During lectures we learn about the python. Problem Solving Methodology how to make a pseudocode correctly with data available and solution testing with a variety of data. How to describe a pseudocode in wording as shown in the works provided in my journal week 2 (The Eggs and pan activity / Customers activity). This helped me to understand the coding part called Pseudocode. We covered building blocks chapter where we learnt how to break programs in smaller parts. We also cover chapter of input and output that shows how to read and write files. We covered chapters like testing, GUI interface, managing codes, data management where we learnt about SQLITE and Pandas to give programs database functionality. For Assessment There were journals which was used like a revision for me to get an overview of the module and secondly programming assignment helped me to understand how to write codes and develop Interfaces and teaches how to handle Git hub repository and Collab Notebook used to write codes and text. During workshop we executed multiple coding to get practice and understand more what was told in lectures as for programming the more we practice the more we perfectionate the output on collab notebook.

**Appendix**

**For (Coding Screenshot 2) Theory**

In this pseodocode we can notice that a list of customers has been done

**Variable -** flavour with value ' vanilla'

**Print -** Displaying a message

**Static value -**'you have won an ice cream sundae' / 'Congratulations'

**Winner is a variable**

Assigning a value to a variable- << = >> (prompt)

wants\_ cherry= **input**(value 'yes' or 'no' ) ......  **Prompt -**Display on screen

Order = 'vanilla' + 'sundae' = < vanilla sundae >

If (want \_ cherry == <yes>) :                                                                                                    **== Assignment Operator**

**'    '    Compare operator**

*In this coding it explain the choice of certain customers and if they win different possibilities of ice cream and flavour ... if they would add topping or not .This is done by trying various possibilities of choice and flavours along the coding to get results accordingly .*

**For (coding Screenshot 4 ) Theory**

This is called the magnet coding explained by a cooking process where the steps must be in order and constant to attain the end of the code successfully.

as in the cooking of an egg example there is a flow first heat up the pan and the is "while" which shows another task being coded to create a constant flow and order in the coding to end with serve using different steps and trying a choice " if " in the coding as illustrated in my working below with full coding process.

* Attached Files:
* [Nikhil-journal 2 Descriptive writing .docx](https://lms.curtin.edu.au/courses/1/2022_1_ISYS2001_V2_L1_A1_INT_758993/blog/_290429_1/post/_470144_1/Nikhil-journal%202%20Descriptive%20writing%20.docx)(13.6kb)
* [Screenshot (4).png](https://lms.curtin.edu.au/courses/1/2022_1_ISYS2001_V2_L1_A1_INT_758993/blog/_290429_1/post/_470144_1/Screenshot%20%284%29.png)(54.4kb)
* [Screenshot (2).png](https://lms.curtin.edu.au/courses/1/2022_1_ISYS2001_V2_L1_A1_INT_758993/blog/_290429_1/post/_470144_1/Screenshot%20%282%29.png)(34.1kb)

Week 3

**Defining functions.**

Function is an important part in Python programming language. It is a piece of code written tocarry out a specific task. For example, print(), help(), and so on.

There is also anonymous functions which are called Lambda, that are not declared with the standard def keyword.

def calculate- is the def keyword and a name

price and discount- parameters or the inputs.

  ,'  ' the start block of the code.

'amount\_saved = price \* (discount/100) to return total\_price' - the block of code.

The space before starting wring each line of the code is the block of code intented.

**Header.**

def calculate\_price(price, discount): - is the **function header.**

1.Use def keyword.

2.List parameters (input).

3.Colon define a code block.

**Body.**

amount\_saved = price \* (discount/100)

sale\_price = price – amount\_saved

gst = sale\_price \* 0.1

total\_price = sale\_price + gst

return total\_price

All of the above is the **Function body.**

1.Set of statement.

2. Code block indented.

3.Optionally returns a value.

**Calling function.**

Calling function is a simple and single instruction that is given to the computer to perform only one task. For example, print("Hello, world!”).

print is used to call a function by using its name in code.

("Hello, world!”) - providing inputs to pass an argument to the function.

