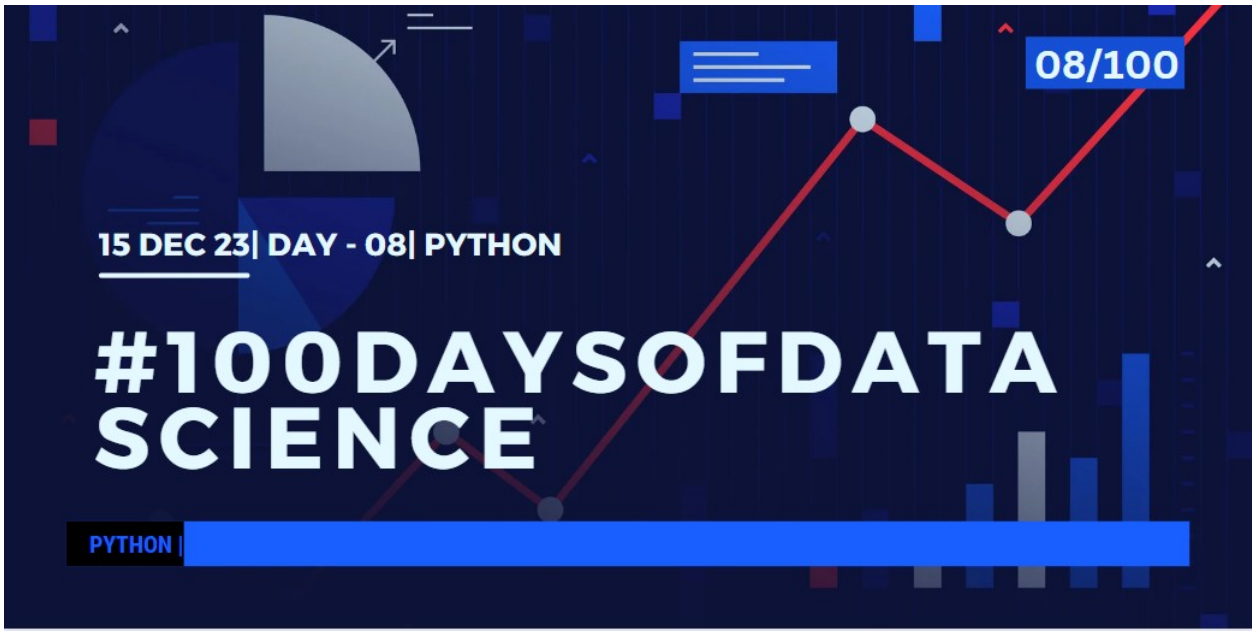

Some Important Python Concepts

Keywords

Python keywords are special reserved words that have specific meanings and purposes and can't be used for anything but those specific purposes.

```
import keyword as kw
print(kw.kwlist)
print(len(kw.kwlist))
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await',
'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except',
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try',
'while', 'with', 'yield']
35
```

isinstance()

The isinstance() function returns True if the specified object is of the specified type, otherwise False.

```
x = isinstance(5, int)
y = isinstance("TRUE", (bool))
```

```
print(x)
print(y)
```

```
True
False
```

enumerate()

- The enumerate() function takes a collection and returns it as an enumerate object.
- The enumerate() function adds a counter as the key of the enumerate object.
- The enumerate () method adds a counter to an iterable and returns it in the form of an enumerating object.
- Syntax: enumerate(iterable, start=0)

```
l1 = ["Grepes", "Apple", "Watermelon", "Pineapple", "Kiwi"]
s1 = "Zahid"

obj1 = enumerate(l1)
obj2 = enumerate(s1)

print ("Return type:", type(obj1))
print (list(enumerate(l1)))
print("----------------------------------------------------------------
---------------------")
print (list(enumerate(s1, 7)))
print("----------------------------------------------------------------
---------------------")
for i in obj2:
    print(i)
```

```
Return type: <class 'enumerate'>
[(0, 'Grepes'), (1, 'Apple'), (2, 'Watermelon'), (3, 'Pineapple'), (4,
'Kiwi')]
------------------------------------------------------------------------
--------------
[(7, 'Z'), (8, 'a'), (9, 'h'), (10, 'i'), (11, 'd')]
------------------------------------------------------------------------
--------------
(0, 'Z')
(1, 'a')
(2, 'h')
(3, 'i')
(4, 'd')
```

strip()

