google colaboratory

Week 1:

- Introduction to Python programming language
- - overview of data analysis
- Anaconda: In this lesson, you will be getting a quick glimpse at the Anaconda environment - one of the most popular environments for doing data analysis in Python.
- Jupyter Notebooks: Jupyter Notebooks are a great tool for getting started with writing python code. Though in production you often will write code in scripts, notebooks are wonderful for sharing insights and data viz! The Data Analysis Process Learn about the data analysis process and practice investigating different datasets using Python and its powerful packages for data analysis.
- Variables, data types, and basic operations
- Working with strings and string manipulation

In [ ]:

```
1
```

Variables and Datatypes

In [1]:

```
1 print("hello world!")
```

```
hello world!
```

In [2]:

```
1 number = 2
2 type(number)
3
4 name = 'caleb'
5 print(name)
```

```
caleb
```

In [3]:

```
1 print(type(number))
```

```
<class 'int'>
```

In [5]:

```
1 first_number = int(input("enter the value: "))
2 second_number = int(input("enter the second value: "))
```

```
enter the value: 3
enter the second value: 3
```

In [6]:

```
1 first_number, second_number
```

Out[6]:

```
(3, 3)
```

In [7]:

```
1 first_number * second_number
```

Out[7]:

```
9
```

In [8]:

```
1 result = first_number/second_number
2 print("the result is:", result)
```

```
the result is: 1.0
```

In [11]:

```
1 type(number)
2 num = int(input("enter the number : "))
```

```
enter the number : 4
```

In [12]:

```
1 print("W \bord")
```

```
Word
```

2, 2.0,wee

In [13]:

```
1 x =2.0
2 type(x)
```

Out[13]:

```
float
```

Python Escape Character Sequences

- ' ' Single Quote
- \ Backslash
- \n New Line
- \t Tab
- \b Backspace

Python Operators

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

https://www.w3schools.com/python/python_operators.asp
(https://www.w3schools.com/python/python_operators.asp)

== != > >= < <= is is not in not in : menbership operators

In [ ]:

```
1
```

division of integer

In [15]:

```
1 ##### first_number=8
2 second_number=int(input('the dividend is'))
3 result=first_number/second_number
4 print('the quotient:',result)
```

```
the dividend is4
the quotient: 0.75
```

addition of floats

In [16]:

```
1 first_number=45.8
2 second_number=int(input('the missing number is'))
3 result=first_number+second_number
4 print('the sum is:',result)
5
```

```
the missing number is6
the sum is: 51.8
```

subtraction of float from integer

In [17]:

```
1 first_number=56
2 second_number=float ( input('the second number is'))
3 result=first_number-second_number
4 print('the difference is:',result)
5
```

```
the second number is7
the difference is: 49.0
```

area of a cone(pi*radius cube*slant height)

In [18]:

```
pi=3.14
radius=int(input('the radius is'))
slant_height=5.6
result=pi*radius*slant_height
print('the area of cone is:',result)

```

```
the radius is8
the area of cone is: 140.672
```

area of a rectangle

In [19]:

```
pi=3.14
radius=3**3
slant_height=int(input ('the height is'))
result=pi*radius*slant_height
print('the area of cone is:',result)

```

```
the height is3
the area of cone is: 254.34
```

In [20]:

```
pi=3.14
radius=3**3
slant_height= float(input('the height is'))
result=pi*radius*slant_height
print('the area of cone is:',result)

```

```
the height is2
the area of cone is: 169.56
```

list

In [23]:

```
my_list = [1,2,3,4,5,6,7,8]
my_list.insert(1,5)
```

In [24]:

```
print(my_list)
```

```
[1, 5, 2, 3, 4, 5, 6, 7, 8]
```

In [25]:

```
#creating a list
list_1 =['b','r','i','g','h','t']
```

In [26]:

```
text = 'emmanuel'
l2 = list(text)
print(l2)
#converting string(ie numbers )to integers
num_text = '12345'
num_list = [int(x)for x in list(num_text)]
print(num_list)
```

```
['e', 'm', 'm', 'a', 'n', 'u', 'e', 'l']
[1, 2, 3, 4, 5]
```

In [27]:

```
a= 'bright'
list_2 =[a]
#printing out the values of your
print(list_1)
```

```
['b', 'r', 'i', 'g', 'h', 't']
```

list operations

- indexing,slicing operation:a way to refer the individual terms within an iterable by its position
- insert(): add items to a list in a specific position
- remove():removes the first occurence of the element with the specified value
- append(); it is a pre-defined method used to add a single item to certain collection types
- sort():arranging data in a particular order
- index():refer to a position within an ordered list
- min(), max()
- pop()
- clear()

