22-07-2024 Training Day -1

Python (INTRODUCTION)

Python: is an Open Source, High Level Language with simple syntax used in versatile domains.

Why Python is so popular:

1. Easy to Learn, is a very flexible programming, very user friendly

It is more English Like language

2. Used in Versatile Domains

3. Python is having a big library set. 2.35 Lakhs+ libraries. Each

Library may have 50 to 50k different ready made functions.

4. Python is an open source programming langauge.

5. Python is having a big community. Lakh+ Community Members including

multiple big MNC's

6. Python is a Platform Independent Langauge.

Portability: WORA: Write Once Run Anywhere

If we write our code on one OS and if the same code, without any change

can be executed on any other OS also then we say our langauge is : Portability

Actual Code: Programming Code, which is human understandable code

Compiled Code: is not human understandable

Platform Independent: CORA: Compile Once Run Anywhere

If we write our code on one OS and compile it,

after compilation, if the compiled file can be executed on any other

OS also then we say our langauge is Platform Independent.

Whether you got it?

7. Python supports Modular Approach: We can divide the project into

different modules and all the modules can be interfaced together

8. Python supports Dynamic Data Type Definition.

9. Python supports Dynamic Memory Allocation.

10. Python is an Interpreter Based language: In python, code

executes line by line.

Python is a dynamic programming language: Everything happens at run time in Python.
C Lang, C++, Java are Compiler based languages.

Python File Extension: .py

Python Compile File Extension: .pyc

Used in Versatile Domains:

Data Driven Domains: DA, BA, ML, DS, DL, AI, NLP, IP....

Other Domains: Web Development, Mobile Application Development,
Hardware Specific Applications, Gaming apps,
Cyber Security, Networking, CAD Designing....

Python supports Dynamic Data Type Definition: Data types in python are defined at run time. You need not to define data types of variables in advance.

```
a=5     #a int type
a=7.5  #a float type
a=True  #a bool type
a=2+3j  #a complex type
```

In C Lang, we need to define the data types of variables at compile time ie in advance before using the variables. So if we want to hold a whole number then we will define: int a; #C Syntax

```
a=5;        #C Syntax
a=7;        #C Syntax
```

But limitation in C : There are 2 Limitation:
1. is that these data types can store a fixed amount of data ie int will have some max values and min value
2. we can't save different types of data in a variable
There limitation are not there in python.

In C Lang int consumes 2 or 4 bytes depends upon the compiler.

1 Byte: 8 bits

1111111111111111: 16 bits: 2 bytes

Int: Signed int: 15 Bits : 111111111111111: Binary to Decimal:

2 power 15 -1= 32767

In C Lang if int is taking 2 bytes: C Lang int Range: -32768 to +32767

Variables are stored in RAM: Volatile Memory

Python supports Dynamic Memory Allocation: ie memory is allocated at run time.

```
a=5     #a will be stored at say 1000 location
a=100000000 #a will be stored at a different location 2000
a="CETPA"  #a will....3000
```

Dynamic Data Type Definition: We can store any type of data in a variable

Dynamic Memory Allocation: We can store any small or big value in a variable.

Data Types: In real world, being humans, we can differentiate different types of data like whole number, decimal point number, complex number, textual data....Similarly in programming also, different types of data are categorized.

Program

```
# import math
# r=math.pow(5,3)      #5 to the power 3
# print(r)
```

# #New Program

```
# import math
# r=math.log(100,10)       #100 means 10 to the power2
# print(r)
# #New Program
# print("CETPA")
# print("Vikas Kalra")
# print("Welcome")
# pr("Hello World")
# print("ABC")
# print("How are you?")
```

```
# #New Program
# a=5
# print(type(a))
# a=2.0
# print(type(a))
# a=True
# print(type(a))
# a=2+3j
# print(type(a))
```

Data Driven Track:

DA (BA) jupyter or sypder, ML (DS) google colab, DL (AI)..

Web Developer Track/Full Stack Developer

Django Framework, Web Designing (HTML, CSS, JSS,,),

Framework (React)

23-07-2024 Training Day -2

Applications of Python:

Data Driven Domains: Data Analytics, Business Analytics,
Data Science, Machine Learning, Deep Learning, Image Processing,
Natural Language Processing, Artificial Intelligence, Generative AI
...
Some Other Domains:
Hardware Specific Applications, Security Domains: Ethical Hacking,
Network Security, Cyber Security, Networking,
Full Stack Web Development, Mobile Application Development,
CAD tools for Electronics, Mechanical, Civil...

