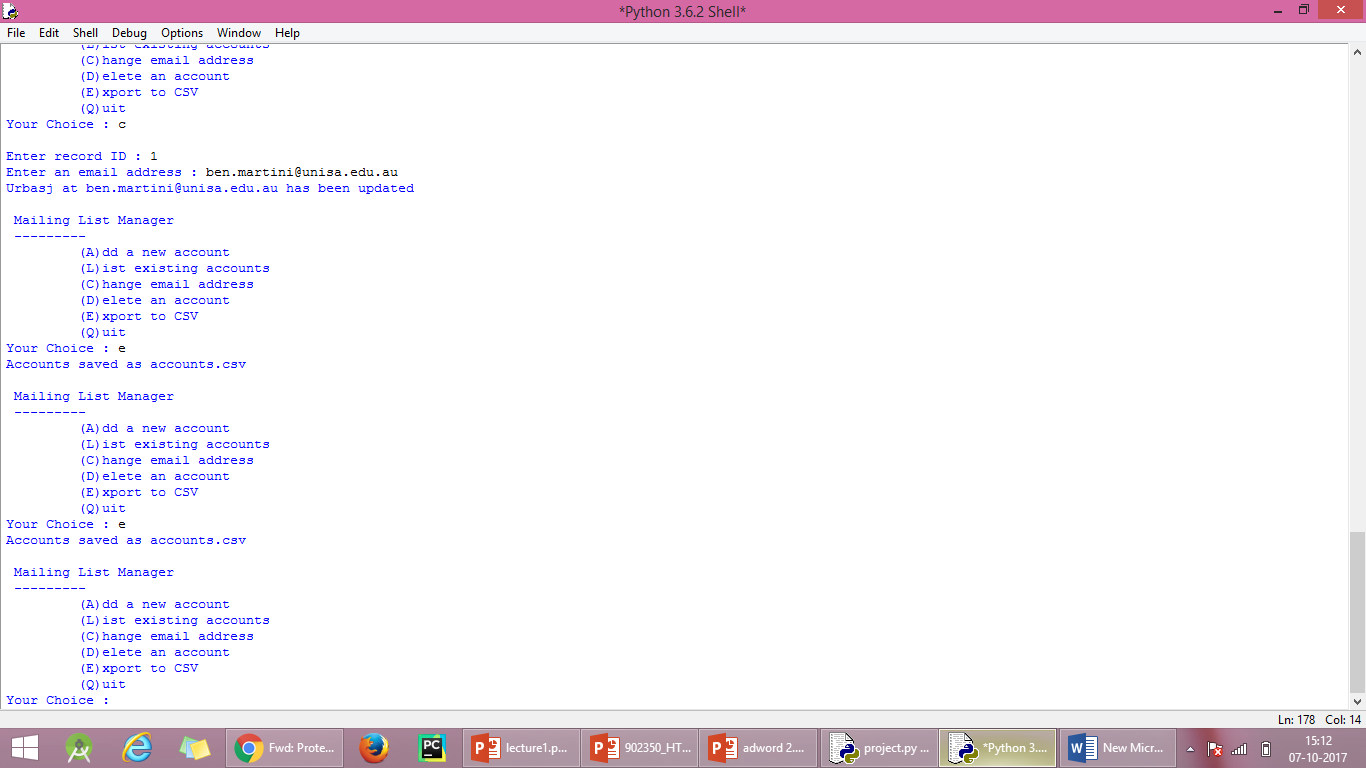
**6.**



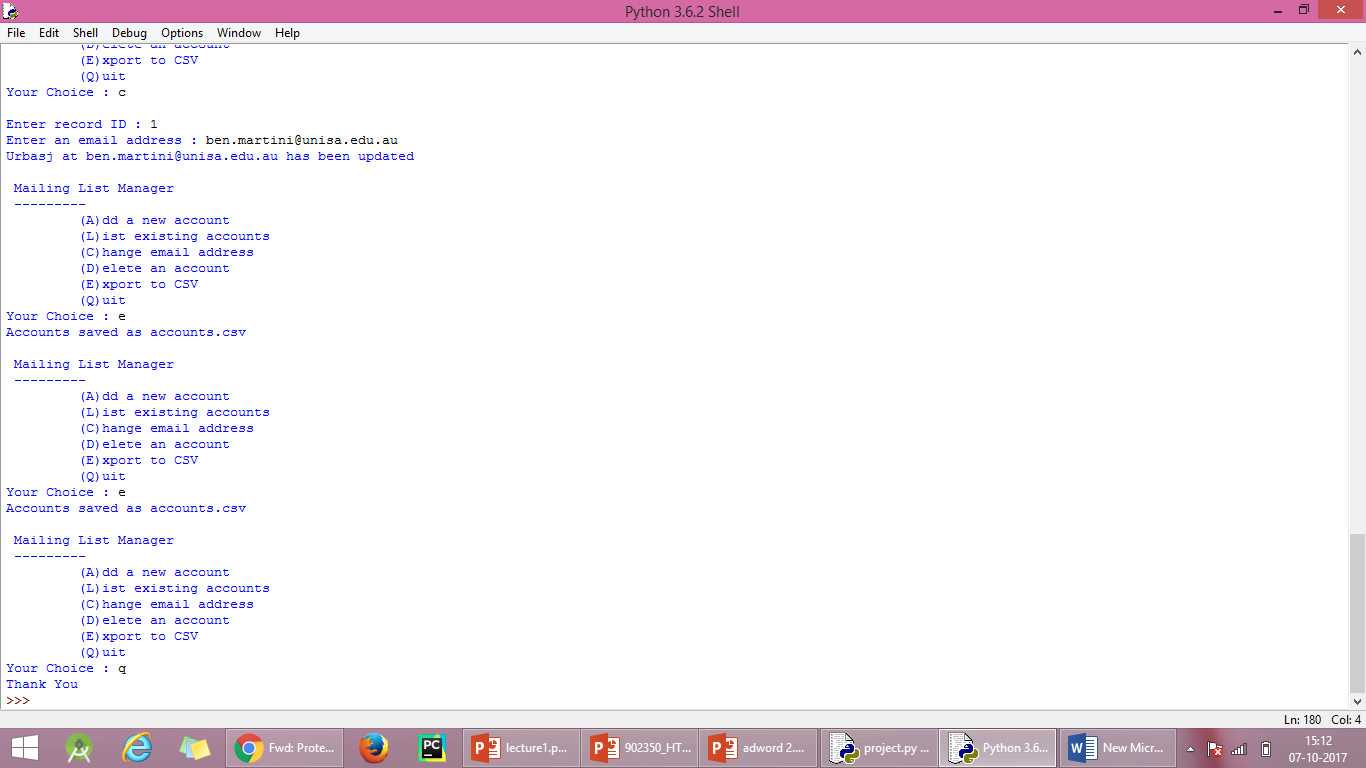
**7**.



**8.**



**9.**



**Programming**

#importing library needed to run program

import time

import pickle

import os

import re

#Main menu messages stored in a varaible for easy access

main\_menu\_message = ('\n Mailing List Manager '

'\n --------- '

'\n\t (A)dd a new account '

'\n\t (L)ist existing accounts '

'\n\t (C)hange email address '

'\n\t (D)elete an account '

'\n\t (E)xport to CSV '

'\n\t (Q)uit')

#Class of User

class User(object):

def \_\_init\_\_(self,name,email):

self.name = name

self.email = email

self.created = time.strftime("%A, %C. %B %Y %I:%M%p")

self.updated = time.strftime("%A, %C. %B %Y %I:%M%p")

#to validte email and return valid email

def emailvalid(email):

while not re.match(r"(^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)", email):

email = input("Enter an email address : ")

if not re.match(r"(^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)", email):

return 'none'

else:

return email

def namevalid(name):

while not name:

name = input("Enter a name : ")

return name

#to load data from file & accept input from user

def load\_func():

with open('data.txt', 'rb') as f:

user\_list = pickle.load(f)

name = namevalid(input('Enter a name : ').title())

email = emailvalid(input("Enter an email address : ").lower())

#creating User object instance

people = [User(name, email)]

#looping through object and storing data in dict

for p in people:

data = dict([

("name", p.name),

("email", p.email),

("created", p.created),

("updated", p.updated)

])

