Python Input and Output

Python User Input from Keyboard – input() function :

- ✓ Python user input from the keyboard can be read using the input() built-in function.
- ✓ The input from the user is read as a string and can be assigned to a variable.
- ✓ After entering the value from the keyboard, we have to press the “Enter” button. Then the input() function reads the value entered by the user.
- ✓ The program halts indefinitely for the user input. There is no option to provide timeout value.
- ✓ If we enter EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), EOFError is raised and the program is terminated.
- ✓ The syntax of input() function is: **input(prompt)**
- ✓ The prompt string is printed on the console and the control is given to the user to enter the value. You should print some useful information to guide the user to enter the expected value.

What is the type of user entered value ?

The user entered value is always converted to a string and then assigned to the variable.

How to get an Integer as the User Input ?

There is no way to get an integer or any other type as the user input. However, we can use the built-in functions to convert the entered string to the integer.

Note : raw_input() function is used to accept user input in Python 2. x only, and is renamed to input() from Python 3. x version.

Output using the print() function :

To output your data to the screen, use the print() function. if your data is "Guido," you can put "Guido" inside the parentheses ( ) after print.

```
>>> print("Guido")
```

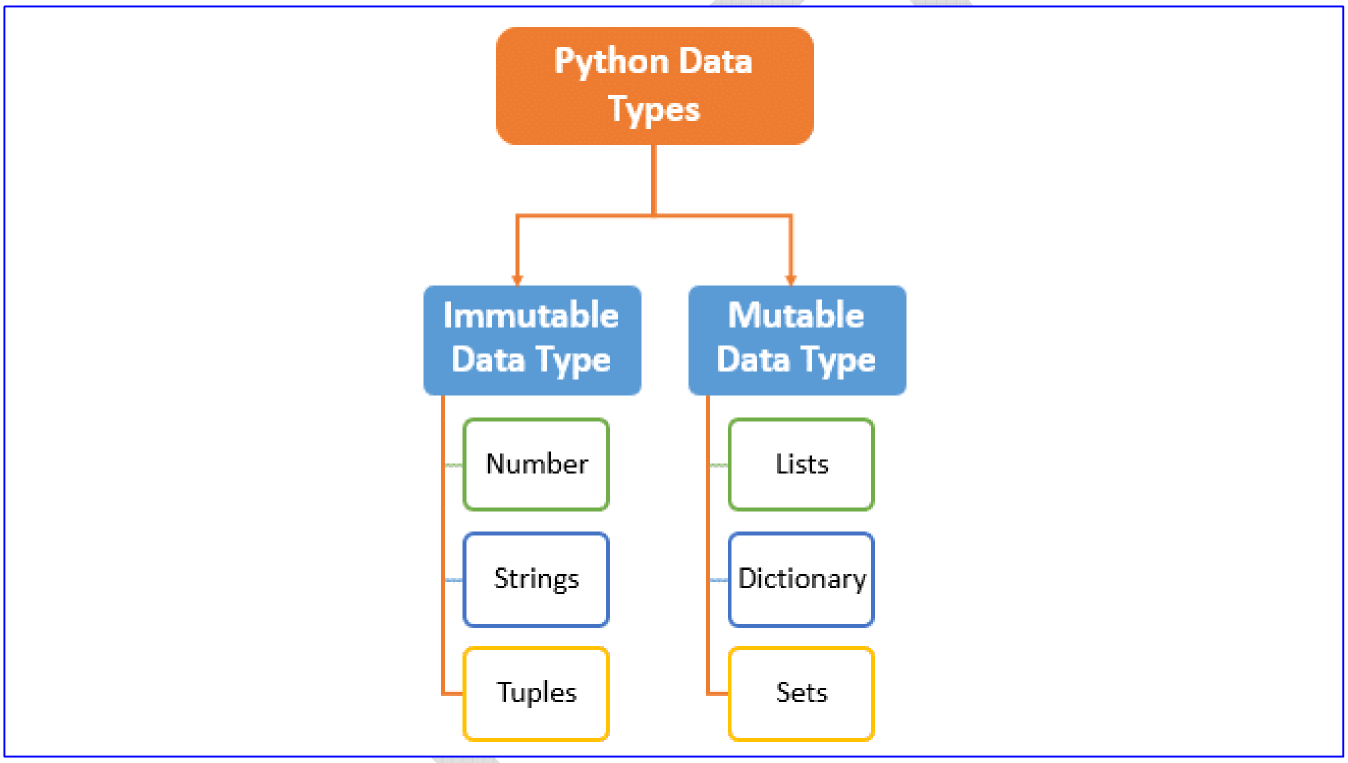
Standard Data Types in Python

As the name suggests, a data type is the classification of the type of values that can be assigned to variables