Another example can be:  name = input("What is your name?”)

name- is a save 'return' value in a variable.

input- some functions 'return' a value.\

In this case, a blank box is given to write any name that is required. A name has been used below as an example

**To summurise Calling functions:**

1.Call using name.

2.Provide zero or more arguments/inputs.

3.Save 'return' value to a variable.

**Modules.**

Modules are files with '.py' extension that contain Python code which then can be imported to another Python programming.

* Builtins.
* 3rd party functions (designed and tested by another person).
* Import

**Main Builtins.**

|  |  |
| --- | --- |
| input() | user input is allowed |
| int() | returns an integer number |
| len() | returns the length of an object |
| print() | prints to the output device |
| type() | returns the type of an object |

**Examples of Module:**import random

                                            random.choice(2,3,6)

import- keyword

random- module that needs to be used.

random.choice- choice() function from the random module.

**More Module examples:**

▪ image processing using pillow.

▪ openCV for computer visions.

▪ scikit-learn for machine learning.

▪ pandas for data analysis.

▪ keras for deep learning.

▪ numpy for numerical computations.

▪ matplotlib for plotting.

**Example of matplotlib:**

import matplotlib.pyplot as plt

plt.plot(1, 3, 5, 7, 9)

In the image above, matplotlib is the module, 'plt' is the alias and plot is the function. "as" is followed by an alias which is 'plt'.

Week 4

**Input output model.**

It is mainly used to analyse and do software engineering in systems to describe the structure of an information program being processed. It is also the most common structure to describe a process. There are 6 steps that are used to create an input output model which are mainly input, output, calculate,store ,decide, repeat

**The Core principles.**

1. Algorithms- expressed in pseudocode) ,sequence (put in order correctly), selection (if-then-else ,repetition (while)

2. Encapsulation- Manage complexity secondly data example (lists group values/variables) ,Code example (function group expressions )

Python.

Python provides various built-in functions that are already available for us to use at the Python prompt. For example, input(), print(), etc.

▪ Values (Literals)

▪ Data Types (everything has a datatype)

- str (string)

- int (integer)

- float (real number, decimal number)

- list []

▪ Expressions

- Mathematical operations à evaluates to value

- Relational Operators à Boolean Expressions

- Assignment. Store/save a value to a variable

- if-else (selection)

- Repetition (for)

▪ Functions

- builtins

- import (other packages to extend python)

Graphical user interface, text, application

Description automatically generatedInput- a=10

            print ('the value of a is', a)

output- the value of a is 10.

Development Environment.

▪ Python Notebook

-Code Cells

-Text Cells (Markdown)

-Interactively run cells

-Output in notebook

▪ Workflow

-Edit in notebook

-Frequently save to GitHub

Week 5

**Modularity.**

Recall Modules:

import random

random.choice([1,2,3,4])

'import' - import keyword

'random' - module that need to be used.

'random.choice'- call of the choice() function from the random module.

Recall Convenience:

import matplotlib.pyplot as plt

plt.plot([1,3,5,7,9], [1,9,25,49.81])

'import' - import keyword.

'lib' - module.

'as' - 'as' followed by an alias (plt).

'plt.plot' - alias followed by a dot then followed by a function.

**Specific import.**

from datetime import time

X= time(hour= 15)

'from'- from keyword.

'datetime' - module.

'import' - import keyword.

'time' - function.

**Module.**

It is a file containing Python codes which are mostly functions, variables and classes. It also organises the codes logically.

**Creating a Module.**

To create a Module, the first step to be done is to define the function or functions. Then, it needs to be saved to files and then imported.

A notebook is created and then it is imported by %run and Margo\_loader.

**Create the module (file: hello.py)**

Define a function:

A screenshot of a computer

Description automatically generated

define a variable

**name = steven**

A screenshot of a computer

Description automatically generated

**Package:**  ▪ Collection of Python files

                  ▪ Directory structure

                  ▪ \_\_init\_\_.py

                  ▪ Bundled together uploaded to PyPi

**Modules and Packages.**

Module is a file that contains the Python codes for a specific user code. A Package also modifies the interpreted user code so it can easily functioned.