#Appending this dict to a list

user\_list.append(data.copy())

#Dumping the updated list to a file

with open('data.txt', 'wb') as f:

pickle.dump(user\_list, f)

print('{} at {} has been recorded'.format(name,email))

main\_menu()

#function add new account

def add\_new\_action():

#checking weather file exit of not

if(os.path.isfile('data.txt')):

load\_func()

#if no file exit , we are creating a new list and opening file and dump empty list to file

else:

user\_list = []

with open('data.txt', 'wb') as f:

pickle.dump(user\_list, f)

load\_func()

#function displaying all accounts

def list\_accounts():

with open('data.txt', 'rb') as f:

user\_list = pickle.load(f)

if not user\_list:

print("No records found")

main\_menu()

for index,x in enumerate(user\_list):

print('\n{}. {}: {} \n \n Created : {} \n \n Updated : {}'.format(index+1,

x['name'],

x['email'],

x['created'],

x['updated']))

main\_menu()

#function to changing email address

def change\_email():

with open('data.txt', 'rb') as f:

user\_list = pickle.load(f)

if not user\_list:

print("No records found")

main\_menu()

userInputID = input("\nEnter record ID : ")

try:

userInputID = int(userInputID)

except ValueError:

change\_email()

uID = userInputID - 1

#checking weather record avaliable or not

try:

data = user\_list[uID]

except IndexError:

print('There are no records with the ID {} '.format(userInputID))

main\_menu()

user\_list[uID]['email'] = emailvalid(input("Enter an email address : ").lower())

user\_list[uID]['updated'] = time.strftime("%A, %C. %B %Y %I:%M%p")

with open('data.txt', 'wb') as f:

pickle.dump(user\_list, f)

print('{} at {} has been updated'.format(user\_list[uID]['name'], user\_list[uID]['email']))

main\_menu()

#function deleting an account

def delete\_account():

with open('data.txt', 'rb') as f:

user\_list = pickle.load(f)

if not user\_list:

print("No records found")

main\_menu()

userInputID = input("\nEnter record ID to delete: ")

try:

userInputID = int(userInputID)

except ValueError:

delete\_account()

uID = userInputID - 1

try:

data = user\_list[uID]

except IndexError:

print('There are no records with the ID {} '.format(userInputID))

main\_menu()

print('You are about to delete {} : {}'.format(user\_list[uID]['name'],user\_list[uID]['email']))

choice = input('Do you wish to delete record? (Y/N) :')

if choice == 'Y' or choice == 'y':

del user\_list[uID]

elif choice == 'N' or choice =='n':

main\_menu()

else:

delete\_account()

with open('data.txt', 'wb') as f:

pickle.dump(user\_list, f)

print("Record Deleted")

main\_menu()

#fucntion exporting to csv file

def export\_csv():

with open('data.txt', 'rb') as f:

user\_list = pickle.load(f)

if not user\_list:

print("No records found")

main\_menu()

import csv

csvfile = "accounts.csv"

with open(csvfile, "w") as output:

writer = csv.writer(output, lineterminator='\n')

writer.writerow(('ID', 'Name', 'Email','Created','Updated'))

for index,x in enumerate(user\_list):

writer.writerow((index+1, x['name'], x['email'], x['created'], x['updated']))

print('Accounts saved as accounts.csv')

main\_menu()

#this runs first - The Menu

def main\_menu():

print(main\_menu\_message)

first\_choice = input('Your Choice : ')

#loading the previous data from text file to a list

# checking weather the input is string or not

# if its string then its given to mainmenu or

# if its number > record is printed on screen

try:

first\_choice = int(first\_choice)

with open('data.txt', 'rb') as f:

user\_list = pickle.load(f)

try:

first\_choice = first\_choice - 1

data = user\_list[first\_choice]

print('\n{}. {}: {} \n \n Created : {} \n \n Updated : {}'.format(first\_choice + 1,

user\_list[first\_choice]['name'],

user\_list[first\_choice]['email'],

user\_list[first\_choice]['created'],

user\_list[first\_choice]['updated']))

except IndexError:

print('There are no records with the ID {} '.format(first\_choice + 1))

main\_menu()

main\_menu()

except ValueError:

if first\_choice == 'A' or first\_choice == 'a':

add\_new\_action()

elif first\_choice == 'L' or first\_choice == 'l':

list\_accounts()

elif first\_choice == 'C' or first\_choice == 'c':

change\_email()

elif first\_choice == 'D' or first\_choice == 'd':

delete\_account()

elif first\_choice == 'E' or first\_choice == 'e':

export\_csv()

elif first\_choice == 'Q' or first\_choice == 'q':

print('Thank You')

exit()

else:

main\_menu()

