**LEARNING COMPUTER PROGRAMMING WITH PYTHON.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Arithmetic operations in Python.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Purpose:**

* Learn and interact with python standard math library,
* Solve a given arithmetic equation.
* Investigate python’s ability to work with arithmetic operations.

**Theory**

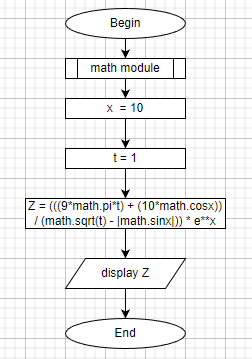
1. **Functions in python math library**

|  |  |
| --- | --- |
| function | use |
| math.sin(x) | Returns the sine of x radians. |
| math.log(x) | Returns the natural logarithm of *x* |
| math.pi | Returns the value of *pi* |
| math.floor(x) | Returns greatest integer value which is less or equal to *x* |
| math.ceil(x) | Returns least integer value which is greater or equal to *x* |
| math.e | Returns the value of *e* |
| math.log10(x) | Returns the logarithm of *x* to base 10 |
| math.cos(x) | Returns the cosine of *x* radians |
| math.atan(x) | Returns the inverse of the tangent of *x* radians |
| math.sinh(x) | Returns the hyperbolic sine of *x* |

1. **Solve for Z,**

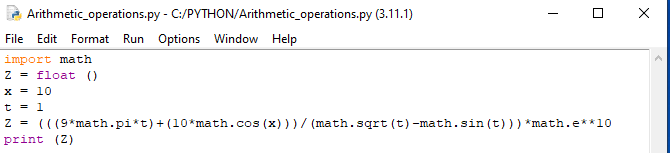
Given: x = 10 ; t = 1

Solution:

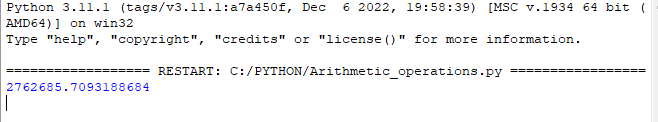


Z = 2762685.7093188684

1. **Code:**



1. **Test data:**



**Conclusion.**

Having interacted with the Python math library, it is prudent to conclude that the library extensively covers arithmetic operations for real numbers. It is also important to know the units of measurement used for effective application of the functions contained therein.

Daniel Kamau Njoroge.

Naivasha, 2023.

**Conversion of decimal number to binary notation.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Purpose:**

* Learn data types, variables and syntax as used in Python programming
* Write a code that converts decimal number to binary notation
* Learn operations that can be executed on various data types.

**Theory.**

As a language, Python uses four major data type, namely:

* + - Numbers :

includes all number type data, ( integers, float numbers and complex numbers)

* Strings:

includes all characters that are not numbers, (alphabets and symbols)

* Lists:

contains various data types and variables with similarities stored together in one variable.

* Boolean data:

when processed, there are only two possible outcomes. For example, (true or false, yes or no). Important in decision making processes.

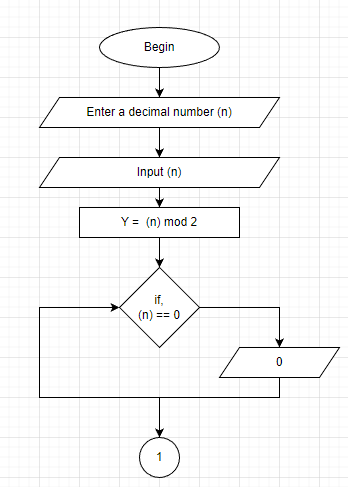
Variables are name tags given to containers that store the data. A variable is assigned to a data type using the sign (=). A variable points to an item in the memory.

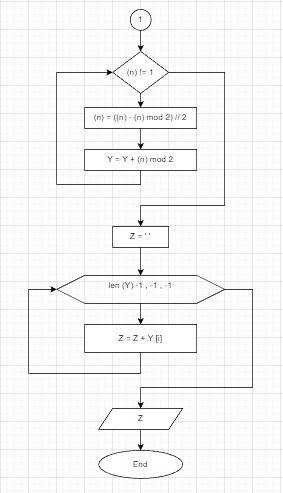
Numbers, as a data type have a wide range of operations, most of them being arithmetic operations.

**Task:**

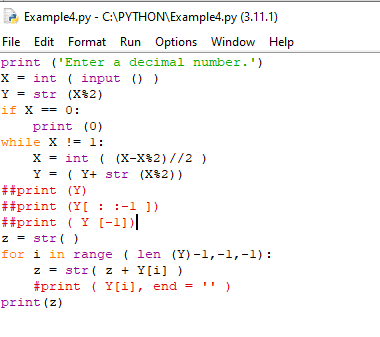
Write a program that converts numbers in decimal notation to binary notation.

**Solution:**

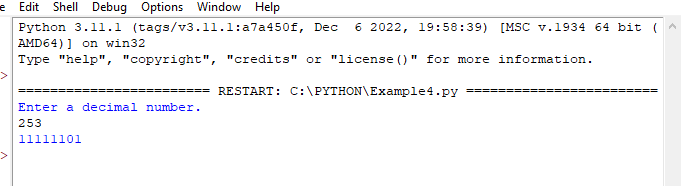




**Code:**

****

**Test data:**

****

**Conclusion:**

After completing the task, in-depth understanding of numbers as a data type has been achieved. Operations which can be carried out on numbers have been widely covered.

The loop function has also been integrated and learnt as well. The function is very useful when a process needs to be repeated in a program.

Daniel Kamau Njoroge.

Naivasha, 2023.

**Strings and string operations.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Purpose:**

* Define and identify strings and their major properties
* Perform various operations on strings using Python
* Write a code to solve an assigned task.

**Theory.**

Strings as a data type deal with text. A string is created by enclosing the text in quote marks. Any character enclosed in single or double quotation marks forms a string.

**String properties:**

* An empty string has nothing written in it.
* Characters in a string are immutable.
* Length of a string is the number of characters contained in the string

