Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.
'hello' is the same as "hello".
You can display a string literal with the print() function:

Example

```
print("Hello")
print('Hello')
```

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

```
a = "Hello"
print(a)
```

Multiline Strings

You can assign a multiline string to a variable by using three quotes:
Example

```
a = " “first one ”, “second one”, “third one””
```

Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.
However, Python does not have a character data type, a single character is simply a string with a length of 1.
Square brackets can be used to access elements of the string.

Example
Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, World!"
print(a[1])
```

String Length

To get the length of a string, use the len() function.

Example

The len() function returns the length of a string:

```
a = "Hello, World!"
print(len(a))
```

Check String

To check if a certain phrase or character is present in a string, we can use the keyword in.

Example

Check if "free" is present in the following text:

```
txt = "The best things in life are free!"
print("free" in txt)
```

Check if NOT

To check if a certain phrase or character is NOT present in a string, we can use the keyword not in.

Example

Check if "expensive" is NOT present in the following text:

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

Use it in an if statement:

Example

print only if "expensive" is NOT present:

```
txt = "The best things in life are free!"
if "expensive" not in txt:
  print("Yes, 'expensive' is NOT present.")
```

Python - Slicing Strings

Slicing

You can return a range of characters by using the slice syntax.
Specify the start index and the end index, separated by a colon, to return a part of the string.
Example
Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"
print(b[2:5])
```

Note: The first character has index 0.

Slice From the Start

By leaving out the start index, the range will start at the first character:
Example
Get the characters from the start to position 5 (not included):

```
b = "Hello, World!"
print(b[:5])
```

Slice To the End

By leaving out the *end* index, the range will go to the end:
Example
Get the characters from position 2, and all the way to the end:

```
b = "Hello, World!"
print(b[2:])
```

Negative Indexing

Use negative indexes to start the slice from the end of the string:
Example
Get the characters:
From: "o" in "World!" (position -5)
To, but not included: "d" in "World!" (position -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

Python - Modify Strings

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

Upper Case

Example

The `upper()` method returns the string in upper case:

```
a = "Hello, World!"
print(a.upper())
```

Lower Case

Example

The `lower()` method returns the string in lower case:

```
a = "Hello, World!"
print(a.lower())
```

Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

Example

The `strip()` method removes any whitespace from the beginning or the end:

```
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!”
```

Replace String

Example

The `replace()` method replaces a string with another string:

```
a = "Hello, World!"
print(a.replace("H", “J"))
```

Split String

The `split()` method returns a list where the text between the specified separator becomes the list items.

Example

The `split()` method splits the string into substrings if it finds instances of the separator:

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

String Format

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

example (this is a wrong example, don't do that)

```
age = 100
txt = "My name xxx, and I am " + age
print(txt)
```

But we can combine strings and numbers by using the `format()` method!
The `format()` method takes the passed arguments, formats them, and places them in the string where the placeholders `{}` are:
Example

Use the `format()` method to insert numbers into strings:
```
age = 100
txt = "My name is xxx, and I am {}"
print(txt.format(age))
```

The format() method takes unlimited number of arguments, and are placed into the respective placeholders:

```
quantity = 3
item_no = 567
price = 49.95
my_order = "I want {} pieces of item {} for {} dollars."
print(my_order.format(quantity, item_no, price))
```

Escape Character

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:

Example (Wrong example, don't do that)

You will get an error if you use double quotes inside a string that is surrounded by double quotes:

```
txt = "Hello today is " good " yeah."
```

To fix this problem, use the escape character `\"`:
Example

The escape character allows you to use double quotes when you normally would not be allowed:

```
txt = "Hello today is \"good\" yeah."
```

Escape Characters

Other escape characters used in Python:

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

Python Functions

A function is a block of organized, **reusable** code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

Defining a Function (Creating a Function)
You can define functions to provide the required functionality. In Python a function is defined using the **def** keyword. Here are simple rules to define a function in Python.

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Syntax

```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

By default, parameters have a positional behavior and you need to inform them in the same order that they were defined.

Example

```
def my_function():
  print(“April 2nd")
```

Practice : Please create 4 function with different function name and different print message

Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call a function:

```
def my_function():
  print(“April 2nd")

my_function()
```

Practice: please call the previous four function you created in the previous session

Arguments

Information can be passed into functions as arguments.
Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
The following example has a function with one argument (x). When the function is called, we pass along a value, which is used inside the function to print the full value:

```
def my_function(x):
  print(x + " Hello ")

my_function("March")
my_function("April")
my_function(“May")
```

Practice : Please create 4 function with argument and print different values

Parameters or Arguments?

The terms *parameter* and *argument* can be used for the same thing: information that are passed into a function.

From a function's perspective:
A parameter is the variable listed inside the parentheses in the function definition.
An argument is the value that is sent to the function when it is called.

Number of Arguments

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

Example :
This function expects 2 arguments, and gets 2 arguments:

```
def my_function(x, y):
  print(x + " " + y)

my_function("12", “21")
```

Practice : Please create 4 function with multiple arguments and print different values

Arbitrary Arguments, *args

if you do not know how many arguments that will be passed into your function, add a `*` before the parameter name in the function definition.
This way the function will receive a *tuple* of arguments, and can access the items accordingly:

Example
If the number of arguments is unknown, add a `*` before the parameter name:

```
def my_function(*item):
  print("The number of item is " + item[2])

my_function("A", "B", “C")
```

```
In class practice :

Please you Python programming method to modify the following string.


str_1 = "The Wall Street Journal is an American business-focused, English-
language international daily newspaper based in New York City, with
international editions also available in Chinese and Japanese. The Journal,
along with its Asian editions, is published six days a week by Dow Jones &
Company, a division of News Corp."

For instance, please capitalize the value of str_Sample
str_Sample = “Today is a good day”
str_Sample.upper()
'TODAY IS A GOOD DAY’

Question 1 :please create a function and name the function Capitalization and
define the function to capitalize str_1’s value.

Question 2 : please create a function and name the function LowerCased and
define the function to lowercase str_1’s value.
```

Question 3 : please create a function and name the function and name it whatever you want such as func_1, and define the function to return the first character of str_1

Question 4 : please create a function and name the function and name it whatever you want such as func_1, and define the function to return the last character of str_1

Question 5 : please create a function and name the function and name it whatever you want such as func_1, and define the function to return the first character of str_1

Question 6 : please create a function and name the function and name it whatever you want such as func_1, and define the function to return the 19th character of str_1

Question 7 : please create a function and name the function and name it whatever you want such as func_1, and define the function to replace all the character w into a

Question 8 : please create a function and name the function and name it whatever you want such as func_1, and define the function to replace all the character i into n

Question 9 : please create a function and name the function and name it whatever you want such as func_1, and convert the linear function f(x) = 3x + 12 in a function

Question 10 : please create four functions and define the following operation : addition, subtraction, multiplication, and division.