#Program Starts here by calling initial screen

main\_menu()

**Mailing List Maintenance**

The goal here is to write some software that would assist a system administrator to maintain a list of email addresses. The assignment will get progressively more complex as it proceeds - the early stages will be reasonable straight forward and will be sufficient for a passing grade, and the more advanced stages are for those who wish to go further and target a higher overall result.

**General Requirements**

All assignments should be written in Python 3 and should run in IDLE. They can be written on any type of computer, but they should be able to be executed in the computer pools.

The highest marks will be for working code. However, in cases where there are problems, you will be assigned marks for how you approached the task. It is also expected that you will provide comments in the programs to explain how it works, and this will account for up to 10% of the overall grade.

**Stage 1: Data Entry**

Write a program which requests that the user enters a name and email address. Store that data in a List. After the user has entered data, request that the user enters Y or N to continue. If they enter N, print the list of names and email addresses to the screen. If they enter Y, repeat the first step.

Your program should produce the following result:

Mailing List Manager  
Enter a name: Gaye Deegan  
Enter an email address: gaye.deegan@unisa.edu.au  
Gaye Deegan at gaye.deegan@unisa.edu.au has been recorded.  
Do you wish to enter another account? (Y/N) Y  
Enter a name: Ben Martini  
Enter an email address: ben.martini@unisa.edu.au  
Ben Martini at ben.martini@unisa.edu.au has been recorded.  
Do you wish to enter another account? (Y/N) N  
1. Gaye Deegan: gaye.deegan@unisa.edu.au  
2. Ben Martini: ben.martini@unisa.edu.au  
Thankyou.

**Stage 2: Validation**

Building on stage 1, you now need to ensure that the program checks for valid entries in each case, and converts the text to the appropriate format.

You need to:

* Ensure that a name and email address was entered, If either the name or email is empty, do not record that item and go straight to asking if they wish to continue.
* Convert the name to title case, and the email to lower case.
* Ensure that the user enters either "Y", "y", "N" or "n" when asked if they wish to continue. Continue to ask until they enter a valid result.
* For extra marks, ensure that the email contains an "@" sign. This was not covered in the course, so will require extra research. If it does not contain "@", as the user to enter the email again.

Sample output:

Mailing List Manager  
Enter a name: Gaye Deegan  
Enter an email address: gaye.deegan@unisa.edu.au  
Gaye Deegan at gaye.deegan@unisa.edu.au has been recorded.  
Do you wish to enter another account? (Y/N) yes  
Do you wish to enter another account? (Y/N) y  
Enter a name:  
No record has been recorded.  
Do you wish to enter another account? (Y/N) Y  
Enter a name: BEN MARTINI  
Enter an email address: Ben.Martini.unisa.edu.au  
Enter an email address: Ben.Martini@unisa.edu.au  
Ben Martini at ben.martini@unisa.edu.au has been recorded.  
Do you wish to enter another account? (Y/N) N  
1. Gaye Deegan: gaye.deegan@unisa.edu.au  
2. Ben Martini: ben.martini@unisa.edu.au  
Thankyou.

**Stage 3: Objects and Persistance**

This involves two separate jobs:

* Create a class to store the details of the account rather than using a List on its own. Employ the Class in your code.
* Store the account details in a text file, and load then when next the program is launched.

Sample Output

Mailing List Manager  
Enter a name: Gaye Deegan  
Enter an email address: gaye.deegan@unisa.edu.au  
Gaye Deegan at gaye.deegan@unisa.edu.au has been recorded.  
Do you wish to enter another account? (Y/N) N  
1. Gaye Deegan: gaye.deegan@unisa.edu.au  
Thankyou.  
  