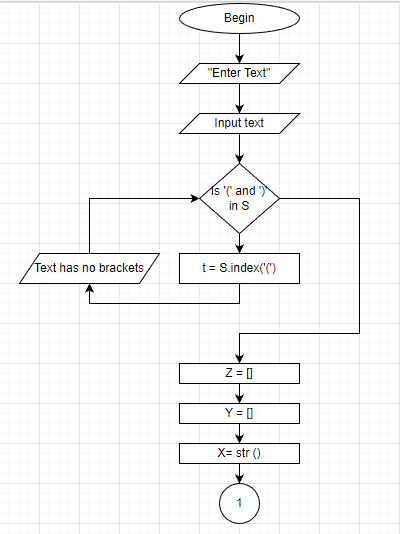
**Operations on strings:**

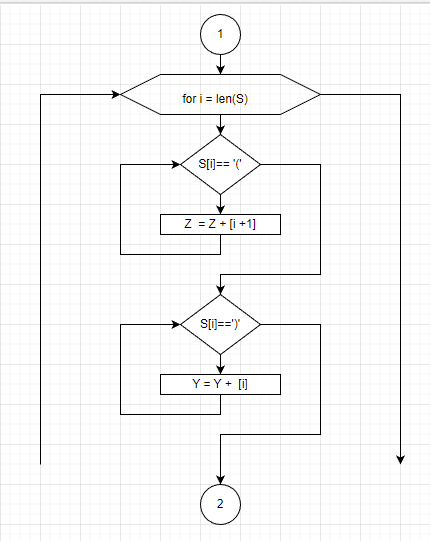
* Concatenation: uses the operator (+) to join one string to another string.
* Repetition: uses the operator (\*) to repeat a string a certain number of times.
* Indexing: helps to access individual characters in a string. Indexing starts from 0 and runs to *( len – 1)* of the string. Negative indices count backwards.
* String methods: are functions that return information about the string or return a new string modified from the original. Example, *lower (), upper ()* converts all alphabets in a string to the defined case.
* Escape characters: uses (\) to get characters which have other uses in the program into a string; examples + \* ‘
* Changing individual characters in a string: characters in a string are immutable. To change a character in a string to another character, one has to create a new string.

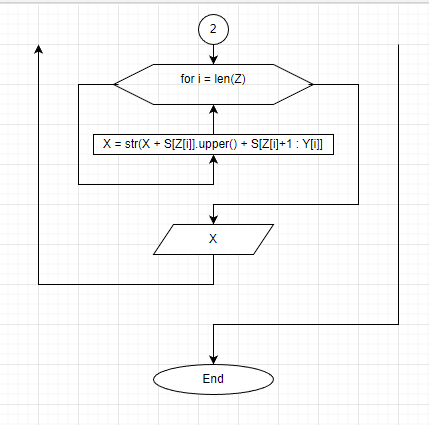
**Task:**

Given a few paragraphs of text containing several pairs of brackets, print on the screen, all characters that appear between the brackets, make the first letter in each pair of brackets to be capital.

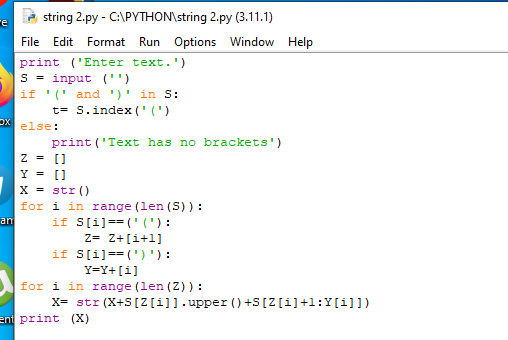
**Solution:**

****

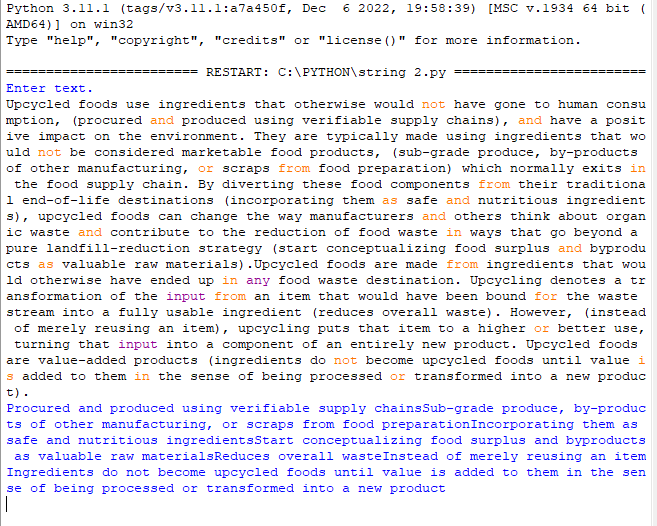
****

****

**Code:**

****

**Test data:**

****

**Conclusion:**

Python, being an interpreted language is more of typed than compiled. As such strings are an integral part in python programming. There are very many operations that can be done on strings when coding so as to solve real-life problems, or to suit the preferences of the user.

Daniel Kamau Njoroge.

Naivasha, 2023.

**Lists and List operations.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Purpose:**

* To learn and interact with lists as a data type in python.
* Understand properties of lists and operations that can be done on the lists when programming.
* Use the acquired skills to solve given task(s).

**Theory**

Python has a number of compound data types, used to group together other values. The most versatile is the list, which can be written as a list of comma-separated values (items) between square brackets. Lists might contain items of different types, but usually the items all have the same type. Other arrays include *tuples, sets* and *dictionaries.*

Properties of lists:

* Elements in a list are mutable
* Lists are ordered and allow indexing
* Lists allow repetition.

There are two types of lists:

1. One-dimensional lists: they are lists with only one row
2. Multi-dimensional lists: they are lists with multiple rows and columns.

Operations on lists:

All operations that can be carried out on strings such as concatenation, slicing, indexing, etc. can also be done on lists.

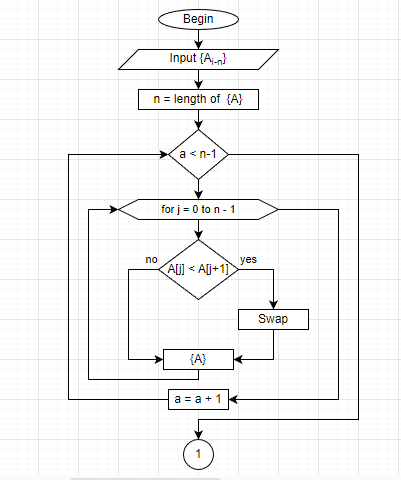
Lists also allow for:

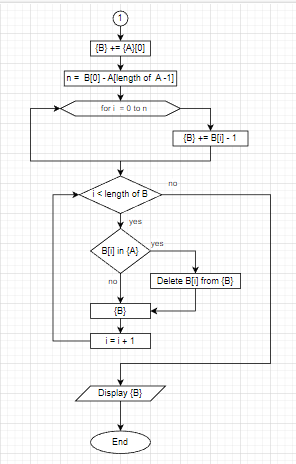
* changing of individual elements of the list
* mathematical operations on elements of the list if they are numbers.
* Clear() deletes an entire list.

**Task 1 (one-dimensional list):**

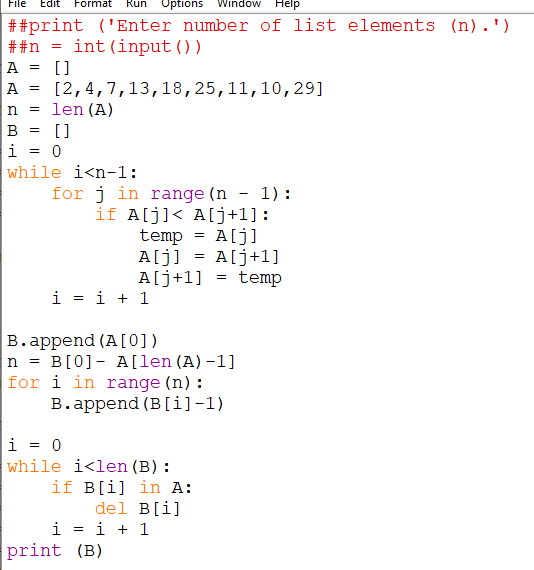
Given a sequence of whole numbers, (a1, a2,………………., an), let amax be the maximum among the numbers and amin be the minimum. Output in descending order all the integers in the interval between amax and amin that are not membersof the given sequence.