Data Driven :- Refers To The Practice Of Making Decisions, Strategies, Or Processes That Are Informed And Guided By Data Analysis And Interpretation, Rather Than Intuition, Assumptions, Or Purely Qualitative Factors. In A Data-Driven Approach, Objective Data Is Used As The Foundation To Improve Outcomes, Efficiency, And Accuracy

Full Stack Web Development Domain

Different between Data Analytics and Data Science

Data Analytics: Analysis of data is DA.

1. Processing a batch of information to get some useful conclusions
2. Making some meaningful conclusions from a raw data.
3. Arranging data in some meaningful or proper way
4. Data Collection in a defined

Finding important insights from the data.

1. Requirement Gathering: DARC: Data Analysis Requirement Specification
2. Collection of Data: Google Spreadsheets, Excel, CSV, TSV
3. Pre-processing of data: Bringing the data into some common units
4. Data Cleaning: Removing Outliers, Removing garbage data..
5. Data Visualize and Analyse
6. Data Reporting...

ML Algo on data and start making the predictions, forecasting, recommendation...

2014 Elections, who won the game: BJP

2019 Elections, who won the game: BJP

2024 Elections, who won the game: BJP

Why BJP won? Data Analyst: To analyse past data or to find if something happened then why it happened

“2029 Elections: Who will win the game? Data Scientist”

IDE'S USED IN PYTHON DEVELOPMENT:

IDE: Integrated Development Environment: Software

1. Pycharm: Python

2. Vscode

3. Jupyter Notebook

4. Spyder

5. Google Colab

Install: Python Install + Pycharm Community Edition Install

Tools used: Pycharm, vscode, ...

23-07-2024 Training Day -3

Install Python

1. Download Python:

- Go to the official Python website.
- Click the **Download Python** button to get the latest version for your operating system (Windows, macOS, or Linux).

2. Install Python:

- **Windows**:
  1. Run the downloaded `.exe` file.
  2. Check the box **"Add Python to PATH"** (very important).
  3. Select **Customize Installation** if needed, or choose **Install Now**.
- **macOS**:
  1. Open the `.pkg` file and follow the instructions.
  2. Python is pre-installed on macOS, but it's recommended to use the latest version.
- **Linux**:
  1. Use your package manager:

     ```
     bash
     Copy code
     sudo apt update
     sudo apt install python3
     ```

3. Verify Installation:

- Open a terminal or command prompt.
- Run:

  ```
  bash
  Copy code
  python --version
  ```

  Or for some systems:

  ```
  bash
  Copy code
  python3 --version
  ```

.

Install PyCharm

1. Download PyCharm:

- Go to the official PyCharm website.
- Choose the **Community Edition** (free) or **Professional Edition** (paid).

2. Install PyCharm:

- **Windows**:
  1. Run the downloaded `.exe` file.
  2. Follow the installation wizard.
  3. Check the box to **add PyCharm to PATH** or create a desktop shortcut.
- **macOS**:
  1. Open the `.dmg` file and drag the PyCharm icon to the Applications folder.
- **Linux**:
  1. Extract the downloaded `.tar.gz` file.
  2. Navigate to the `bin` folder in the extracted directory.
  3. Run:

     ```
     bash
     Copy code
     ./pycharm.sh
     ```

3. Configure PyCharm:

- Open PyCharm.
- Configure your Python interpreter:
  1. Go to **File** > **Settings** > **Project** > **Python Interpreter**.
  2. Click **Add Interpreter** and choose the installed Python version.

Types of Application:

1. Console Applications

Console applications are programs that run in a command-line interface (CLI) or terminal window. They primarily use text-based input and output and do not have a graphical user interface (GUI).

Use Cases:

- Automation scripts.
- System utilities (e.g., disk cleanup tools).

Examples:

- `ping` (networking tool).
- `gcc` (C compiler).
- Python command-line scripts.

2. Windows Applications

Windows applications are programs with a graphical user interface (GUI) designed to run on the Microsoft Windows operating system. They are user-friendly and visually interactive.

