**Day 1 - What is Programming and Python?**

**What is Programming**

Programming is a way for us to tell computers what to do. Computer is a very dumb machine and it only does what we tell it to do. Hence we learn programming and tell computers to do what we are very slow at - computation. If I ask you to calculate 5+6, you will immediately say 11. How about 23453453 X 56456?

You will start searching for a calculator or jump to a new tab to calculate the same. This 100 days of code series will help you learn python from starting to the end. We will start from 0 and by the time we end this course, I promise you will be a Job ready Python developer!

**What is Python?**

* Python is a dynamically typed, general purpose programming language that supports an object-oriented programming approach as well as a functional programming approach.
* Python is an interpreted and a high-level programming language.
* It was created by Guido Van Rossum in 1989.

**Features of Python**

* Python is simple and easy to understand.
* It is Interpreted and platform-independent which makes debugging very easy.
* Python is an open-source programming language.
* Python provides very big library support. Some of the popular libraries include NumPy, Tensorflow, Selenium, OpenCV, etc.
* It is possible to integrate other programming languages within python.

**What is Python used for**

* Python is used in Data Visualization to create plots and graphical representations.
* Python helps in Data Analytics to analyze and understand raw data for insights and trends.
* It is used in AI and Machine Learning to simulate human behavior and to learn from past data without hard coding.
* It is used to create web applications.
* It can be used to handle databases.
* It is used in business and accounting to perform complex mathematical operations along with quantitative and qualitative analysis.

**Why Replit?**

* Replit is very easy to share tutorials and code.
* You can easily fork this repl and continue learning in your own style. Video, code as well as text tutorial on the same page which makes things easy!
* For fellow teachers out there, you create a .tutorial folder to create tutorials using replit.

**Day 3 - Modules and pip in Python!**

Module is like a code library which can be used to borrow code written by somebody else in our python program. There are two types of modules in python:

1. Built in Modules - These modules are ready to import and use and ships with the python interpreter. there is no need to install such modules explicitly.
2. External Modules - These modules are imported from a third party file or can be installed using a package manager like pip or conda. Since this code is written by someone else, we can install different versions of a same module with time.

**The pip command**

It can be used as a package manager [pip](https://pip.pypa.io/en/stable/) to install a python module. Lets install a module called pandas using the following command

pip install pandas

Copy

**Using a module in Python (Usage)**

We use the import syntax to import a module in Python. Here is an example code:

import pandas

# Read and work with a file named 'words.csv'

df = pandas.read\_csv('words.csv')

print(df) # This will display first few rows from the words.csv file

Copy

Similarly we can install other modules and look into their documentations for usage instructions.  
We will find ourselved doing this often in the later part of this course

**Day 5 - Comments, Escape sequence & Print in Python**

Welcome to Day 5 of 100DaysOfCode. Today we will talk about Comments, Escape Sequences and little bit more about print statement in Python. We will also throw some light on Escape Sequences

**Python Comments**

A comment is a part of the coding file that the programmer does not want to execute, rather the programmer uses it to either explain a block of code or to avoid the execution of a specific part of code while testing.

**Single-Line Comments:**

To write a comment just add a ‘#’ at the start of the line.

**Example 1**

#This is a 'Single-Line Comment'

print("This is a print statement.")

Copy

Output:

This is a print statement.

Copy

**Example 2**

print("Hello World !!!") #Printing Hello World

Copy

Output:

Hello World !!!

Copy

**Example 3:**

print("Python Program")

#print("Python Program")

Copy

**Output:**

Python Program

Copy

**Multi-Line Comments:**

To write multi-line comments you can use ‘#’ at each line or you can use the multiline string.

**Example 1:** The use of ‘#’.

#It will execute a block of code if a specified condition is true.

#If the condition is false then it will execute another block of code.

p = 7

if (p > 5):

print("p is greater than 5.")

else:

print("p is not greater than 5.")

Copy

Output:

p is greater than 5.

Copy

**Example 2:** The use of multiline string.

"""This is an if-else statement.

It will execute a block of code if a specified condition is true.

If the condition is false then it will execute another block of code."""

p = 7

if (p > 5):

print("p is greater than 5.")

else:

print("p is not greater than 5.")

Copy

**Output**

p is greater than 5.

Copy

**Escape Sequence Characters**

To insert characters that cannot be directly used in a string, we use an escape sequence character.