**Solution:**

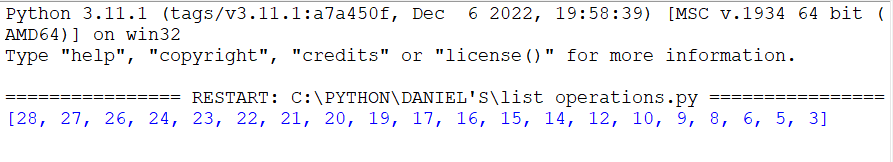
****

****

**Code:**

****

**Test data:**

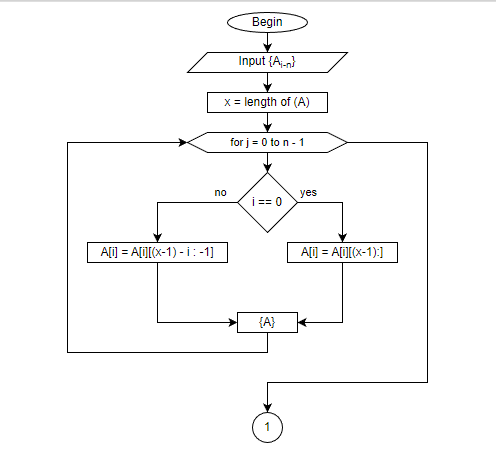
****

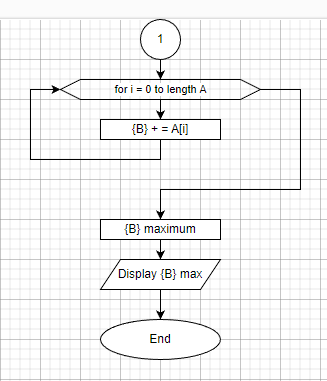
**Task 2 (two-dimensional list):**

Find the maximum element in a rectangular matrix in the shaded area.

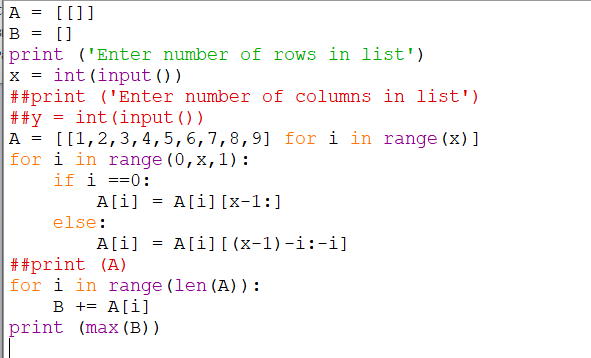


**Solution:**

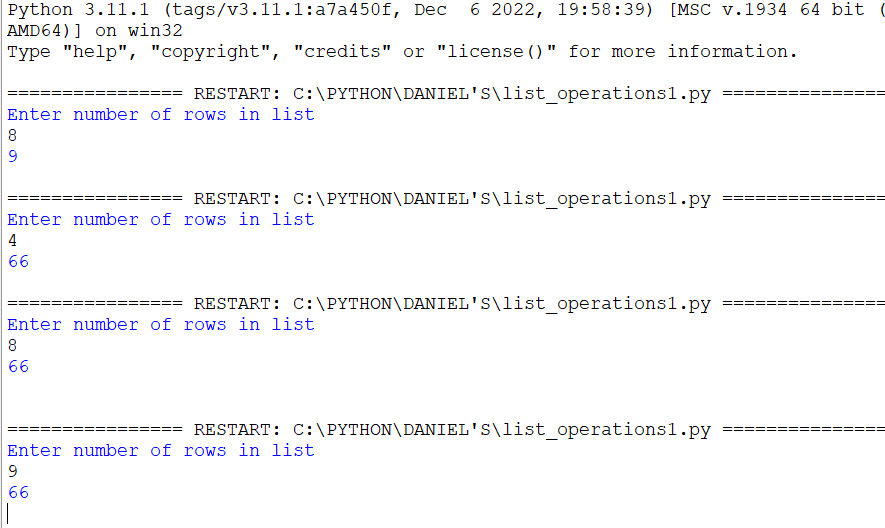
****

****

**Code:**

****

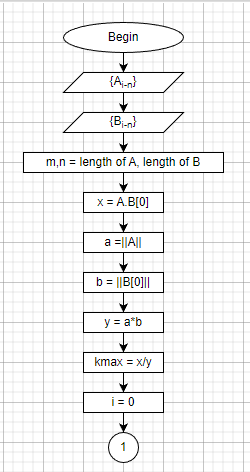
**Test data:**

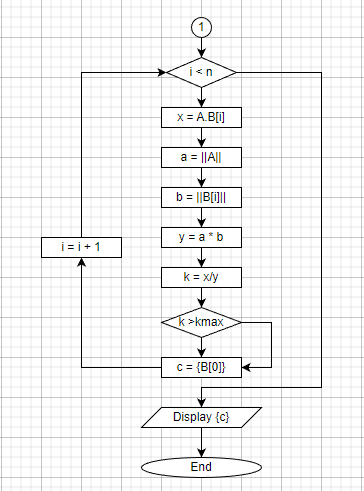


**Task 3 (multi-dimensional list):**

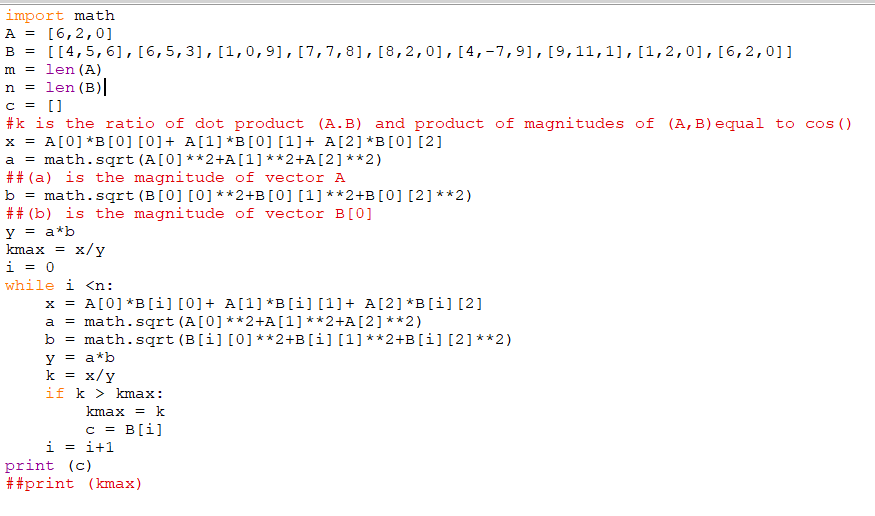
Every row of a matrix represents coordinates of a vector in space. Write a program that determines which of these vectors forms the minimum angle with a given vector (A = [ax , ay , az].

