Q- What is programming language?
A- Computer is designed to understand binary language that runs on Zeeros and Ones (10101). Computer can't understand normal languages like English, German, Spainish, Hindi etc. It is quite difficiult to write in a language that computer can understand. You would like to write a whole page just 10101010....1010101....? No ? We introduced programming languages for that crap. Programming language is nothing just a set of instruction that computer can understand and then produce desired output (or sometime not, if you are fogetting semi colon ha ha ha). Python, Java, C++ etc are examples of programming languages.

Q- What is python?
A- Python is interpreted language. What does that means? Interpreted means it get executed line by line. You don't need to write whole script to check if it is running good or not. (you write a wrong code, boom python will generate an error). It is also a high level, general programming language, meaning it can do basic things and complicated things too, depending how you take care of it. (like a good wife hehehe). It was developed in late 1980 by Guido Van Rossum.

LET'S GET STARTED, GUYS

Printing your first line - Python is easy, did I say "Easy"? Oh sorry, "Super Easy" I would prefer.

You can print anything in python by calling ***print()*** function-
```
>>>print("Here my python journey begins")
```
O/P = Here my python journey begins

See it was easy. We will discuss everything, just stay in touch and we will learn this together.

Variable- we all have kitchen and jars in there. Jar is something that hold something in it like sugar, salt or a chrochroach (if you love to be dirty {in bed is silent}). Variable are like same, **it holds something that you give to it.**

```
>>>message= "This is something I want to store"
>>>print(message)
```
O/P= This is something I want to store

Explaination- You stored a sentence in message by assigning it to the message. Here message is holding your sentence, it is a variable(jar, in your kitchen). I chose message, you can choose anything you love to choose.

For example-

```
>>>me= "Yeah, I am storing something in me, I am not pragnent too ha ha ha"
>>>print(me)
O/P= Yeah, I am storing something in me, I am not pregnent too ha ha ha
```

Things to notice-

1) Variable can contain- letters, numbers and underscores
*2) can't begin with number (**1_variable** is wrong but **variable_1** is not)*
3) spaces are not allowed (it is not galaxy, right)
4) avoid using python keywords (you'll come to know later)
*5) your variable name should be sensible, you are not the only one who is going to read your code. Keep it short and informative. **my_variable** is good but **this_is_my_variable_ and_I_love_it** is bullshit.*
6) use lowecase letters to name variable not uppercase one

DATA TYPES-

There are different kinds of data you can feed to python language. These types are called data types. My name is Akshansh Rawat (akshanshofficial@gmail.com +91 8384891269) and I will be teaching you about the python here. Lets get started with Data Types. First Data Type we will be discussing is "string"

Data Type – String

Strings are quite simple at first glance, but you can use it many different ways. It is a series or combination of characters (letters or words). Anything between single quotes (''), double quotes ("") and triple quotes(" " " """) is a string in python.

Examples- 'This is string in single quotes.'

"This is string in double quotes."

“ “ “ This is string in triple quotes.”””

Printing a string in python-

>>>print(‘This is first string’)

O/P= This is first string

>>>print(“This is another string, in double quotes this time”)

O/P= This is another string, in double quotes this time

>>>print(“ “ “This is triple quotes string”””)

O/P= This is triple quotes string

Slicing in String-

Slicing, umh hmm what does that mean? Slice of a mango??? You know it, cutting down mango in pieces. Yes! You got it right. Slicing is cutting string in different parts. Effectively speaking, slicing is accessing characters of a string at different positions.

#I am creating a variable and will store a string into it.

my_string= “Python”

taking full string, cutting nothing (don’t get offended here, I am testing your patience)

>>>print(my_string[:])

O/P= Python

>>>print(my_string[0:])

#this will print string from position 0 to the last

0=P, 1=y, 2=t, 3=h, 4=0, 5=n (0,1,2,3... are called index)

O/P= Python

>>>print(my_string[1:])

#will print from position(index) 1 to the end

O/P= ython

>>>print(my_sting[0:3])

#will print from position 0(P) to 2(t) (3-1=2 position)

O/P= Pyt

>>>print(my_string[2:4])

#will print from position 2 to 3 (4-1)

O/P= th

>>>print(my_string[-1:])

#will print from last (due to negative)

O/P= n

>>>print(my_string[-3,-1])

#will print from position 3 from back side to (-1-1=-2) 2 position from back

O/P= ho

Note that strings are immutable, that means string can’t be changed. Once string is created, you can’t change it. It is like a decision made by your wife/gf. (you can’t mute, immutable)

capitalize() method

This capitalizes the first letter of your string. Suppose we have a string -

strng= “this is String”

>>>print(strng.capitalize())

O/P= This is string. (Note that first letter is capitalized and other are lowercase, like “s” in string)

count() method

strng= “Apple”

>>>print(strng.count(p))

O/P= 2 (counts number of p in Apple)

endswith() method

strng= “Python”

>>>print(strng.endswith(“o”))

O/P= False (because it ends with ‘n’)

find() method

strng= “Python”

>>>print(strng.find(“t”))

O/P= 2 (because t is at position 2, starting from 0)

>>>print(strng.find(“z”))

O/P= -1 (returns -1 when it is not in string)

format() method

strng= “Python”

>>>print(strng.format())

O/P= (it format or delete the whole string)

index() method

name= “Akshansh Rawat”

>>>print(name.index(“Rawat”)

O/P=10 (returns the position of R, which is 10, count the space too)

swapcase() method

my_string= “this is PYTHON”

>>>print(my_string.swapcase())

O/P= THIS IS python (changes Upper case to lowercase and lower case to upper case)

strip() method

my_string= “ this is a string. “

>>>print(my_string.strip())

O/P= this is a string. (removes spaces from begining and from end)

replace() method

see_this= “this is python”

>>>print(see_this.replace(“python”, “java”))

O/P= this is java (replaced old value “python” to new one “java”)

title() method

new_string = “This is python language”

>>>print(new_string.title())

O/P= This Is Python Language (see first letter of each word is capitalized, it is a title now)

split() method

new_string= “ This is a python”

>>>print(new_string.split())

O/P =[‘This’, ‘is’, ‘a’, ‘python’] (split string from space)

>>>print(new_string.split(“is”))

O/P= [‘This’, ‘a’, ‘python’] (split from “is”, which is removed too)

Summary and try with your own: Best of the luck, Pythonistas

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