Python data types are categorized into two as follows :

Mutable Data Types : Data types in python where the value assigned to a variable can be changed

Immutable Data Types : Data types in python where the value assigned to a variable cannot be changed



Numbers :

✓ The number data type in Python is used to store numerical values.

✓ It is used to carry out the normal mathematical operations.

Strings :

✓ A string in Python is a sequence of characters.

✓ Python does not have a character data type, a single character is simply a string with a length of 1

- ✓ Strings in Python can be created using single quotes or double quotes or even triple quotes which are know delimiters
- ✓ They are used to carry out operations that perform positional ordering among items.

Lists :

- ✓ A list is a collection of values.
- ✓ Remember, it may contain different types of values.
- ✓ To define a list, you must put values separated with commas in square brackets

Tuples :

- ✓ A tuple is like a list.You declare it using parentheses instead.
- ✓ However, Python tuple is immutable. Once declared, you can't change its size or elements.

Sets :

- ✓ Sets in Python are a data type that can be considered as an unordered collection of data without any duplicate items.
- ✓ Define it using curly braces

Dictionaries :

- ✓ Dictionaries in Python can store multiple objects, but unlike lists, in dictionaries, the objects are stored by keys and not by positions
- ✓ A dictionary holds key-value pairs. Declare it in curly braces, with pairs separated by commas. Separate keys and values by a colon(:)

In-Built Functions Dealing With Data Types:

i. **type() :**

It takes one argument, and returns which class it belongs to.

ii. isinstance()

- ✓ It takes two arguments. The first is the construct(ex- a variable or a list), and the second is a class.
- ✓ It returns True or False based on whether the construct belongs to that class

iii. id():

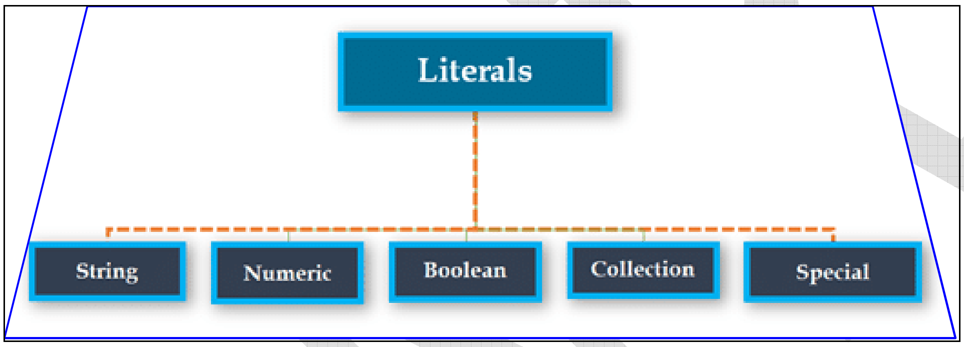
✓ **id()** is a built-in function in Python 3, which returns the identity of an object.

✓ The identity is a unique integer for that object during its lifetime.

✓ This is also the address of the object in memory

Python Literals

Literal is a raw data given in a variable or constant. In Python, there are various types of literals they are as follows:



1. **Numeric Literals**
2. **String literals**
3. **Boolean literals :**

✓ A Boolean literal can have any of the two values: **True or False.**

✓ True represents the value as 1 and False as 0.