**Solution:**

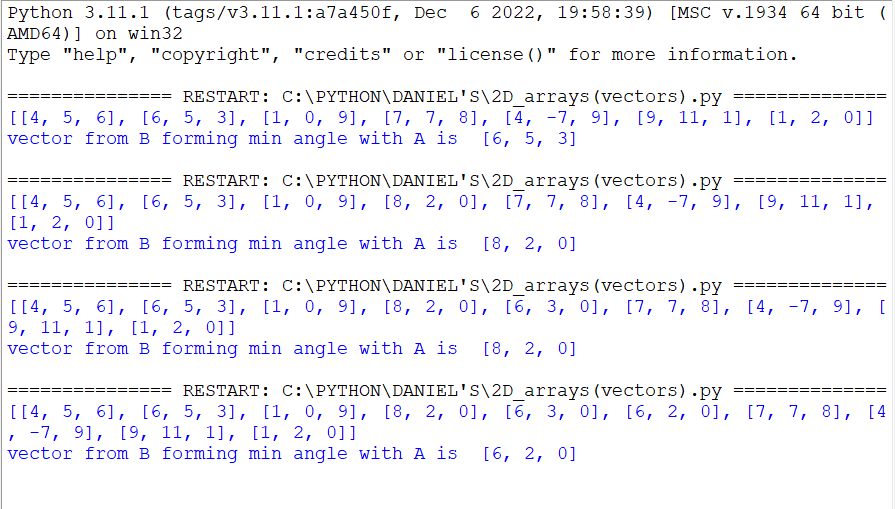
****

****

**Code:**

****

**Test data:**

****

**Arrays Continued: Sets, Tuples and Dictionaries.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Purpose:**

* To learn and apply other python arrays (sets, tuples and dictionaries)
* To learn characteristics of different arrays and the most suitable times to use them.
* To solve given tasks on python arrays.

**Theory.**

Arrays are a collection of data with specific characteristics. Other than lists, other arrays include: sets, tuples and dictionaries.

**Sets:**

Sets are used to store multiple items in a single variable. A set is a collection which is unordered, unchangeable, and unindexed. Duplicates are not allowed

To create a set, curly brackets {} are used. Once created ,set items cannot be changed. However, the items can be removed or new items added into the set.

**Tuples:**

Tuples are used to store multiple items in a single variable. A tuple is a collection which is *ordered* and *unchangeable*. It allows *indexing*.

Tuples are written with round brackets (). Tuples allow duplicates.

**Dictionaries:**

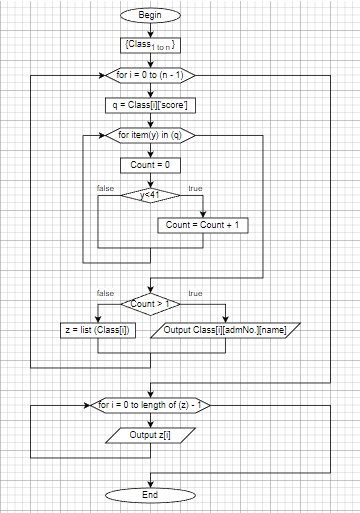
Dictionaries are used to store data values in key : value pairs. A dictionary is a collection which is *ordered, changeable* and *do not allow duplicates.* Duplicates which are not allowed are those of keys .

Dictionaries are written with curly brackets, and have keys and values: {‘key’: ‘value’}

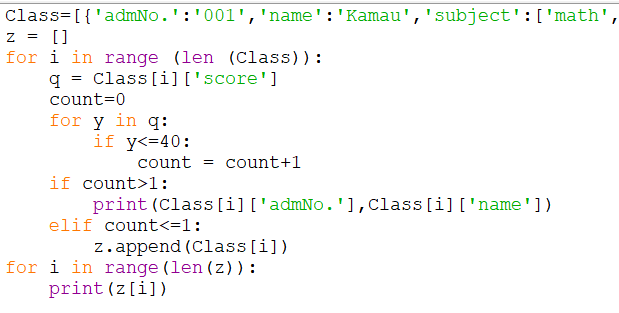
**Task (Dictionaries):**

A list of records is given. Each entry contains information about the student of a class: last name and grades in five subjects. Delete records of students who have more than one unsatisfactory grade. Print the surnames of these students.

**Solution:**

****

**Code:**

****

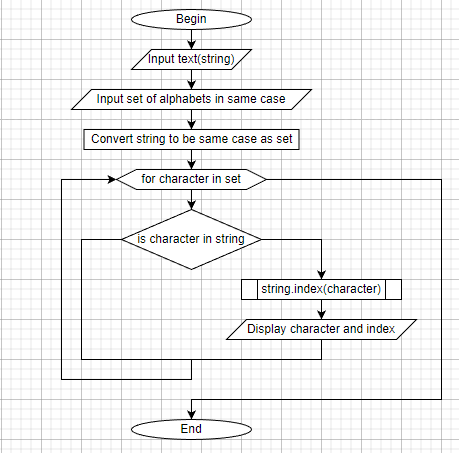
**Test data:**

****

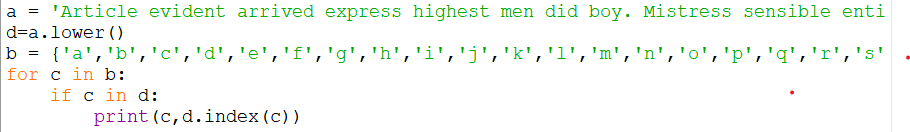
**Task (Sets):**

Given a text ending with a full stop, print the first occurrences of each letter of the Latin alphabet.

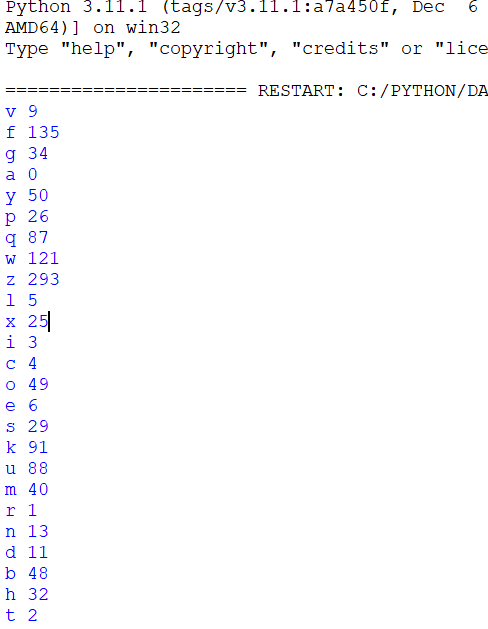
**Solution:**

****

**Code:**

****

**Test data:**



**Conclusion:**

All array types in python programming have been learnt and understood. Arrays are a useful programming tool in the sense that they consolidate a lot of data in one variable. If used correctly, arrays can help to protect integrity of data in programs.

Daniel Kamau Njoroge.

Naivasha, 2023.