Mailing List Manager  
Enter a name: Ben Martini  
Enter an email address: ben.martini@unisa.edu.au  
Ben Martini at ben.martini@unisa.edu.au has been recorded.  
Do you wish to enter another account? (Y/N) N  
1. Gaye Deegan: gaye.deegan@unisa.edu.au  
2. Ben Martini: ben.martini@unisa.edu.au  
Thankyou.

**Stage 4: Options and CSV**

In this stage we need to add an optional "Export to CSV" option, along with the ability to choose to list existing entries and/or add a new one. The program should now ask the user to choose between 4 options: Add a new account, List existing accounts, Export to CSV, or Quit. If they do not choose one of those four options, they need to be asked again. If they choose Export, we need to output the data to accounts.csv. Otherwise we need to add, list or quite as required.

Now that things are getting more complex, you are expected to use functions to manage these tasks in your program.

Sample output:

Mailing List Manager

-----

(A)dd a new account

(L)ist existing accounts

(E)xport to CSV

(Q)uit

Your choice: A

Enter a name: Gaye Deegan

Enter an email address: gaye.deegan@unisa.edu.au

Gaye Deegan at gaye.deegan@unisa.edu.au has been recorded.

-----

(A)dd a new account

(L)ist existing accounts

(E)xport to CSV

(Q)uit

Your choice: L

1. Gaye Deegan: gaye.deegan@unisa.edu.au

-----

(A)dd a new account

(L)ist existing accounts

(E)xport to CSV

(Q)uit

Your choice: a

Enter a name: Ben Martini

Enter an email address: gaye.deegan@unisa.edu.au

Ben Martini at gaye.deegan@unisa.edu.au has been recorded.

-----

(A)dd a new account

(L)ist existing accounts

(E)xport to CSV

(Q)uit

Your choice: e

Accounts saved as accounts.csv.

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: Q

Thankyou.

Stage 5: Deleting and Updating

For the last stage we need to make this a bit more general purpose by introducing methods to delete and update records. As we're doing this, we will also need to record when events happen. Accordingly, you will need to add:

* A date/time created property in the account.
* A date/time updated property in the account.
* An ability to delete a record, where the user enters the ID (number) of the record and it is removed after confirming the record with the user. You will need to learn how to delete an item from a List and to update the save file.
* The ability to edit a record by changing the email address. As with delete, the user will need to specify the ID of the record to be edited. If no ID is specified, or if the ID is incorrect, it will not continue.

Sample output:

Mailing List Manager

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: A

Enter a name: Gaye Deegan

Enter an email address: gaye.deegan@unisa.edu.au

Gaye Deegan at gaye.deegan@unisa.edu.au has been recorded.

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: l

1. Gaye Deegan: gaye.deegan@unisa.edu.au

Created: Tuesday, 05. September 2017 07:25AM

Updated: Tuesday, 05. September 2017 07:25AM

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: d

Enter record ID: 1

You are about to delete Gaye Deegan: gaye.deegan@unisa.edu.au

Do you wish to delete this record (Y/N): N

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: d

Enter record ID: 1

You are about to delete Gaye Deegan: gaye.deegan@unisa.edu.au

Do you wish to delete this record (Y/N): Y

Record deleted.

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: l

No records to display

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: a

Enter a name: Ben Martini

Enter an email address: gaye.deegan@unisa.edu.au

Ben Martini at gaye.deegan@unisa.edu.au has been recorded.

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: l

1. Ben Martini: gaye.deegan@unisa.edu.au

Created: Tuesday, 05. September 2017 07:30AM

Updated: Tuesday, 05. September 2017 07:30AM

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: C

Enter record ID: 2

There are no records with the ID "2".

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: C

Enter record ID: 1

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: C

Enter record ID: 1

Enter an email address: ben.martini@unisa.edu.au

Ben Martini at ben.martini@unisa.edu.au has been updated.

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: l

1. Ben Martini: gaye.deegan@unisa.edu.au

Created: Tuesday, 05. September 2017 07:30AM

Updated: Tuesday, 05. September 2017 07:32AM

-----

(A)dd a new account

(L)ist existing accounts

(C)hange email address

(D)elete an account

(E)xport to CSV

(Q)uit

Your choice: Q

Thankyou.