An escape sequence character is a backslash \ followed by the character you want to insert.

An example of a character that cannot be directly used in a string is a double quote inside a string that is surrounded by double quotes:

print("This doesnt "execute")

print("This will \" execute")

Copy

**More on Print statement**

The syntax of a print statement looks something like this:

print(object(s), sep=separator, end=end, file=file, flush=flush)

Copy

**Other Parameters of Print Statement**

1. object(s): Any object, and as many as you like. Will be converted to string before printed
2. sep='separator': Specify how to separate the objects, if there is more than one. Default is ' '
3. end='end': Specify what to print at the end. Default is '\n' (line feed)
4. file: An object with a write method. Default is sys.stdout

Parameters 2 to 4 are optional

**DAY 6 - Variables and Data Types**

**What is a variable?**

Variable is like a container that holds data. Very similar to how our container in kitchen holds sugar, salt etc Creating a variable is like creating a placeholder in memory and assigning it some value. In Python it’s as easy as writing:

a = 1

b = True

c = "Harry"

d = None

These are four variables of different data types.

**What is a Data Type?**

Data type specifies the type of value a variable hold.> This is required in programming to do various operations without causing an error.  
In python, we can print the type of any operator using type function:

a = 1

print(type(a))

b = "1"

print(type(b))

By default, python provides the following built-in data types:

**1. Numeric data: int, float, complex**

* int: 3, -8, 0
* float: 7.349, -9.0, 0.0000001
* complex: 6 + 2i

**2. Text data: str**

str: "Hello World!!!", "Python Programming"

=================================================================

**3. Boolean data:**

Boolean data consists of values True or False.

=================================================================

**4. Sequenced data: list, tuple**

**list:** A list is an ordered collection of data with elements separated by a comma and enclosed within square brackets. Lists are mutable and can be modified after creation.

**Example:**

list1 = [8, 2.3, [-4, 5], ["apple", "banana"]]

print(list1)

Output:

[8, 2.3, [-4, 5], ['apple', 'banana']]

**Tuple:** A tuple is an ordered collection of data with elements separated by a comma and enclosed within parentheses. Tuples are immutable and cannot be modified after creation.

**Example:**

tuple1 = (("parrot", "sparrow"), ("Lion", "Tiger"))

print(tuple1)

Output:

(('parrot', 'sparrow'), ('Lion', 'Tiger'))

=================================================================

**5. Mapped data: dict**

**dict:** A dictionary is an unordered collection of data containing a key:value pair. The key:value pairs are enclosed within curly brackets.

**Example:**

dict1 = {"name":"Sakshi", "age":20, "canVote":True}

print(dict1)

Output:

{'name': 'Sakshi', 'age': 20, 'canVote': True}

**DAY 7 - Operators**

Python has different types of operators for different operations. To create a calculator, we require arithmetic operators.

**Arithmetic operators**

| **Operator** | **Operator Name** | **Example** |
| --- | --- | --- |
| + | Addition | 15+7 |
| - | Subtraction | 15-7 |
| \* | Multiplication | 5\*7 |
| \*\* | Exponential | 5\*\*3 |
| / | Division | 5/3 |
| % | Modulus | 15%7 |
| // | Floor Division | 15//7 |

**Exercise**

n = 15

m = 7

ans1 = n+m

print("Addition of",n,"and",m,"is", ans1)

ans2 = n-m

print("Subtraction of",n,"and",m,"is", ans2)

ans3 = n\*m

print("Multiplication of",n,"and",m,"is", ans3)

ans4 = n/m

print("Division of",n,"and",m,"is", ans4)

ans5 = n%m

print("Modulus of",n,"and",m,"is", ans5)

ans6 = n//m

print("Floor Division of",n,"and",m,"is", ans6)

Copy

**Explaination**

Here 'n' and 'm' are two variables in which the integer value is being stored. Variables 'ans1' , 'ans2' ,'ans3', 'ans4','ans5' and 'ans6' contains the outputs corresponding to addition, subtraction,multiplication, division, modulus and floor division respectively.

**Exercise 1 - Create a Calculator**

Create a calculator capable of performing addition, subtraction, multiplication, and division operations on two numbers. Your program should format the output in a readable manner!

**DAY 9 - Typecasting in python**

The conversion of one data type into the other data type is known as type casting in python or type conversion in python.