4. **Special literals :**

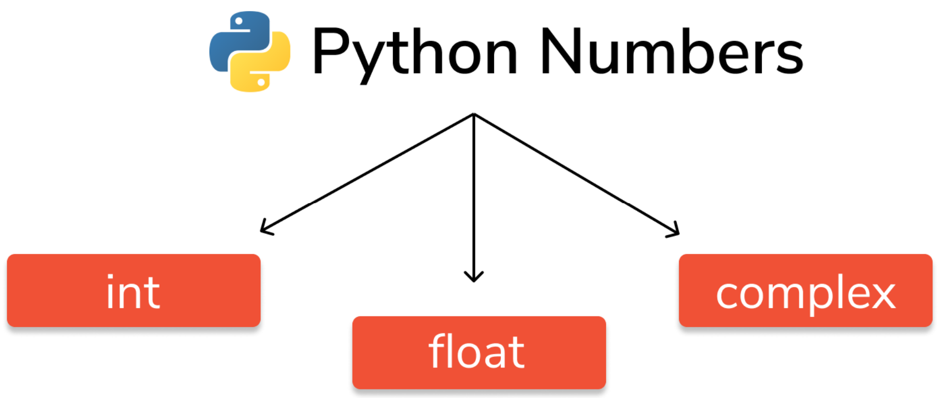
Python contains one special literal **i.e. None**. We use it to specify that the field has not been created.

5. **Literal Collections :**

There are four different literal collections List literals, Tuple literals, Dict literals, and Set literals.

Number Literals

- ✓ A number is an arithmetic entity that lets us measure something.
- ✓ Python allows us to store the integer, floating, and complex numbers and also lets us convert between them



a) Integers :

- ✓ Integers are whole numbers without any decimal values.
- ✓ They can be either positive or negative.
- ✓ In Python, integers are of unlimited size.An integer's size is limited only by the memory available in the system.
- ✓ Different number systems like Binary (base 2), Octal (base 8), Hexadecimal (base 16) are also considered to be of integer datatype.
- ✓ While working with numbers belonging to these number systems, you need to add these prefixes before the number:

For **Binary, '0b' or '0B'**

For **Octal,'0o' or '0O'**

For **Hexadecimal,'0x' or '0X'**

b) Floating Point Numbers

- ✓ Real numbers with a decimal point separating the integer and fractional parts are called floating-point numbers.
- ✓ They can be either positive or negative.
- ✓ A floating-point number is accurate only up to 15 decimal places, i.e fractional part can contain only 15 digits.
- ✓ Float values can also be scientific numbers with the notation 'E' or 'e' like this 23e2. 23e2 here is equivalent to 23 * 10^2

C)Python Complex Numbers :

- ✓ A complex number is a Python number type made of real and imaginary parts.
- ✓ It is represented as a+bj.
- ✓ However, you can't use the letter 'i', like you would do on paper.

Note: There is no 'long' integer in Python 3, int and long are unified now. This means int behave more like long

Type Conversion of Numbers in Python :

- ✓ Type conversion is the process of converting a data type into another data type.
- ✓ **Implicit type conversion** is performed by Python interpreter only.
- ✓ **Explicit type conversion** is performed by the user by explicitly using type conversion functions in the program code.
- ✓ Python, when performing implicit type casting, avoids the loss of data.
- ✓ We can convert one type of number into another. This is also known as coercion.
- ✓ Operations like addition, subtraction coerce integer to float implicitly (automatically), if one of the operands is float
- ✓ int(), float() can't convert a complex

A numeric object of one type can be converted in another type using the following functions:

Built-in Function	Description
int()	Returns the integer object from a float or a string containing digits.
float()	Returns a floating-point number object from a number or string containing digits with decimal point or scientific notation.
complex()	Returns a complex number with real and imaginary components.
hex()	Converts a decimal integer into a hexadecimal number with 0x prefix.
oct()	Converts a decimal integer in an octal representation with 0o prefix.
bin()	Converts a decimal integer in an octal representation with 0b prefix.

Inbuilt Functions Related to Numbers in Python

The below mentioned Python in-built functions will help make it easy to perform mathematical calculations.