**Files and File Methods in Python.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Purpose:**

* Learn different types of files.
* Learn how to create, access and modify files in python
* Adopt use of files in programming by solving given tasks.

**Theory:**

Python provides inbuilt functions for creating, writing, and reading files. There are two types of files that can be handled in python, normal text files and binary files (written in binary language, 0s, and 1s).

* **Text files:** In this type of file, Each line of text is terminated with a special character called EOL (End of Line), which is the new line character (‘\n’) in python by default. Text files are in a human-readable format and typically adhere to specific character sets, such as Unicode.
* **Binary files:** In this type of file, there is no terminator for a line, and the data is stored after converting it into machine-understandable binary language. A binary file is a file whose content is in a binary format consisting of a series of sequential bytes, each of which is eight bits in length. The content must be interpreted by a program or a hardware processor that understands in advance exactly how that content is formatted and how to read the data. Binary files include a wide range of file types, including executables, libraries, graphics, databases, archives.

**File Access Modes**

Access modes govern the type of operations possible in the opened file. It refers to how the file will be used once it’s opened. These modes also define the location of the **File Handle** in the file. File handle is like a cursor, which defines from where the data has to be read or written in the file. There are 6 access modes in python.

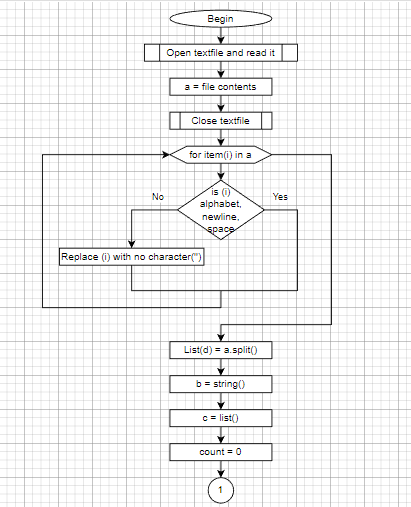
1. **Read Only (‘r’) :** Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exists, raises the I/O error. This is also the default mode in which a file is opened.
2. **Read and Write (‘r+’):** Open the file for reading and writing. The handle is positioned at the beginning of the file. Raises I/O error if the file does not exist.
3. **Write Only (‘w’) :** Open the file for writing. For the existing files, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist.
4. **Write and Read (‘w+’)** : Open the file for reading and writing. For an existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.
5. **Append Only (‘a’)**: Open the file for writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.
6. **Append and Read (‘a+’) :** Open the file for reading and writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.

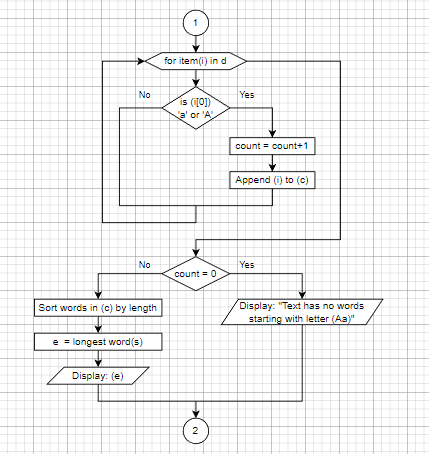
For binary files, a ‘b’ is added to the letter(s) denoting access mode in the above illustration. For example, if one wants to open a binary file in the read only mode one will use **(‘rb’).**

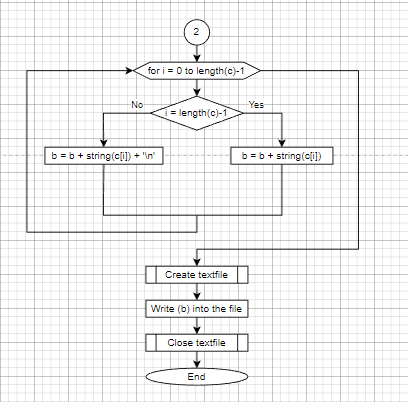
**Task1:**

Given a text file, find the longest word among the words starting with the letter "a". If there are no such words, print a message on the screen. Words starting with the letter "a" should be written to another file.

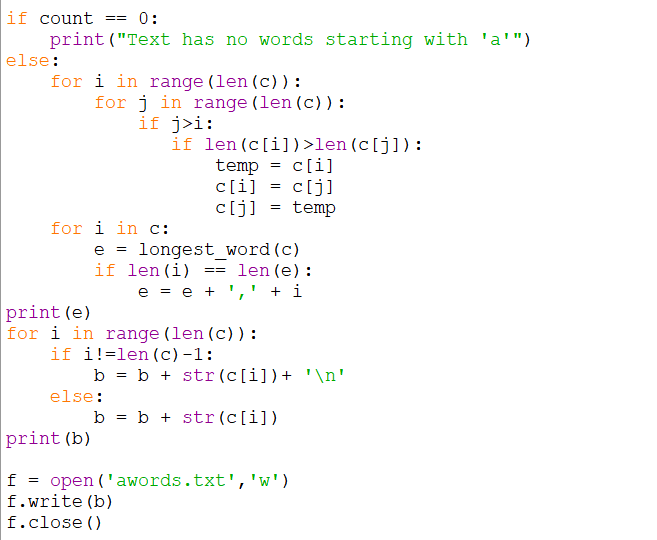
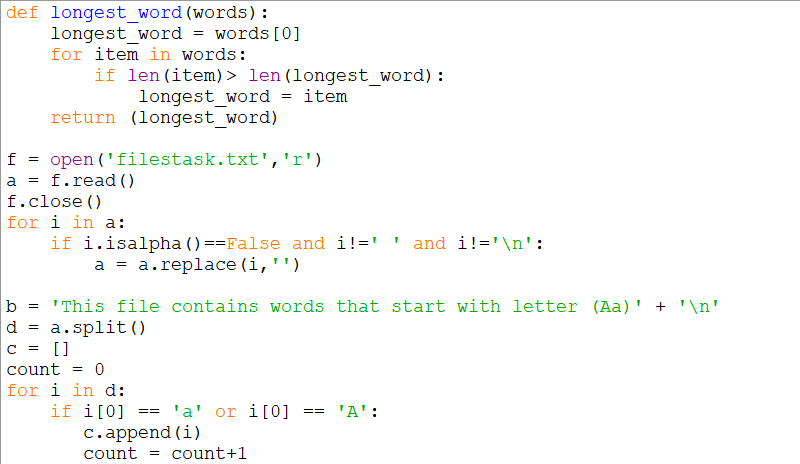
**Solution:**