Python supports a wide variety of functions or methods like: int(), float(), str(), ord(), hex(), oct(), tuple(), set(), list(), dict(), etc. for the type casting in python.

**Two Types of Typecasting:**

1. Explicit Conversion (Explicit type casting in python)
2. Implicit Conversion (Implicit type casting in python).

**Explicit typecasting:**

The conversion of one data type into another data type, done via developer or programmer's intervention or manually as per the requirement, is known as explicit type conversion.

It can be achieved with the help of Python’s built-in type conversion functions such as int(), float(), hex(), oct(), str(), etc .

**Example of explicit typecasting:**

string = "15"

number = 7

string\_number = int(string) #throws an error if the string is not a valid integer

sum= number + string\_number

print("The Sum of both the numbers is: ", sum)

Copy

**Output:**

The Sum of both the numbers is 22

Copy

**Implicit type casting:**

Data types in Python do not have the same level i.e. ordering of data types is not the same in Python. Some of the data types have higher-order, and some have lower order. While performing any operations on variables with different data types in Python, one of the variable's data types will be changed to the higher data type. According to the level, one data type is converted into other by the Python interpreter itself (automatically). This is called, implicit typecasting in python.

Python converts a smaller data type to a higher data type to prevent data loss.

**Example of implicit type casting:**

# Python automatically converts

# a to int

a = 7

print(type(a))

# Python automatically converts b to float

b = 3.0

print(type(b))

# Python automatically converts c to float as it is a float addition

c = a + b

print(c)

print(type(c))

Copy

**Ouput:**

<class 'int'>

<class 'float'>

10.0

<class 'float'>

**DAY 10 - Taking User Input in python**

In python, we can take user input directly by using input() function. This input function gives a return value as string/character hence we have to pass that into a variable.

**Syntax:**

variable=input()

But input function returns the value as string. Hence we have to typecast them whenever required to another datatype.

**Example:**

variable=int(input())

variable=float(input())

We can also display a text using input function. This will make input() function take user input and display a message as well

**Example:**

a=input("Enter the name: ")

print(a)

**Output:**

Enter the name: Harry

Harry

**DAY 11 - Strings**

**What are strings?**

In python, anything that you enclose between single or double quotation marks is considered a string. A string is essentially a sequence or array of textual data. Strings are used when working with Unicode characters.

**Example**

name = "Harry"

print("Hello, " + name)

Copy

**Output**

Hello, Harry

Copy

Note: It does not matter whether you enclose your strings in single or double quotes, the output remains the same.

Sometimes, the user might need to put quotation marks in between the strings. Example, consider the sentence: He said, “I want to eat an apple”.

How will you print this statement in python?: He said, "I want to eat an apple". We will definitely use single quotes for our convenience

print('He said, "I want to eat an apple".')

Copy

**Multiline Strings**

If our string has multiple lines, we can create them like this:

a = """Lorem ipsum dolor sit amet,

consectetur adipiscing elit,

sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua."""

print(a)

Copy

**Accessing Characters of a String**

In Python, string is like an array of characters. We can access parts of string by using its index which starts from 0.  
Square brackets can be used to access elements of the string.

print(name[0])

print(name[1])

Copy

**Looping through the string**

We can loop through strings using a for loop like this:

for character in name:

print(character)

Above code prints all the characters in the string name one by one!

**DAY 12 – Strings Slicing & Operations on String**

**Length of a String**

We can find the length of a string using len() function.

**Example:**

fruit = "Mango"

len1 = len(fruit)

print("Mango is a", len1, "letter word.")

Copy

**Output:**

Mango is a 5 letter word

Copy

**String as an array**

A string is essentially a sequence of characters also called an array. Thus we can access the elements of this array.

**Example:**

pie = "ApplePie"

print(pie[:5])

print(pie[6]) #returns character at specified index

Copy

**Output:**

Apple

i

Copy

Note: This method of specifying the start and end index to specify a part of a string is called slicing.

**Slicing Example:**

pie = "ApplePie"

print(pie[:5]) #Slicing from Start

print(pie[5:]) #Slicing till End

print(pie[2:6]) #Slicing in between

print(pie[-8:]) #Slicing using negative index

Copy

**Output:**

Apple

Pie

pleP

ApplePie

Copy

**Loop through a String:**

Strings are arrays and arrays are iterable. Thus we can loop through strings.