You can include these inbuilt functions in your program after importing the required module by using this syntax.

import modulename #mandatory

x = 25

y = modulename.function () #syntax to call a function

print (y)

Modules Related To Numbers :

I. Random

II. Fraction

III. Decimal

IV. Math

Function	Purpose
abs (x)	Returns the absolute value of a function
exp (x)	Returns the value of ex i.e exponential of x
log10 (x)	Returns the logarithm of x to base 10 i.e log10 x
min ()	Returns the minimum among the given arguments
max ()	Returns the maximum among the given arguments
pow (x, y)	Returns the value of x to the power of y i.e x^y
round (x, number of digits)	Returns the rounded value of x
sqrt (x)	Return the square root of x

Constants :

A constant is a type of variable whose value cannot be changed. It is helpful to think of constants as containers that hold information which cannot be changed later generally used in modules

Eg : Pi

String literals

Creating a String :

Strings in Python can be created using single quotes or double quotes or even triple quotes

✧ Single Line String

✧ Multi line String

✧ Raw String

✧ **Unicode String :**

✓ Normal character strings can only draw from a set of 256 characters. There are many more than 256 characters in the world (eg, with notations like α = 5 Å) which means some things can't be written using a simple (byte) string. Instead, people have agreed on Unicode strings, which extends the character space to up to 65536 characters and even more.

✓ I won't cover Unicode at all because it's too complicated for an introductory class when all the data files we'll use are written in the limited set of ASCII charaters.

How to Use Quotes inside Python String ?

✓ Since we delimit strings using quotes, there are some things you need to take care of when using them inside a string.

✓ If you need to use double quotes inside a Python string, delimit the string with single quotes.

✓ If you need to use single quotes inside a string, delimit it with double quotes.

Accessing Python Strings :

We can access individual characters using **indexing** and a range of characters using **slicing**

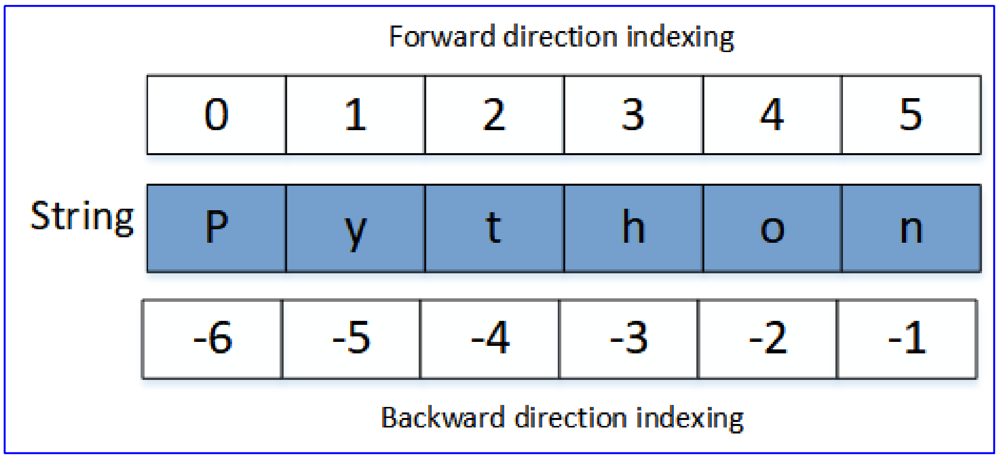
String Length :

The **len()** function can be used to find out total number of characters in the string.

Indexing :

✓ A character (also called element) of a string can be accessed with it's index number.