In [28]:

```
#creating a list
score = [2,3,5,7,1,7,9]
#checking for the minimum value in a list
min(score)
```

Out[28]:

1

In [29]:

```
#checking for the maximum values in a list
max(score)
```

Out[29]:

```
9
```

In [30]:

```
#creating another list
my_list = [1,2,3,4]
print(my_list)
```

```
[1, 2, 3, 4]
```

In [31]:

```
#using the append funtion to add element to the end of the list
my_list.append(6)
print(my_list)
```

```
[1, 2, 3, 4, 6]
```

In [32]:

```
#remove
my_list.remove(2)
```

In [33]:

```
print(my_list)
```

```
[1, 3, 4, 6]
```

In [34]:

```
#Creating a list
list_2 = [1,4,2,5,7,4,0,9,8]
list_2.sort()
```

In [35]:

```
print(list_2)
```

```
[0, 1, 2, 4, 4, 5, 7, 8, 9]
```

In [36]:

```
min_score = min(list_2)
print(min_score)
```

```
0
```

In [37]:

```
max_score = max(list_2)
print(max_score)
```

```
9
```

In [38]:

```
#q2a##len is used in getting the exact length of the above given list
len(list_2)
```

Out[38]:

```
9
```

In [39]:

```
####list indexing/slicing operations
```

In [40]:

```
#1st element
list_2[0]
```

Out[40]:

```
0
```

In [41]:

```
list_2[1]
```

Out[41]:

```
1
```

In [42]:

```
list_2[-1]
```

Out[42]:

```
9
```

In [43]:

```
list_2
```

Out[43]:

```
[0, 1, 2, 4, 4, 5, 7, 8, 9]
```

In [44]:

```
#slicing operations
list_2[2:5]
```

Out[44]:

```
[2, 4, 4]
```

In [45]:

```
list_2[2:4]
```

Out[45]:

```
[2, 4]
```

In [46]:

```
list_2[2:5]
```

Out[46]:

```
[2, 4, 4]
```

insert is used to add an element to the list

In [47]:

```
#in this case insert is used to add 2 and 6
list_2.insert(-1,12)
print(list_2)
```

```
[0, 1, 2, 4, 4, 5, 7, 8, 12, 9]
```

In [48]:

```
str_list = ["mango", 'banana', 'orange']
```

In [49]:

```
#remove is used to abolish 4 from the list
list_2.remove(4)
```

In [50]:

```
#this is calling list 2 to find out if the recent activity  has been performe
list_2
```

Out[50]:

```
[0, 1, 2, 4, 5, 7, 8, 12, 9]
```

In [51]:

```
#append is used to add a single element to a list
list_2.append(25)
```

In [52]:

```
list_2
```

Out[52]:

```
[0, 1, 2, 4, 5, 7, 8, 12, 9, 25]
```

In [53]:

```
#sort is used to arrange data in a particular order
list_2.sort()
```

In [54]:

```
list_2
```

Out[54]:

```
[0, 1, 2, 4, 5, 7, 8, 9, 12, 25]
```

In [55]:

```
#it is used to refer to a position in a list
list_2.index(4)
```

Out[55]:

```
3
```

In [56]:

```
#it is used to identify the lowest value
min(list_2)
```

Out[56]:

```
0
```

In [57]:

```
#it is used to identify the highest value
max(list_2)
```

Out[57]:

```
25
```

In [58]:

```
#it removes an item from a specific index

list_2.pop(3)
```

Out[58]:

```
4
```

In [59]:

```
list_2
```

Out[59]:

```
[0, 1, 2, 5, 7, 8, 9, 12, 25]
```

In [60]:

```
#it is used to remove all items on a list
list_2.clear()
```

In [61]:

```

list_2
```

Out[61]:

```
[]
```

In [62]:

```
#creating a new list
list_3=[1,2,3,4,5,6,7,8]
list_4 = ['2','4']
```

In [63]:

```
#extend can be used to join two lists together
list_3.extend(list_4)
```

In [64]:

```
#printing the outcome
print('list after extend():',list_3)
```

```
list after extend(): [1, 2, 3, 4, 5, 6, 7, 8, '2', '4']
```

In [65]:

```
l_4 = [1,2,3,4,5,6,7,8]
```

In [66]:

```
l_4[:3]
```

Out[66]:

```
[1, 2, 3]
```

TUPLE

> Tuple is a collection which is ordered and unchangeable. it is used to store multiple items in a single variable.