The strip() method removes any leading, and trailing whitespaces.

lstrip()

Removes all leading whitespace in string.

rstrip()

Removes all trailing whitespace of string.

```
mystr = "      Zahid Salim Shaikh      "

normal_strip = mystr.strip()
print(normal_strip)

left_strip = mystr.lstrip()
print(left_strip)

right_strip = mystr.rstrip()
print(right_strip)
```

```
Zahid Salim Shaikh
Zahid Salim Shaikh
      Zahid Salim Shaikh
```

The strip() function is also used to remove a specific set of characters from a string.

```
txt = ",,,,,rrttgg.....Zahid....rrr"
x = txt.strip(",.grt")
print(x)
```

```
Zahid
```

```
mystr = "*********Hello World**********"
print(mystr)
normal_strip = mystr.strip("*")
print(normal_strip)
```

```
*********Hello World**********
Hello World
```

lower()

The lower() method returns the lowercased string from the given string. It converts all uppercase characters to lowercase python. If no uppercase characters exist, it returns the original string.

```
mystr = "Zahid Salim Shaikh"
print(mystr.lower())
```

```
zahid salim shaikh
```

upper()

The upper() method returns the uppercased string from the given string. It converts all lowercase characters to uppercase python. If no lowercase characters exist, it returns the original string.

```
mystr = "Zahid Salim Shaikh"
print(mystr.upper())
```

```
ZAHID SALIM SHAIKH
```

title()

title() function is used to change the initial character in each word to Uppercase and the subsequent characters to Lowercase and then returns a new string.

```
mystr = "zahid salim shaikh"
print(mystr.title())
```

```
Zahid Salim Shaikh
```

swapcase()

The swapcase() method returns a string where all the upper case letters are lower case and vice versa.

```
mystr = "Zahid Salim Shaikh"
print(mystr.swapcase())
```

```
zAHID sALIM sHAIKH
```

isalpha()

- The isalpha() method returns True if all the characters are alphabet letters (a-z).
- Example of characters that are not alphabet letters: (space)!#%&? etc.

```
mystr1 = "ZahidSalimShaikh"
print(mystr1.isalpha())
mystr2 = "Zahid Salim Shaikh" #in this there is special character
(space)
print(mystr2.isalpha())

True
False
```

isdigit()

- The isdigit() method returns True if all the characters are digits, otherwise False.
- Exponents, like ², are also considered to be a digit.

```
y = '12345'
print(y.isdigit())

True
```

isalnum()

- The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).
- Example of characters that are not alphanumeric: (space)!#%&? etc.

```
password = "passw0rd123"
x = password.isalnum()
print(x)

True
```

replace()

The replace() method replaces a specified phrase with another specified phrase.

```
new_str = "Hello World"
print(new_str.replace("Hello" , "Hey"))
print("-----------")
print(new_str.replace(" " , "")) # Remove all whitespaces using
replace function
```

```
Hey World
-----------
HelloWorld
```

count()

The count() method returns the number of elements with the specified value.

```
mylist = ["one", "two", "Three", "one", "two", "one", "three", "One",
"Two", "Three", "three", "One", "Two"]
print("Number of one: ",mylist.count("one"))  # Number of times
substring "one" occurred in string.
print("------------------")
print("Number of One: ",mylist.count("One"))  # Number of times
substring "One" occurred in string.
```

```
Number of one:  3
------------------
Number of One:  2
```

startswith()

The startswith() method returns True if the string starts with the specified value, otherwise False.

endswith()

The endswith() method returns True if the string ends with the specified value, otherwise False.

```
mystr = "one two three one two two three one two three one two two
three"
print(mystr.startswith("one")) # Return boolean value True if string
starts with "one"
print("-----")
print(mystr.endswith("three"))  # Return boolean value True if string
ends with "three"
```

```
True
-----
True
```

split()