****

****

****

**Code:**

****

**Test Data:**

****

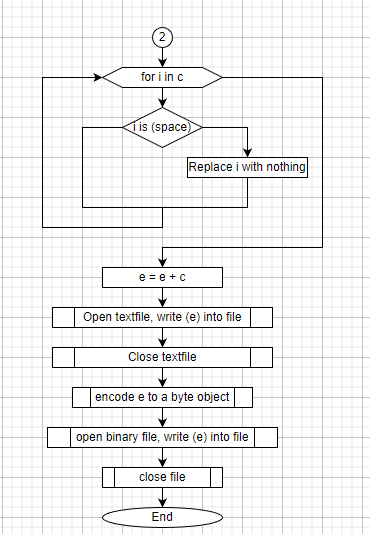
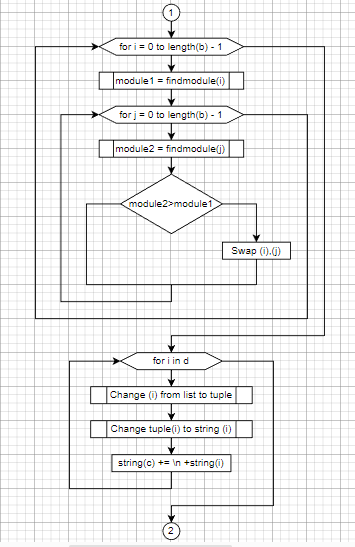
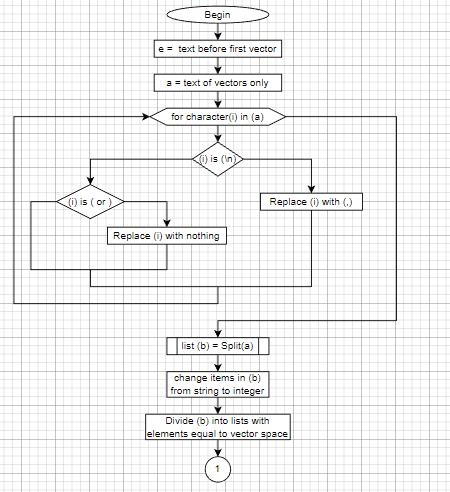
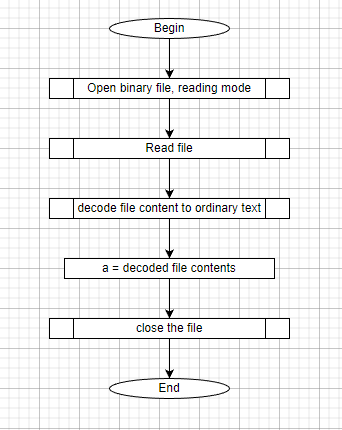
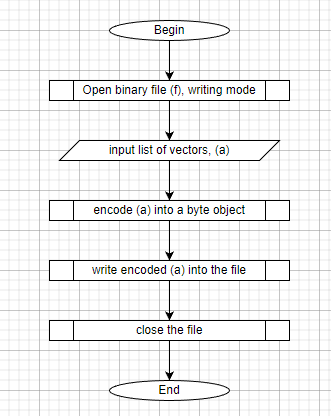
****

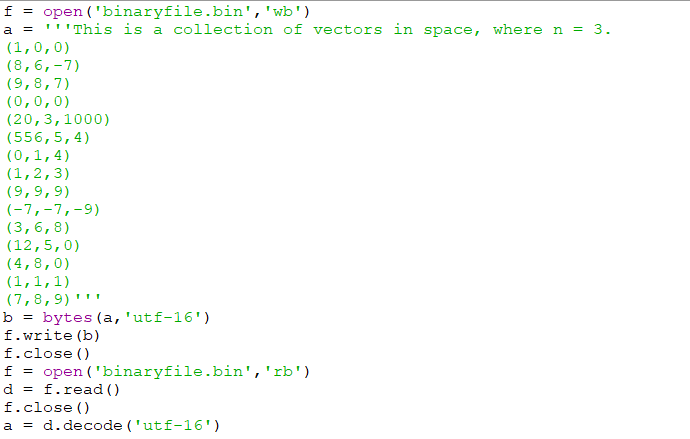
**Task2:**

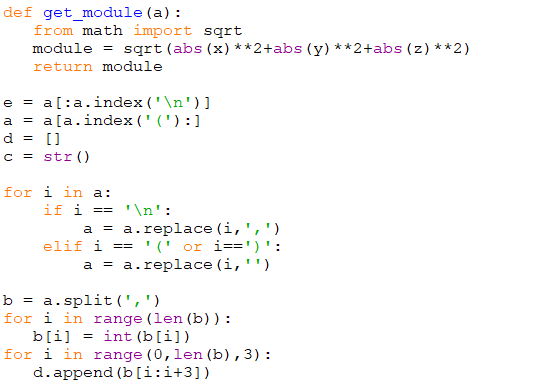
Given a file whose components are n-dimensional vectors (n- constant ).

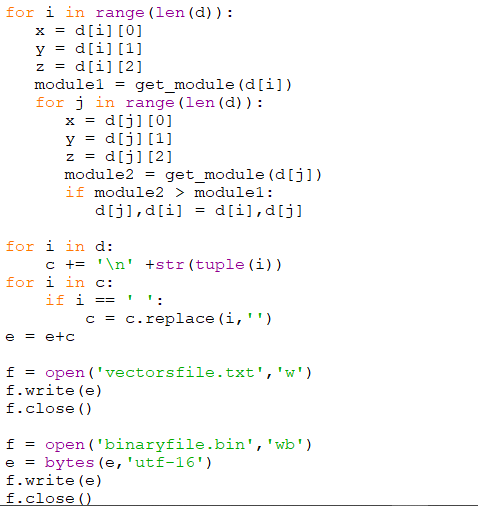
Move vectors with the largest module to the end of the file.

**Solution:**

****

**Code: **

****

****

**Test Data:**

****

****

****

**Conclusion:**

Files are a great tool in programming. They help in quick input of data and storage of information processed by running a program.

Of importance to note is that most files use strings as their primary data type. As such it is paramount to understand the basic string operations and how to utilize the foresaid operations in dealing with files.

The basic concepts of files have been learnt, comprehended and adopted to solve task(s). However, much remains to be covered so as to hone skills acquired.

Daniel Kamau Njoroge.

Naivasha, 2023.

**OBJECT ORIENTED PROGRAMMING.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Purpose:**

* To learn the basics of OOP paradigm.
* To identify strengths of OOP over procedural programming.
* To create classes in OOP and instantiate objects.
* Adopt OOP in tackling real life problems.

**Theory.**

Object Oriented Programming is one of the most popular paradigms of systems development in today’s computer world. OOP was developed to make programming easier by solving limitations associated with procedural programming.

Comparison of OOP and procedural programming

|  |  |
| --- | --- |
| **Procedural programming** | **Object oriented programming** |
| Emphasis on procedure | Emphasis on objects’ data |
| Uses top-down approach | Uses bottom-up approach |
| Code reusability is difficult | Ease of code reusability |
| Integrity of data is not guaranteed | Data hiding helps to maintain its integrity |

Code reusability is a major strength of OOP. It helps to solve time as similar problems can be solved with the same abstract code.