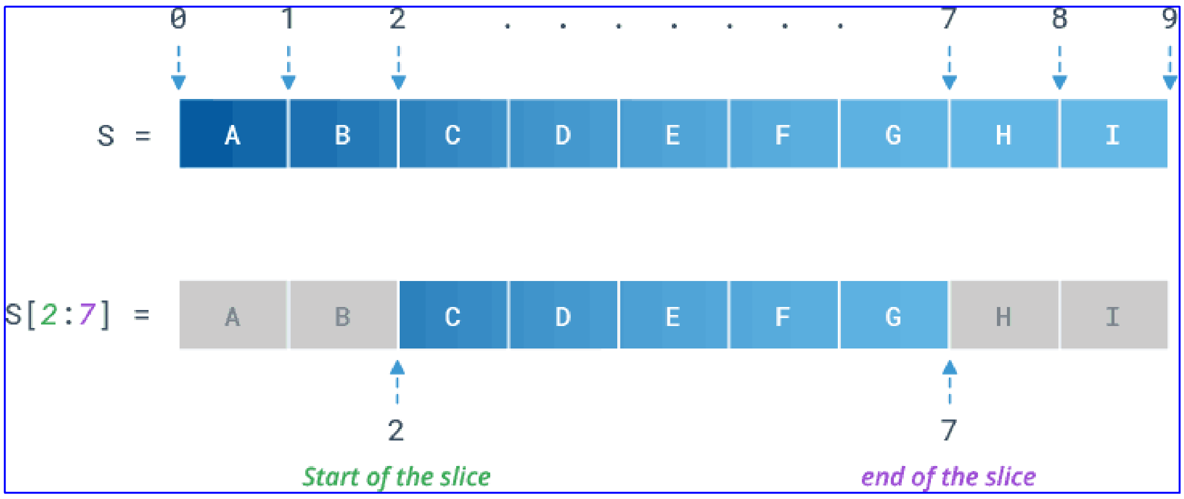
✓ In Python, index number starts with 0 in forward direction(Positive Indexing) and -1 in backward direction(Negetive Indexing)



- ✓ Square bracket **[ ]** notation used to access elements in a sequence through their indices
- ✓ While accessing an index out of the range will cause an **IndexError.**
- ✓ Only Integers are allowed to be passed as an index, float or other types will cause a **TypeError.**

Sliceing :

- ✓ You can return a range of characters by using the slice syntax.
- ✓ Specify the start index and the end index, separated by slicing operator **colon :** , to return a part of the string.
- ✓ If S is a string, the expression S [start : stop : step] returns the portion of the string from index start to index stop, at a step size step.
  - ✧ Start - Where to begin slicing
  - ✧ Stop - Where to stop slicing +1
  - ✧ Step - How much to increment between each index (Optional)



Deleting/Updating from a String :

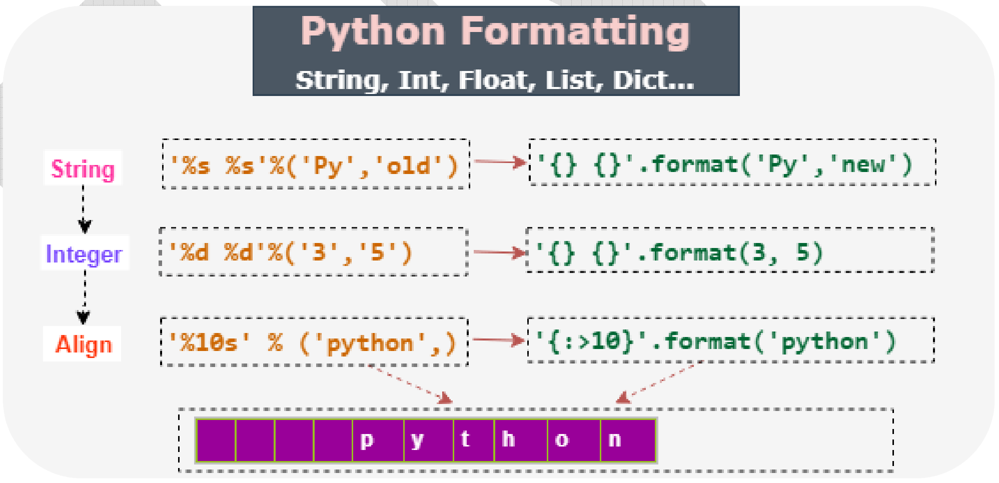
- ✓ In Python, Updation or deletion of characters from a String is not allowed.
- ✓ This will cause an error because item assignment or item deletion from a String is not supported
- ✓ Although deletion of entire String is possible with the use of a built-in **del** keyword.
- ✓ Only new strings can be reassigned to the same name.
- ✓ This is because Strings are immutable, hence elements of a String cannot be changed once it has been assigned.