**Example:**

alphabets = "ABCDE"

for i in alphabets:

print(i)

Copy

**Output:**

A

B

C

D

E

**DAY 13 – Strings methods**

Python provides a set of built-in methods that we can use to alter and modify the strings.

**upper() :**

The upper() method converts a string to upper case.

**Example:**

str1 = "AbcDEfghIJ"

print(str1.upper())

Copy

**Output:**

ABCDEFGHIJ

Copy

**lower()**

The lower() method converts a string to lower case.

**Example:**

str1 = "AbcDEfghIJ"

print(str1.lower())

Copy

**Output:**

abcdefghij

Copy

**strip() :**

The strip() method removes any white spaces before and after the string.

**Example:**

str2 = " Silver Spoon "

print(str2.strip)

Copy

**Output:**

Silver Spoon

Copy

**rstrip() :**

the rstrip() removes any trailing characters. Example:

str3 = "Hello !!!"

print(str3.rstrip("!"))

Copy

**Output:**

Hello

Copy

**replace() :**

The replace() method replaces all occurences of a string with another string. Example:

str2 = "Silver Spoon"

print(str2.replace("Sp", "M"))

Copy

**Output:**

Silver Moon

Copy

**split() :**

The split() method splits the given string at the specified instance and returns the separated strings as list items.

**Example:**

str2 = "Silver Spoon"

print(str2.split(" ")) #Splits the string at the whitespace " ".

Copy

**Output:**

['Silver', 'Spoon']

Copy

We can use various other string methods to modify our strings.

**capitalize() :**

The capitalize() method turns only the first character of the string to uppercase and the rest other characters of the string are turned to lowercase. The string has no effect if the first character is already uppercase.

**Example:**

str1 = "hello"

capStr1 = str1.capitalize()

print(capStr1)

str2 = "hello WorlD"

capStr2 = str2.capitalize()

print(capStr2)

Copy

**Output:**

Hello

Hello world

Copy

**center() :**

The center() method aligns the string to the center as per the parameters given by the user.

**Example:**

str1 = "Welcome to the Console!!!"

print(str1.center(50))

Copy

**Output:**

Welcome to the Console!!!

Copy

We can also provide padding character. It will fill the rest of the fill characters provided by the user.

**Example:**

str1 = "Welcome to the Console!!!"

print(str1.center(50, "."))

Copy

**Output:**

............Welcome to the Console!!!.............

Copy

**count() :**

The count() method returns the number of times the given value has occurred within the given string.

**Example:**

str2 = "Abracadabra"

countStr = str2.count("a")

print(countStr)

Copy

**Output:**

4

Copy

**endswith() :**

The endswith() method checks if the string ends with a given value. If yes then return True, else return False.

**Example :**

str1 = "Welcome to the Console !!!"

print(str1.endswith("!!!"))

Copy

**Output:**

True

Copy

We can even also check for a value in-between the string by providing start and end index positions.

**Example:**

str1 = "Welcome to the Console !!!"

print(str1.endswith("to", 4, 10))

Copy

**Output:**

True

Copy

**find() :**

The find() method searches for the first occurrence of the given value and returns the index where it is present. If given value is absent from the string then return -1.

**Example:**

str1 = "He's name is Dan. He is an honest man."

print(str1.find("is"))

Copy

**Output:**

10

Copy

As we can see, this method is somewhat similar to the index() method. The major difference being that index() raises an exception if value is absent whereas find() does not.

**Example:**

str1 = "He's name is Dan. He is an honest man."

print(str1.find("Daniel"))

Copy

**Output:**

-1

Copy

**index() :**

The index() method searches for the first occurrence of the given value and returns the index where it is present. If given value is absent from the string then raise an exception.

**Example:**

str1 = "He's name is Dan. Dan is an honest man."

print(str1.index("Dan"))

Copy

**Output:**

13

Copy

As we can see, this method is somewhat similar to the find() method. The major difference being that index() raises an exception if value is absent whereas find() does not.

**Example:**

str1 = "He's name is Dan. Dan is an honest man."

print(str1.index("Daniel"))

Copy

**Output:**

ValueError: substring not found

Copy

**isalnum() :**

The isalnum() method returns True only if the entire string only consists of A-Z, a-z, 0-9. If any other characters or punctuations are present, then it returns False.

**Example 1:**

str1 = "WelcomeToTheConsole"

print(str1.isalnum())

Copy

Output:

True

Copy

**isalpha() :**

The isalnum() method returns True only if the entire string only consists of A-Z, a-z. If any other characters or punctuations or numbers(0-9) are present, then it returns False.