In [67]:

```
num = '2'
int(num)*5
```

Out[67]:

```
10
```

In [68]:

```
score = (1,2,3)

```

In [69]:

```
#creating a tuple
tuple_1=('ememm','lionel','blessing')
print(tuple_1)
print(type(tuple_1))
```

```
('ememm', 'lionel', 'blessing')
<class 'tuple'>
```

In [70]:

```
#switching a tuple to a list
list_8=list(tuple_1)
print(list_8)
print(type(list_8))
```

```
['ememm', 'lionel', 'blessing']
<class 'list'>
```

In [71]:

```
type(list_8)
```

Out[71]:

```
list
```

In [72]:

```
#switching a list to a tuple
tuple_1=tuple(list_8)
print(tuple_1)
```

```
('ememm', 'lionel', 'blessing')
```

In [ ]:

```

```

In [73]:

```
####Dealing with set2
fruits_1={'orange','mango','watermelon',"salt",'pear','avocado'}
fruits_2={'strawberry','blueberry','pear','mango','raspberries', "salt"}
fruits_2.discard("pear")
print(fruits_2)
```

```
{'raspberries', 'salt', 'strawberry', 'blueberry', 'mango'}
```

In [74]:

```
#Using the update set operation
# fruits_2=list(fruits_2)
# print(fruits_2)
fruits_1.update(("fruits_2", "grains",))
print(fruits_1)
```

```
{'orange', 'pear', 'watermelon', 'avocado', 'fruits_2', 'salt', 'gra
ins', 'mango'}
```

In [75]:

```
#Using the remove set operation
fruits_1.remove("mango")
```

In [76]:

```
#calling fruits_1
fruits_1
```

Out[76]:

```
{'avocado', 'fruits_2', 'grains', 'orange', 'pear', 'salt', 'waterme
lon'}
```

In [77]:

```
#using the union set operation
fruits_1.union(fruits_2)
```

Out[77]:

```
{'avocado',
 'blueberry',
 'fruits_2',
 'grains',
 'mango',
 'orange',
 'pear',
 'raspberries',
 'salt',
 'strawberry',
 'watermelon'}
```

In [78]:

```
#Using the union sign to write a code
fruits_1|fruits_2
```

Out[78]:

```
{'avocado',
 'blueberry',
 'fruits_2',
 'grains',
 'mango',
 'orange',
 'pear',
 'raspberries',
 'salt',
 'strawberry',
 'watermelon'}
```

In [79]:

```
#using the intersection set operations
fruits_1.intersection(fruits_2)
```

Out[79]:

```
{'salt'}
```

In [80]:

```
#using the intersection sign to write a code
fruits_1 & fruits_2
```

Out[80]:

```
{'salt'}
```

In [81]:

```
#using the difference sign to write the code
print(fruits_1,'\n\n',fruits_2)
fruits_1 - fruits_2
```

```
{'orange', 'pear', 'watermelon', 'avocado', 'fruits_2', 'salt', 'gra
ins'}

 {'raspberries', 'salt', 'strawberry', 'blueberry', 'mango'}
```

Out[81]:

```
{'avocado', 'fruits_2', 'grains', 'orange', 'pear', 'watermelon'}
```

In [82]:

```
#Using the difference set operation
fruits_1.difference(fruits_2)
```

Out[82]:

```
{'avocado', 'fruits_2', 'grains', 'orange', 'pear', 'watermelon'}
```

In [83]:

```
#USING THE SYMMETRIC DIFFERENCE
fruits_1 ^ fruits_2
```

Out[83]:

```
{'avocado',
 'blueberry',
 'fruits_2',
 'grains',
 'mango',
 'orange',
 'pear',
 'raspberries',
 'strawberry',
 'watermelon'}
```

In [84]:

```
fruits_1.symmetric_difference(fruits_2)
```

Out[84]:

```
{'avocado',
 'blueberry',
 'fruits_2',
 'grains',
 'mango',
 'orange',
 'pear',
 'raspberries',
 'strawberry',
 'watermelon'}
```

In [85]:

```
a={2,3}
b={9,8,7,0,2,3,4}
```

In [86]:

```
a.issubset(b)
```

Out[86]:

```
True
```

In [87]:

```
b.issuperset(a)
```

Out[87]:

```
True
```

In [88]:

```
1 a.isdisjoint(b)
```

Out[88]:

```
False
```

In [ ]:

```
1
```