- The split() method splits a string into a list.
- You can specify the separator, default separator is any whitespace.

```
mystr = "one two three four one two two three five five six seven six
seven one eight two five seven eight"
print(mystr.split())
```

```
['one', 'two', 'three', 'four', 'one', 'two', 'two', 'three', 'five',
'five', 'six', 'seven', 'six', 'seven', 'one', 'eight', 'two', 'five',
'seven', 'eight']
```

```
mystr =
"one#two#three#four#one#two#seven#one#eight#two#five#seven#eight"
print(mystr.split("#",9)) # setting the maxsplit parameter to 9
```

```
['one', 'two', 'three', 'four', 'one', 'two', 'seven', 'one', 'eight',
'two#five#seven#eight']
```

find() & index()

- The find() method finds the first occurrence of the specified value.
- The find() method returns -1 if the value is not found.
- The find() method is almost the same as the index() method, the only difference is that the index() method raises an exception if the value is not found.

```
mystr = "one two three four five six seven eight nine ten"

loc1 = mystr.find("seven") # Find the location of word 'seven' in the
string "mystr"
print(loc1)

loc2 = mystr.find("eleven") # 'eleven' is not present then it will
give -1
print(loc2)

loc3 = mystr.index("seven") # Find the location of word 'seven' in the
string "mystr"
print(loc3)
```

```
28
-1
28
```

Python Membership Operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

```
mystr = "one two three four five six seven eight nine ten"
if 'seven' in mystr: # Check if 'seven' exist in the list
    print('seven is present in the string')
else:
    print('seven is not present in the string')
```

```
seven is present in the string
```

```
mystr = "one two three four five six seven eight nine ten"
print('seven' not in mystr)
```

```
False
```

iter()

- iter() method returns the iterator object, it is used to convert an iterable to the iterator.
- Properties of Iterators
  - The iteration object remembers the iteration count via the internal count variable.
  - Once the iteration is complete, it raises a StopIteration exception and the iteration count cannot be reassigned to 0.
  - Therefore, it can be used to traverse the container just once.

next()

- The next() function returns the next item in an iterator.
- You can add a default return value, to return if the iterable has reached to its end.

```
x = iter(["grapes", "apple", "mango", "cherry", "kiwi"]) # directly converting list using iter()
# prints values one by one using next()
print(next(x))
print(next(x))
print(next(x))
```

```
print(next(x))
print(next(x))
```

```
grapes
apple
mango
cherry
kiwi
```

```
x = ["Grapes", "Apple", "Mango", "Cherry", "Kiwi",
     "Banana", "Watermelon", "Pineapple", "Peach",
     "Dates", "Fig", "Pomegranate"]

# converting list using iter()
x = iter(x)

# prints this
print("Values at 1st iteration : ")
for i in range(0, 4):
    print(next(x))
print()
print("Values at 2nd iteration : ")
for i in range(0, 4):
    print(next(x))
print()
print("Values at 3rd iteration : ")
for i in range(0, 4):
    print(next(x))
```

```
Values at 1st iteration :
Grapes
Apple
Mango
Cherry

Values at 2nd iteration :
Kiwi
Banana
Watermelon
Pineapple

Values at 3rd iteration :
Peach
Dates
Fig
Pomegranate
```

```
mylist = iter(["Grapes", "Apple", "Mango", "Cherry", "Kiwi"])
x = next(mylist, "Kiwi")
print(x)
```

```
x = next(mylist, "Kiwi")
print(x)
x = next(mylist, "Kiwi")
print(x)
x = next(mylist, "Kiwi")
print(x)
x = next(mylist, "Kiwi")
print(x)
x = next(mylist, "Kiwi")
print(x)
```

```
Grapes
Apple
Mango
Cherry
Kiwi
Kiwi
```

all()

- All Returns true if all of the items are True (or if the iterable is empty).
- All can be thought of as a sequence of AND operations on the provided iterables.

any()

- Any Returns true if any of the items is True and returns False if empty or all are false.
- Any can be thought of as a sequence of OR operations on the provided iterables.

```
L1 = [0,1,2,3,4,True,False]
print(all(L1))  # Returns false as two values are false and 0
print(any(L1)) # Will Return True as we have items in the list with
True value
```

```
False
True
```

```
L2 = [1,2,3,4,True,5]
print(all(L2))  # Will return True as all items in the list are True
print(any(L2)) # Will Return True as we have items in the list with
True value
```

```
True
True
```