▪ One job

▪ Hide internal workings

▪ Application Programmers Interface (API)

▪ Also called library

▪ Python Path

**Modular design.**

▪ it is used to manage complexity

▪ simplifies big problems into small ones

▪ small problems become functions

▪ group related functions into a module

▪ group related modules into a package

Week 6

**Testing.**

Problem Solving Methodology:

▪ Problem must be stated clearly

▪ The input and output must be described

▪ Work a simple example by hand

▪ Develop and algorithm (and convert to Python)

▪ Testing solution with a variety of data

Good habits to put in practice:

▪ Comments

▪ Docstrings

▪ Use or create functions

▪ Use or create modules

▪ Make small frequent commits

**Software Bug.**

It is a problem that causes the program to provide an invalid output or a crash. It misdirects the computer to a place in the program where the instruction does not exists and might cause the computer to stop at that point.

Types of Errors:

- Syntax error.

- Run- time error.

- Logic error.

EXAMPLE OF ERROR TESTING

Graphical user interface, text, application, email

Description automatically generated

**Debugging.**

It is where codes are used step by step in a debugging tool for example, Visual Studio, to find and see where a programming mistake was made.

▪ Step/Trace through code

- print()

- logging()

▪ Inspect Objects

- type()

- inspect module

▪ Python debugger – pdb

- breakpoint()

- traceback and other methods not enough

**Strategies for Debugging.**

▪ Apply Trail and Error

▪ Compare to similar code

▪ Copy and paste for working code

▪ Ask for help (peer, forum, google)

▪ Use IDE

- Syntax Highlighting.

- Auto completion.

- Linting.

**Program Testing.**

It is the process of executing a program to find errors. It is more reliable, reproducible and shareable.

Types of tests:

▪ Unit – test an individual isolated component.

▪ Integration – test multiple units work together.

▪ End-to-End – act as user, test entire stack.

▪ Acceptance Test – verify user store works as expected.

**Test Driven Development.**

It is a software practice to concentrates on creating a unit test case before developing the actual code.

**Approach Notebooks.**

▪ Write python script to test notebook

- Package: testbook

▪ Write test and code in one notebook

- Packages: assert(), doctest, unittest, nose2 and so on.

▪ ‘Testing Notebook’ and import notebook(s)

- Packages: assert(), doctest, unittest, nose2 and so on.

**Testing everything.**

▪ Build up a suite of Tests

▪ Run all test with one cell/notebook/script

▪ Regression testing

▪ Test Table

-  Sane test, then edge cases

Week 7

**Enterprise Software Today.**

It is cloud based and is complex independent of operating system (browser based).

Feature of Software Ecosystem:

▪ Reliability

▪ Accessibility

▪ Reusability

▪ Interpretability

▪ Flexibility

▪ Agility

Ecosystem Considerations:

1. How do you get the data?

2. Where does the code run?

3. How is your code organised?

4. Where do your outputs go?

Application;

▪ Development.

▪ Production.

▪ Interface - Widgets + Code.

▪ Deployment.

**Visual Environments.**

▪ Creating a virtual environment.

▪ Activating a virtual environment.

▪ Installing the required packages.

▪ Deactivating environment.

**Development.**

▪ Local Environment

- Jupyter Lab, VSCode.

▪ Cloud Environment

- AWS, Google Compute, Asure.

▪ Notebook Specific

- Paperspace, Colab.

**Deployment.**

▪ Notebooks as Packages

- NBDev

▪ Notebooks as (Web) Apps

- Ipywidgets, Voila, Appmode, Anvill

▪ Deploy/Share

- nbviewer, GitHub, Binder, Docker

- nbconvert to Script to Package/Exe

There is also binder and appmode in this module which is explained by the picture below :

  Above is shown an example of how url are typed then build logs appears when we click on launch it.

**APPMODE**

anyone can view any notebook in appmode by clicking appmode in toolbar button or simply change the url and when it is opened in app mode all codes cell are automatically executed.A notebook can also be opened multiplr times in appmode without interface.

for appmode there are 3 configuration options

- appmode trusted path

-appmode show edit button

-appmode show other buttons.