String Formatting :

- ✓ String formatting methods in Python3,allows multiple substitutions and value formatting.
- ✓ This method lets us concatenate elements within a string through positional formatting.
- ✓ In Python a string of required formatting can be achieved by:

1) Using %

2) Using { }.format( )



```
'%s %s'%('Py','old')      '{} {}'.format('Py','new')
'%d %d'%('3','5')         '{} {}'.format(3, 5)
'%10s' % ('python',)      '{:>10}'.format('python')
```

- ✓ The formatting using %   :

  %d – integer

  %f – float

  %s – string

  %x – hexadecimal

  %o – octal

format() Method :

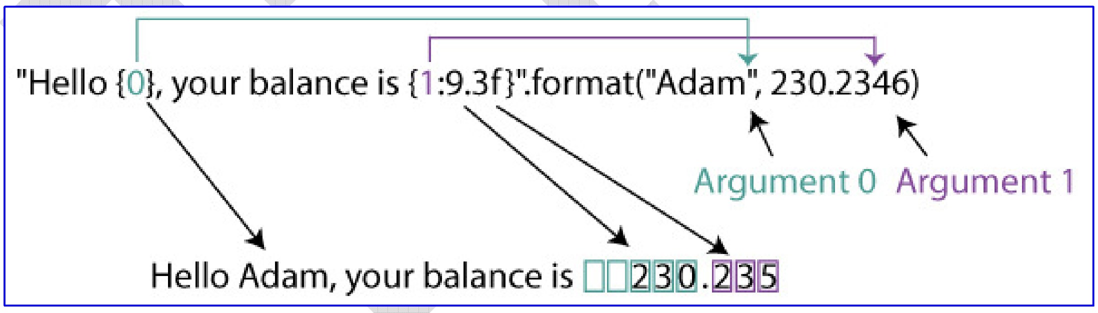
- ✓ The format() method that is available with the string object is very versatile and powerful in formatting strings.
- ✓ Format strings contain **curly braces {}** as placeholders or replacement fields which get replaced.
- ✓ We can use positional arguments or keyword arguments to specify the order.
- ✓ we can left-justify <, right-justify > or center ^ a string in the given space
- ✓ We can also format integers as binary, hexadecimal, etc. and floats can be rounded or displayed in the exponent format
- ✓ We can also Truncat strings with format

Syntax : { } .format(value)

Parameters :

(value) : Can be an integer, floating point numeric constant, string, characters or even variables.

Returntype : Returns a formatted string with the value passed as parameter in the placeholder position.


"Hello {0}, your balance is {1:9.3f}".format("Adam", 230.2346)
Argument 0
Argument 1
Hello Adam, your balance is   230.235

Concatenation and Replication of Strings :

- ✓ Joining of two or more strings into a single one is called concatenation.The + operator does this in Python.
- ✓ Repetition of String is called Replication . The * operator can be used to repeat the string for a given number of times.

String Conversions :

Function	Description
chr()	Converts an integer to a character
ord()	Converts a character to an integer
str()	Returns a string representation of an object

Python Escape Sequences

- ✓ To insert characters that are illegal in a string, use an escape character.
- ✓ An escape character is a backslash \ followed by the character you want to insert.
- ✓ An example of an illegal character is a double quote inside a string that is surrounded by double quotes

The below table contains a list of Python Escape sequence characters and relevant examples :

Escape Sequence	Description
\\	Prints Backslash
\`	Prints single-quote
\"	Pirnts double quote
\a	ASCII bell makes ringing the bell alert sounds (eg. xterm)
\b	ASCII backspace (BS) removes previous character
\f	ASCII formfeed (FF)
\n	ASCII linefeed (LF)
\N{name}	Prints a character from the Unicode database
\r	ASCII carriage return (CR). Moves all characters after (CR) to the beginning of the line while overriding same number of characters moved.
\t	horizontal tab (TAB). Prints TAB

\v	vertical tab (VT).
\uxxxx	Prints 16-bit hex value Unicode character
\Uxxxxxxxx	Prints 32-bit hex value Unicode character
\ooo	Prints character based on its octal value
\xhh	Prints character based on its hex value

Ignoring Escape Sequences - Raw String :

To **ignoring escape sequences in the string**, we make the string as **"raw string" by placing "r" or “R” before the string**. **"raw string"** prints as it assigned to the string

String Methods :

Python has a set of built-in methods that you can use on strings.

Note: All string methods returns new values. They do not change the original string.

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value

expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces

istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements of an iterable to the end of the string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list

rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning