week 2

1) Programmer's perspective:

- We would be more precise by giving an appropriate name to a variable.
- We keep comments in the console using '#'. Comments are useful to make note of the process being done or to refer it when the code runs for n number of times.

2) Variable Dynamic Typing:

- "Dynamic" stands for things to change.

- In Python, the moment when declare a variable with integer, it does not mean to remain constant.

- We could change the variable to whatever datatype we want, that is, it provides flexibility for us to use a variable for something as an integer and then make it a string or any later on.

3) More on Variables, operators and expressions

- Key words can be defined as a part of programming language themselves.

- For this reason, keywords cannot be used as variable names.

- There is a set of rules which define what can be used as a variable name.

Rule 1: Only Alphanumeric characters and underscore "_" can be used in a variable name.

Rule 2: A variable name can start with an alphabet or an underscore "_" but not with an integer.

Rule 3: Variable names are case sensitive.

- Multiple assignment can be used when we want assign two or more values in a single line. There are 2 methods of multiple assignment:

1. using a comma in between the respective variables.
2. using an equal to symbol "=" for the values that are equal in input.

- We could alter the values in multiple assignment by declaring the variables in opposite order.

- We can delete a variable using "del ()" function.

- Shorthand operators are similar to arithmetic operators but have unique features. When we to increase or Update a variable by 1, we write, for example, x = x +1. Doing this repeatedly is a bit tedious and un-necessary process and this is where shorthand operators can be used. We could increment x with 1 by giving x += 1 simply.

- 'in' operator checks if a particular value exists in something that's defined earlier. The result of 'in' operator is either a True or False.
- When we use multiple relational operators in a single statement, then it is called as chaining operators.

4) Escape characters and types of quotes

- There are some characters that have a special meaning when used in a string. But what do you do if you would like to insert that character in the string as is, without invoking its meaning.

- For understanding this, let us take a simple example. We use single quotes or double quotes to define a string. Suppose, we define a string with single quotes.

- The first occurrence of single quote marks the start of string and the second occurrence marks the end of the string. Now, consider that we would like to have a single quote in our string. What do we do now. If we place a single quote just like that in the middle of string, Python would think that this is the end of the string, which is actually not.

- To insert these characters, we need the help of a special character like backslash ' \ '.

- We get an EOL (End Of Line) error when a string is not closed with a single/double quote.

- Triple quotes are used to store multiline strings.

- Following table provides the list of escape characters in Python.

Code	Description
\’	Single Quote
\”	Double Quote
\n	New Line
\t	Tab space

5) String Methods

- String Methods are nothing but functions or commands.
- There are String Methods in Python which are:
- A table is given below with the functioning and an example of each of the above String Methods.

Method	Description	Code x = ‘pytHoN sTrIng mEthOdS'	Output
lower()	Converts a string into lower case	print(x.lower())	python string methods
upper()	Converts a string into upper case	print(x.upper())	PYTHON STRING METHODS
capitalize()	Converts the first character to upper case	print(x.capitalize())	Python string methods
title()	Converts the first character of each word to upper case	print(x.title())	Python String Methods
swapcase()	Swaps cases, lower case becomes upper case and vice versa	print(x.swapcase())	PYThOn StRiNG MeTHoDs

Method	Description	Code	Output
isdigit()	Returns True if all characters in the string are digits	x = '123' print(x.isdigit())	True
		x = '123abc' print(x.isdigit())	False
isalpha()	Returns True if all characters in the string are in alphabets	x = 'abc' print(x.isalpha())	True
		x = 'abc123' print(x.isalpha())	False
isalnum()	Returns True if all characters in the string are alpha-numeric	x = 'abc123' print(x.isalnum())	True
		x = 'abc123@*#' print(x.isalnum())	False

Method	Description	Code	Output
islower()	Returns True if all characters in the string are lower case	x = 'python' print(x.islower())	True
		x = 'Python' print(x.islower())	False
isupper()	Returns True if all characters in the string are upper case	x = 'PYTHON' print(x.isupper())	True
		x = 'PYTHoN' print(x.isupper())	False
istitle()	Returns True if the string follows the rules of a title	x = 'Pyhton String Methods' print(x.istitle())	True
		x = 'Pyhton string methods' print(x.istitle())	False

Method	Description	Code x = '-----Python-----'	Output
strip()	Returns a trimmed version of the string	print(x.strip('-'))	Python
lstrip()	Returns a left trim version of the string	print(x.lstrip('-'))	Python-----
rstrip()	Returns a right trim version of the string	print(x.rstrip('-'))	-----Python

Method	Description	Code x = 'Python'
startswith()	Returns True if the string starts with the specified value	print(x.startswith('P')) print(x.startswith('p'))
endswith()	Returns True if the string ends with the specified value	print(x.endswith('n')) print(x.endswith('N'))

Method	Description	Code x = 'Python String Methods'	Output
count()	Returns the number of times a specified value occurs in a string	print(x.count('t')) print(x.count('s'))	3 1
index()	Searches the string for a specified value and returns the position of where it was found	print(x.index('t')) print(x.index('s'))	2 20
replace()	Returns a string where a specified value is replaced with a specified value	x = x.replace('S', 's') x = x.replace('M', 'm') print(x)	Python string methods

If – Else Loop and their Applications

- An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the if statement resolves to o or a FALSE value.

The *else* statement is an optional statement and there could be at most only one **else** statement following **if**.

- Syntax of if-else statement is :

```
If (condition 1):
        Statement(s1)
else:
    statement( s2 )
```

- A statement gets printed for every input that we give if it is present out of the loop.

Elif of If-Else conditional loop :

- The **elif** statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

- Similar to the **else**, the **elif** statement is optional. However, unlike **else**, for which there can be at most one statement, there can be an arbitrary number of **elif** statements following an **if**.

Libraries

- A library is a collection of existing functions that can be used in your code.

- The import keyword lets you import entire libraries or specific library functions into your code.

- ## Random Library and Math Library:

- The random library is a collection of functions that all have to do with randomization. This library uses a certain algorithm or equation to add randomness, so in a way, it is not true randomization. However, the library can be useful for small and personal projects

- Random library helps in simulating some cases where we face a face of probability like the cases of a coin toss, drawing a card from a pile etc.

- The math library is a collection of arithmetic operations that can be applied between two or more numbers.

- With the help of math library, we can easily make arithmetic calculations in the console without using a scientific calculator as an external application.

- ## Calendar library:

- The calendar library is used to get the calendar of a specific month of a particular year and also calendar for an year instantly.

- The syntax of this operation is calendar.month(year,month number) and calendar.calendar(year) in the case if we want calendar for one complete year .

- We can import a library into code using 'from' keyword which also does the user work with library in a easier way.

  Example would be in the case of printing month of a calendar we generally use calendar.month(year,month) after using import calendar statement.

Another easier approach would be: from calendar import month

print(month(2021,12))

- This would also give the same output which is surprising to us.

- import 'libraryname' as 'some alphabet' would help in saving time for the user.

- An example would be import calendar as c.

- We could also import only a required function from the library like from 'libraryname' import 'one of its component'.

- An example is : from calendar import month.