How to remove whitespaces from a string in Python?

To remove the whitespaces and trailing spaces from the string, Python providies strip([str]) built-in function. This function returns a copy of the string after removing whitespaces if present. Otherwise returns original string.

Example:

```
string = "  javatpoint "
string2 = "    javatpoint       "
string3 = "       javatpoint"
print(string)
print(string2)
print(string3)
print("After stripping all have placed in a sequence:")
print(string.strip())
print(string2.strip())
print(string3.strip())
```

Output:

```
javatpoint
    javatpoint
       javatpoint
After stripping all have placed in a sequence:
Javatpoint
javatpoint
javatpoint
```

How to remove leading whitespaces from a string in the Python?

To remove leading characters from a string, we can use lstrip() function. It is Python string function which takes an optional char type parameter. If a parameter is provided, it removes the character. Otherwise, it removes all the leading spaces from the string.

Example:

```
string = "  javatpoint "
string2 = "    javatpoint       "
print(string)
print(string2)
print("After stripping all leading whitespaces:")
```

```
6. print(string.lstrip())
7. print(string2.lstrip())
```

Output:

```
javatpoint
    javatpoint
After stripping all leading whitespaces:
javatpoint
javatpoint
```

Why do we use join() function in Python?

The join() is defined as a string method which returns a string value. It is concatenated with the elements of an iterable. It provides a flexible way to concatenate the strings. See an example below.

Example:

```
1. str = "Rohan"
2. str2 = "ab"
3. # Calling function
4. str2 = str.join(str2)
5. # Displaying result
6. print(str2)
```

Output:

```
aRohanb
```

an example of shuffle() method?

This method shuffles the given string or an array. It randomizes the items in the array. This method is present in the random module. So, we need to import it and then we can call the function. It shuffles elements each time when the function calls and produces different output.

Example:

```
1. # import the random module
2. import random
3. # declare a list
```

```
4. sample_list1 = ['Z', 'Y', 'X', 'W', 'V', 'U']
5. print("Original LIST1: ")
6. print(sample_list1)
7. # first shuffle
8. random.shuffle(sample_list1)
9. print("\nAfter the first shuffle of LIST1: ")
10. print(sample_list1)
11. # second shuffle
12. random.shuffle(sample_list1)
13. print("\nAfter the second shuffle of LIST1: ")
14. print(sample_list1)
```

Output:

```
Original LIST1:
['Z', 'Y', 'X', 'W', 'V', 'U']

After the first shuffle of LIST1:
['V', 'U', 'W', 'X', 'Y', 'Z']

After the second shuffle of LIST1:
['Z', 'Y', 'X', 'U', 'V', 'W']
```

Which are the file related libraries/modules in Python?

The Python provides libraries/modules that enable you to manipulate text files and binary files on the file system. It helps to create files, update their contents, copy, and delete files. The libraries are os, os.path, and shutil.

Here, os and os.path - modules include a function for accessing the filesystem

while shutil - module enables you to copy and delete the files

What are the different file processing modes supported by Python?

Python provides **four** modes to open files. The read-only (r), write-only (w), read-write (rw) and append mode (a). 'r' is used to open a file in read-only mode, 'w' is used to open a file in write-only mode, 'rw' is used to open in reading and write mode, 'a' is used to open a file in append mode. If the mode is not specified, by default file opens in read-only mode

What is an operator in Python?

An operator is a particular symbol which is used on some values and produces an output as a result. An operator works on operands. Operands are numeric literals or variables which hold some values. Operators can be unary, binary or ternary. An operator which requires a single operand known as a **unary operator**, which require two operands known as a **binary operator** and which require three operands is called **ternary operator**

ternary operator.

Example:

```
1. # Unary Operator
2. A = 12
3. B = -(A)
4. print (B)
5. # Binary Operator
6. A = 12
7. B = 13
8. print (A + B)
9. print (B * A)
10. #Ternary Operator
11. A = 12
12. B = 13
13. min = A if A < B else B
14.
15. print(min)
```

Output:

```
# Unary Operator
-12
# Binary Operator
25
156
# Ternary Operator
12
```

What are the different types of operators in Python?

Python uses a rich set of operators to perform a variety of operations. Some individual operators like membership and identity operators are not so familiar but allow to perform operations.

- Arithmetic OperatorsRelational Operators
- Assignment Operators
- Logical Operators
- Membership Operators
- Identity Operators
- Bitwise Operators

How to create a Unicode string in Python?

In Python 3, the old Unicode type has replaced by "str" type, and the string is treated as Unicode by default. We can make a string in Unicode by using art.title.encode("utf-8") function.

:Example

```
unicode_1 = ("\u0123", "\u2665", "\U0001f638", "\u265E", "\u265F", "\u2168")
print (unicode_1)
```

Output:

```
unicode_1: ('ģ', '♥', '😸', '♞', '♟', 'Ⅸ')
```

what are the differences between Python 2.x and Python 3.x?

The most visible difference between Python2 and Python3 is in print statement (function). In Python 2, it looks like print "Hello", and in Python 3, it is print ("Hello").

String in Python2 is ASCII implicitly, and in Python3 it is Unicode.

The xrange() method has removed from Python 3 version. A new keyword as is introduced in Error handling.

How Python does Compile-time and Run-time code checking?

In Python, some amount of coding is done at compile time, but most of the checking such as type, name, etc. are postponed until code execution. Consequently, if the Python code references a user-defined function that does not exist, the code will compile successfully. The Python code will fail only with an exception when the code execution path does not exist

What is the usage of enumerate () function in Python?

The enumerate() function is used to iterate through the sequence and retrieve the index position and its corresponding value at the same time.

Example:

```
1. list_1 = ["A","B","C"]
2. s_1 = "Javatpoint"
3. # creating enumerate objects
4. object_1 = enumerate(list_1)
5. object_2 = enumerate(s_1)
6.
7. print ("Return type:",type(object_1))
8. print (list(enumerate(list_1)))
9. print (list(enumerate(s_1)))
```

Output:

```
Return type:
[(0, 'A'), (1, 'B'), (2, 'C')]
[(0, 'J'), (1, 'a'), (2, 'v'), (3, 'a'), (4, 't'), (5, 'p'), (6, 'o'), (7,
'i'), (8, 'n'), (9, 't')]
```

Why do lambda forms in Python not have the statements?

Lambda forms in Python does not have the statement because it is used to make the new function object and return them in runtime

How do you achieve multithreading in Python?

1. Python has a multi-threading package but if you want to multi-thread to speed your code up, then it's usually not a good idea to use it.
2. Python has a construct called the Global Interpreter Lock (GIL). The GIL makes sure that only one of your 'threads' can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread.
3. This happens very quickly so to the human eye it may seem like your threads are executing in parallel, but they are really just taking turns using the same CPU core.
4. All this GIL passing adds overhead to execution. This means that if you want to make your code run faster then using the threading package often isn't a good idea

What is the `@property` in Python?

The `@property` is a decorator. In Python, decorators enable users to use the class in the same way (irrespective of the changes made to its attributes or methods). The `@property` decorator allows a function to be accessed like an attribute.

How is try/except used in Python?

An exception is an error that occurs while the program is executing. When this error occurs, the program will stop and generate an exception which then gets handled in order to prevent the program from crashing.

The exceptions generated by a program are caught in the `try` block and handled in the `except` block.

- `Try`: Lets you test a block of code for errors.
- `Except`: Lets you handle the error.

Table of difference between Iterator vs Generators

Iterator	Generator
Class is used to implement an iterator	Function is used to implement a generator.
Local Variables aren't used here.	All the local variables before the yield function are stored.
Iterators are used mostly to iterate or convert other objects to an iterator using iter() function.	Generators are mostly used in loops to generate an iterator by returning all the values in the loop without affecting the iteration of the loop
Iterator uses iter() and next() functions	Generator uses yield keyword
Every iterator is not a generator	Every generator is an iterator