Basic concepts of OOP:

* **Class:** a set of object which shares common characteristics/ behavior and common properties/ attributes.  
  It is not a real world entity. It is just a template or blueprint or prototype from which objects are created and does not occupy memory.
* **Object:** basic unit of OOP and represents real-life entities consists of :

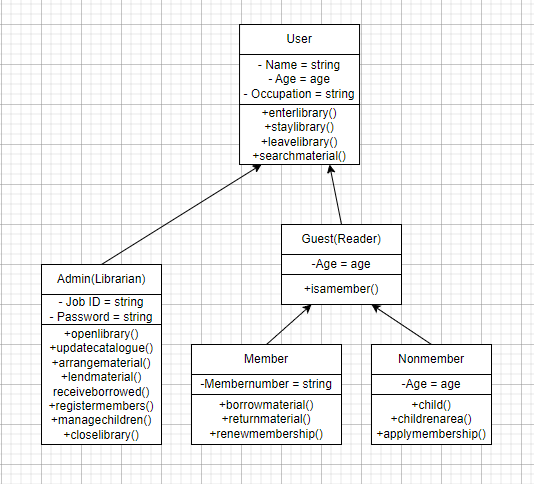
1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior**: It is represented by the methods of an object. It also reflects the response of an object with other objects.
3. **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

* **Inheritance:** process by which an object of one class (subclass) acquire properties of another class (superclass). It supports hierarchical classification and aids reusability. Additional features can be added to an existing class with little or no modification.Methods in subclass take precedence over methods in superclass. It also enables abstraction (representing essential features without including the background details or explanations); classes are abstract.
* **Encapsulation:** wrapping up of data and functions into a single unit. Data is only accessible to the functions wrapped with it. It helps in hiding data and protecting it from unintended manipulation.

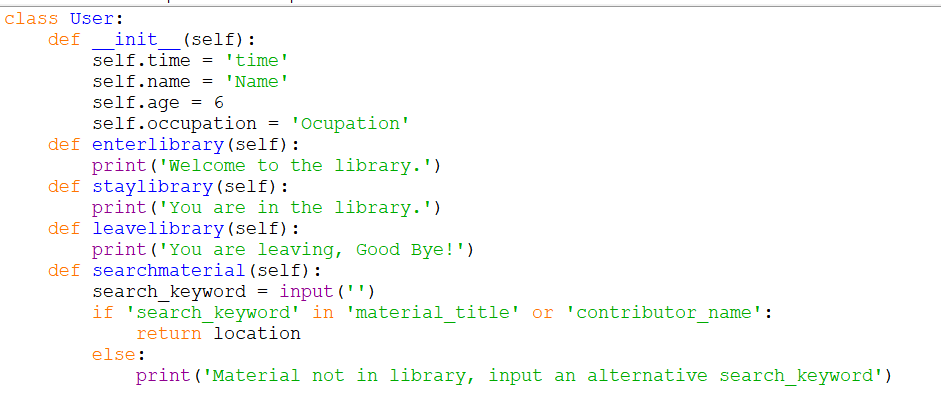
**Task:**

Create an object oriented program to show inheritance among library users.

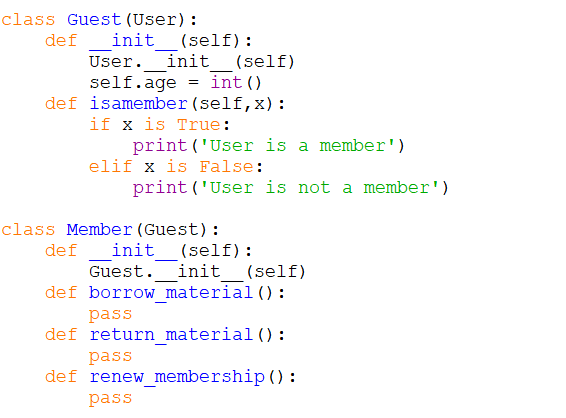
**Solution:**

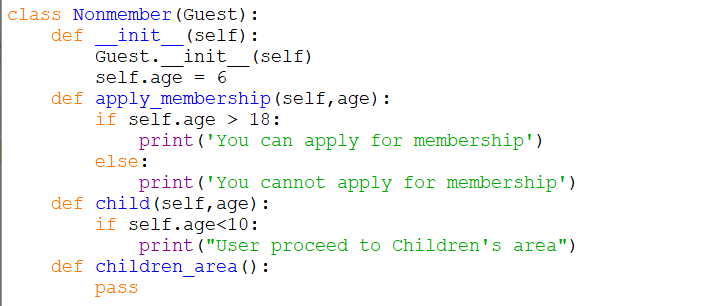
****

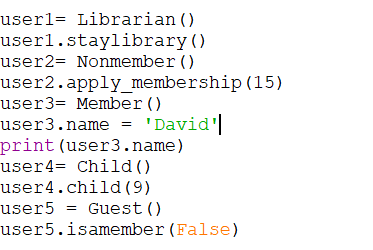
**Code:**

****

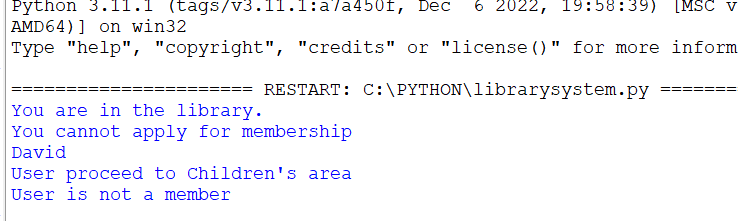
****

****

****



**Test data:**

****

**Conclusion.**

Creation of classes and instances has been adequately learned. Inheritance as a concept in object oriented programming has been learn, tested and adopted. A subclass cannot have more than one parent class.

Mastering inheritance and OOP at large is an indispensable skill for modern world programming.

Daniel Kamau Njoroge.

Naivasha, 2023.

**HANDS ON SKILLS**

**PROJECT1: SIMPLE SEA BATTLES GAME.**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Purpose:**

1. Learn to map real-world tasks to code-able entities
2. Learn to use Object Oriented Programming to map real-world entities to programmable logic.
3. Learn to decompose large tasks into smaller, simpler and easier to understand tasks.
4. Use acquired skills to solve a task(creating sea battles game)

**Theory**

Sea battles is a guessing board game. Two players prepare their battlefields by placing naval ships on different locations on the board. Opponents do not the location of ships, they play the game by guessing where the ships would be. The winner is the player who sinks all opponent’s ships first.

The main objects in the game are players, board and ships. All these objects will interact with each other to effect playing the game. The board presents the sea for the game.

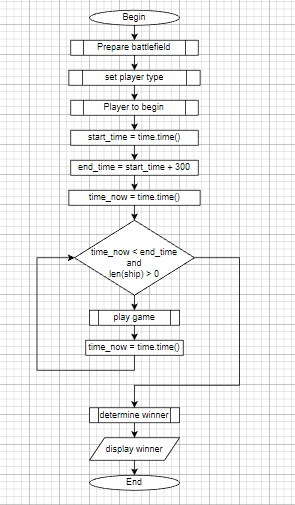
**Task:**

Two navies A (Player 1) and B (Player 2) each with x and y war ships respectively, start shooting at each other in the sea. You are required to develop an OOP model for this game. Make it possible for a single player to play against the computer.