**Example :**

str1 = "Welcome"

print(str1.isalpha())

Copy

**Output:**

True

Copy

**islower() :**

The islower() method returns True if all the characters in the string are lower case, else it returns False.

**Example:**

str1 = "hello world"

print(str1.islower())

Copy

**Output:**

True

Copy

**isprintable() :**

The isprintable() method returns True if all the values within the given string are printable, if not, then return False.

**Example :**

str1 = "We wish you a Merry Christmas"

print(str1.isprintable())

Copy

**Output:**

True

Copy

**isspace() :**

The isspace() method returns True only and only if the string contains white spaces, else returns False.

**Example:**

str1 = " " #using Spacebar

print(str1.isspace())

str2 = " " #using Tab

print(str2.isspace())

Copy

**Output:**

True

True

Copy

**istitle() :**

The istitile() returns True only if the first letter of each word of the string is capitalized, else it returns False.

**Example:**

str1 = "World Health Organization"

print(str1.istitle())

Copy

**Output:**

True

Copy

**Example:**

str2 = "To kill a Mocking bird"

print(str2.istitle())

Copy

**Output:**

False

Copy

**isupper() :**

The isupper() method returns True if all the characters in the string are upper case, else it returns False.

**Example :**

str1 = "WORLD HEALTH ORGANIZATION"

print(str1.isupper())

Copy

**Output:**

True

Copy

**startswith() :**

The endswith() method checks if the string starts with a given value. If yes then return True, else return False.

**Example :**

str1 = "Python is a Interpreted Language"

print(str1.startswith("Python"))

Copy

**Output:**

True

Copy

**swapcase() :**

The swapcase() method changes the character casing of the string. Upper case are converted to lower case and lower case to upper case.

**Example:**

str1 = "Python is a Interpreted Language"

print(str1.swapcase())

Copy

**Output:**

pYTHON IS A iNTERPRETED lANGUAGE

Copy

**title() :**

The title() method capitalizes each letter of the word within the string.

**Example:**

str1 = "He's name is Dan. Dan is an honest man."

print(str1.title())

Copy

**Output:**

He'S Name Is Dan. Dan Is An Honest Man.

**DAY 14 – if-else Statements**

Sometimes the programmer needs to check the evaluation of certain expression(s), whether the expression(s) evaluate to True or False. If the expression evaluates to False, then the program execution follows a different path than it would have if the expression had evaluated to True.

Based on this, the conditional statements are further classified into following types:

if

if-else

if-else-elif

nested if-else-elif.

An if……else statement evaluates like this:

if the expression evaluates True:

Execute the block of code inside if statement. After execution return to the code out of the if……else block.\

if the expression evaluates False:

Execute the block of code inside else statement. After execution return to the code out of the if……else block.

Example:

applePrice = 210

budget = 200

if (applePrice <= budget):

print("Alexa, add 1 kg Apples to the cart.")

else:

print("Alexa, do not add Apples to the cart.")

Output:

Alexa, do not add Apples to the cart.

**DAY 16 – Match Case Statements**

To implement switch-case like characteristics very similar to if-else functionality, we use a match case in python. If you are coming from a C, C++ or Java like language, you must have heard of switch-case statements. If this is your first language, dont worry as I will tell you everything you need to know about match case statements in this video!

A match statement will compare a given variable’s value to different shapes, also referred to as the pattern. The main idea is to keep on comparing the variable with all the present patterns until it fits into one.

The match case consists of three main entities :

The match keyword

One or more case clauses

Expression for each case

The case clause consists of a pattern to be matched to the variable, a condition to be evaluated if the pattern matches, and a set of statements to be executed if the pattern matches.

Syntax:

match variable\_name:

case ‘pattern1’ : //statement1

case ‘pattern2’ : //statement2

…

case ‘pattern n’ : //statement n

Example:

x = 4

# x is the variable to match

match x:

# if x is 0

case 0:

print("x is zero")

# case with if-condition

case 4 if x % 2 == 0:

print("x % 2 == 0 and case is 4")

# Empty case with if-condition

case \_ if x < 10:

print("x is < 10")

# default case(will only be matched if the above cases were not matched)

# so it is basically just an else:

case \_:

print(x)

Output:

x % 2 == 0 and case is 4