Data Type: Strings

A string is a **sequence** of characters (i.e. letters) which is delimited by a pair of single quotes or double quotes.
Eg. x = "Good Morning" means "Good Morning" is assigned to x and x is of data type string.

A. String operators (+, *, in**):**

Activity 1:

Enter the following programs and observe the output. Explain what + and * do in strings.

Instructions	Output	Functions of the operators
Enter the following instruction: x = "SPCS" y = "Secondary Section" print (x + y)		+ in string operation means _____ which _____ _____
Modify the third statement from above print (x+y) *so that it produces the output as shown in the next column.*	SPCS (Secondary Section) is my school	
Enter the following lines of codes: x = "Hi, girls!" Print (x * 3)		
Create the following program: string1="Banana" for letter in string1: print(letter)		The word "in" is a Boolean operator that takes two strings and returns TRUE if the first (i.e. letter) is a substring in the second (i.e. string1).

Create the following program:
string1="Banana is a nice fruit"
string2="I prefer watermelon"
for letter in string1:
  if letter in string2:
  print(letter, end=" ")
 	
#what does this program do?
		It prints all the letters from string1 which also appear in string2

B. String functions:

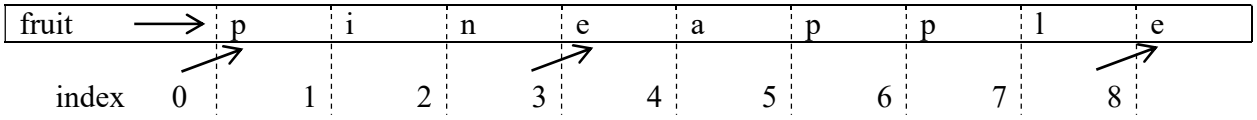
1. The length of string: len()
len() returns the number of characters in a string: for example

>>> x = 'Hi, girls!'

>>> len(x)
gives 10, for there are 10 characters in “Hi, girls!”

2. Accessing characters in string

Python does not support a *character* type; these are treated as strings of length one, or a substring. To access substrings, use the square brackets for slicing along with the index to obtain the substring.
For example



Fruit = 'pineapple'
letter = fruit[1] #gives the letter “i”
letter = fruit[4] #gives the letter “a”

Activity 2:

Enter the following instructions in interactive mode. Observe its results and try to explain why the result is as such.

Instructions	Python response	Explanation
fruit[0]		
fruit[7]		
fruit[10]		
fruit[-1]		
fruit[-4]		
fruit[2.5]		

So the index of a string starting from _____ to _____

Alternatively, you can use negative indices, which count backward from the end of the string. To get the last letter of the string, we can use fruit[-1]. The expression fruit[-1] yields the last letter, fruit[-2] yields the second to last, and so on.

3. Slicing string

A segment of a string is called a **slice**.

>>> s = “Good Morning”
>>> print s[1:4]
Good
>>> print s[5:11]
Morning

The operator [n:m] returns the part of the string from the “nth” character to the “mth-1” character, *including the first but excluding the last*. The indices are “pointing” between the characters, as in the above figure.
If you omit the first index (before the colon), the slice starts at the beginning of the string. If you omit the second index, the slice goes to the end of the string:

>>> fruit = 'pineapple'
>>> fruit[:4]
gives “pine”
>>> fruit[4:]
gives “apple”. However, if fruit = 'pineapple', you slice the string as fruit[3:3], it will give you an empty string (“ “). So if n>=m, ie. the first index is greater than or equal to the second, the result is an **empty string**, represented by two quotation marks: “ ”

Activity 3:
How are you going to slice the string, as shown below, to produce the output? Test it on the interactive Python.

String	Your code	Output
ss=”Congratulations on embarking a journey of coding! Are you enjoying it?” substring= *?????* print(substring)		journey
		barking
		Congratulations
		enjoying it?

4. String methods

A **method** is similar to a function, but the syntax is different. It uses *a dot notation*. For example, the method *upper* takes a string and returns a new string with all uppercase letters:

Program	output
word = 'happy birthday' newword = word.upper() print (newword)	HAPPY BIRTHDAY

It uses the method syntax word.upper(), instead of upper(word)

This form of dot notation specifies the name of the string (ie. word) to apply the method to and the name of the method (ie. upper) and the empty parentheses indicate that this method takes no argument.

Activity 4:

Enter the following instructions on interactive Python. Observe its result. Then replace the method in line 2 (the shaded method) of the program by each of the other methods. Then write down its output and explain what the method does.

Instructions	Output	Explanation
1 word = 'Happy Birthday' 2 newword = word.upper() 3 print (newword)		
word.lower()		
word.find(‘a’)		Able to find character enclosed by “ “

word.find('y')		Able to find character character enclosed by " "
word.find('y', 5, 15)		Find the second y. 5 refers the beginning of the search and 15 represents the end of search of the string
word.find('day')		Able to find substring
word.count('p')		Count the number of occurrence of 'p' Can be extended to word.count('p', start, end)

5. String comparison

The relational operators work on strings. To see if two strings are comparable, refer to the following program:

Program

```
word1 = "Good day!"
word2 = "congratulations!"
if word1>word2:
   print(word1, "should come before", word2)
else:
   print(word1, "should come before", word2)
```

Python does not handle uppercase and lowercase letters the same way that people do. All the uppercase letters come *before* all the lowercase letters.

A common way to address this problem is to convert strings to a standard format, such as all lowercase, before performing the comparison. Then a<c, and B<N

6. Looping and counting involving strings

The following program counts the number of times the letter "a" appears in a string:

Program

```
word = 'banana is a nice fruit'
count = 0
for letter in word:
   if letter == 'a':
      count = count + 1
print ("the letter 'a' appears ", count, "times")
```

This program demonstrates another pattern of computation called a **counter**. The variable count is *initialized* to 0 and then incremented each time an "a" is found. When the loop exits, count contains the result—the total number of a's.

Activity 5:

1. Modify the above program so that it counts the number of words in the word string.'
2. Enhance the above program by encapsulating the code in a function named lettercount, and generalize it so that it accepts any string and any letter as arguments. *Please save your program as countingletter.py and submit it to your teacher..*