Notes:

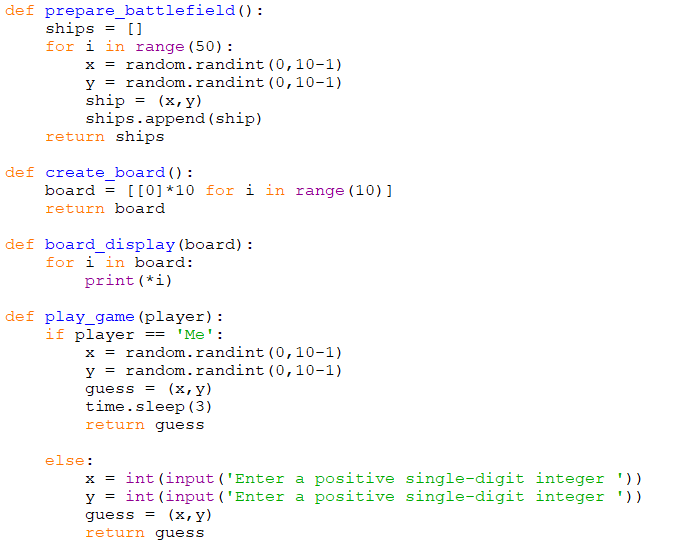
* A graphical user interface is not required.
* Use the knowledge accumulated over the learning period to model a working game model.
* Use random number generator function to generate random coordinates of the naval ships.
* One game should last for five minutes. After which the % of destroyed warships for each player is calculated and printed on the screen.
* The player with the highest % of own destroyed ships loses.

**Solution:**

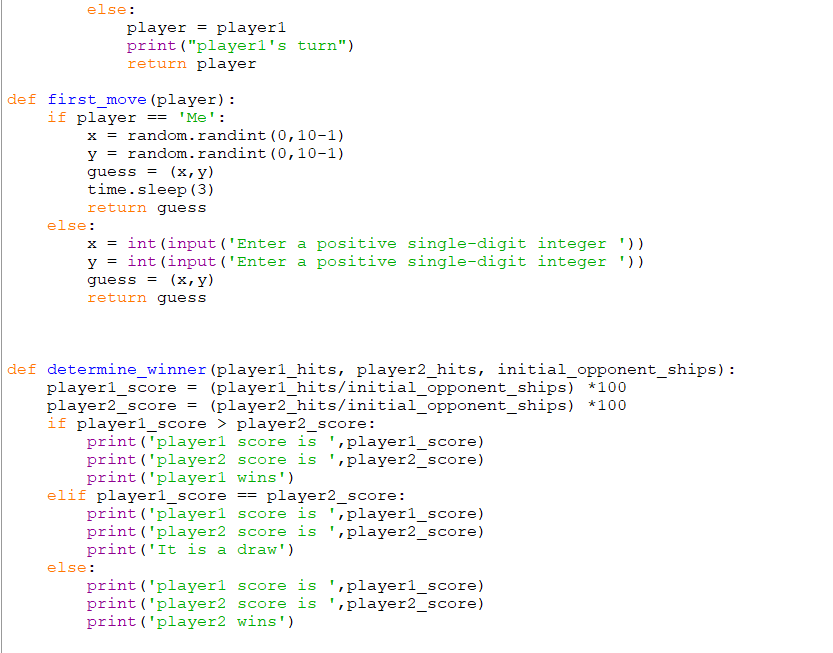
****

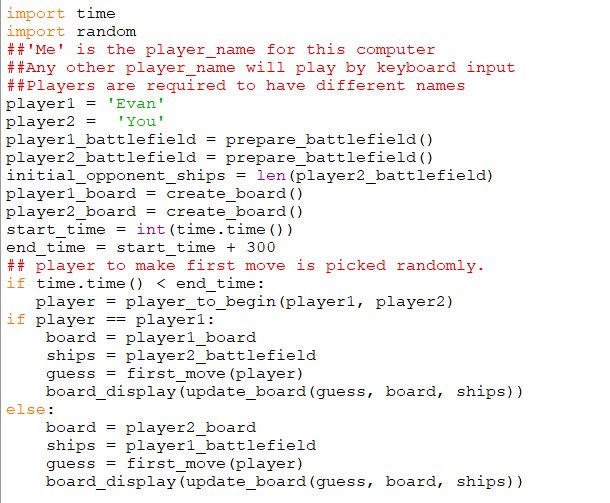
**Code:**

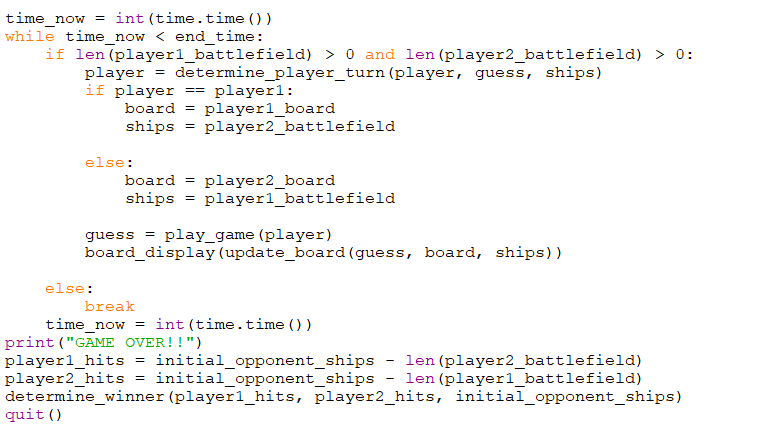
****

****

****

****

****

****

**Conclusion:**

The project has been a great insight matters programming. The learnt skills have been put to test in a daunting process and much has been achieved. Application of different data types and programming processes such as conditionals and loops has been very helpful in meeting the objectives laid out at the start of the project.

A simple battlefields game has been created and tested. Much still needs to be done to improve on the game structure and make it better.

Daniel Kamau Njoroge.

Naivasha, 2023.

**VERSION CONTROL SYSTEM, GIT AND GITHUB**

Daniel Kamau Njoroge.

Naivasha, 2023.

**Version control system (VCS)**

A system that records changes to a file or set of files over time so that you can recall specific versions later.

It allows one to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS means that if one screws things up or loses files, they can easily recover them.

* **Centralized Version Control Systems**

The next major issue that people encounter is that they need to collaborate with developers on other systems. To deal with this problem, Centralized Version Control Systems (CVCSs) were developed. These systems (such as CVS, Subversion, and Perforce) have a single server that contains all the versioned files, and a number of clients that check out files from that central place. For many years, this has been the standard for version control.

This setup offers many advantages. For example, everyone knows to a certain degree what everyone else on the project is doing. Administrators have fine-grained control over who can do what, and it’s far easier to administer a CVCS than it is to deal with local databases on every client.

However, this setup also has some serious downsides. The most obvious is the single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they’re working on. If the hard disk the central database is on becomes corrupted, and proper backups haven’t been kept, you lose absolutely everything — the entire history of the project except whatever single snapshots people happen to have on their local machines.

* **Distributed Version Control Systems**

In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don’t just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data. Many of these systems deal pretty well with having several remote repositories they can work with, so you can collaborate with different groups of people in different ways simultaneously within the same project. This allows you to set up several types of workflows that are not possible in centralized systems.