Use Cases:

- Desktop productivity tools (e.g., Microsoft Word, Excel).
- Multimedia applications (e.g., VLC Media Player).
- Gaming applications.
- Enterprise software (e.g., accounting or ERP systems).

Examples:

- Microsoft Paint.
- Notepad++.
- Adobe Photoshop.

User Interaction:

1 Input:

Users interact through a graphical interface using a combination of mouse clicks, keyboard inputs, or touch (on touchscreen devices).

Interactive elements include text boxes, buttons, sliders, drop-down menus, and more.

2 Output:

Feedback is displayed visually via pop-ups, labels, images, charts, or other GUI components.

Allows for richer data presentation, like tables, graphs, or multimedia elements

Data Types: To represent different types of values

Single element:

int: Whole Numbers 22 -563

float: Decimal Point: 2.0 ,3.62

complex: 2+3j , Real+Imaginary

Bool: Only 2 values, True, False

None Type: None

Multi element:

Iterators, str, list, tuple, dict, set, frozenset

```
# #New Program
# a=b        #NameError: name 'b' is not defined
# print(a)

# #New Program
# a=2+3j
# print(type(a))
```

STRING

str: Syntax: characters in a sequence included within single quotes, double quotes, triple quotes

str is a collection of homogeneous data types ie collection of characters.

Single Line String: Single, Double or Triple Quotes

Multi Line String: Only triple quotes

```
"""
# s='Welcome to CETPA'
# print(s)
# s="Welcome to CETPA"
# print(s)
# s='''Welcome to CETPA'''
# print(s)
# s="""Welcome to CETPA"""
# print(s)

# #New Program
# a="a"
# print(type(a))

# #New Program
# s='''Welcome to CETPA.
# CETPA is an award winning training company.
# CETPA is awarded by Chetan Bhagat, Shasho Tharoor and ...'''
# print(s)
# s="""Welcome to CETPA.
# CETPA is an award winning training company.
# CETPA is awarded by Chetan Bhagat, Shasho Tharoor and ..."""
# print(s)
```

"""

len function is created to find the length of iterators

len function won't work on single elements data type

INPUT FUNCTION:

Syntax:

input("Message for user")

var_name=input("Message for user").

Input function will hold the program till the time we press enter or pass the value and press enter.

…………………New Program………………………

```
# s=input("Enter Your Name:")
# print(s)
```

…………………New Program………………………

Addition of two input numbers

```
# a=input("Enter First No:")      #a=5
# b=input("Enter Second No:")     #b=7
# r=a+b                   #
# print(r)                #
```

Whenever we interact with the user in python then data is always travelled in string format.

input function always returns strings type of data in our program.

print function firstly call a __**str**____method on the arguments of the print function, and this str method always returns string data type and this string value returned from __**str**_____ method is printed on the screen.

```
# # print(25)
# L=[10,20,30]
# s=str(L)
# print(s)
# print(L)
# r=L.__str__()
# print(r)+ operator if applied on string then it concatenate the
strings ie join the strings.
+ operator works as a concatenation operator in case of
strings.
"""
```

…………………New Program……………………..

```
# a=5
# b=7
# s=a+b
# print(s)
# a="Ram"
# b="Shyam"
# s=a+b
# print(s)
```

…………………New Program……………………..

```
# a="5"
# b="7"
# r=a+b
# print(r)
```

…………………New Program……………………..

Addition of Two numbers: Incorrect Approach

```
# no1=input("Enter First No:")     #no1="5"
# no2=input("Enter Second No:")  #no2="7"
# res=no1+no2                    #res="57"
# print(res)
```

Whenever we take input from user then first and foremost thing is plan to consider the data type of input.

Type Casting

To convert one data type to another data type in programming then this concept is called type casting. also called type conversion, is the process of converting a variable from one data type to another. It is commonly used in programming to make operations compatible with variables of different types. There are two main types of type casting:.

1. Implicit Type Casting (Type Coercion):

- **Performed automatically by the compiler or interpreter.**
- Happens when a conversion is safe, such as from a smaller data type to a larger one (e.g., `int` to `float`).

Example in Python:

```
python
Copy code
x = 5    # Integer
y = 2.5  # Float
z = x + y  # x is implicitly converted to a float
print(z)  # Output: 7.5
.
```

2. Explicit Type Casting (Type Conversion):

- **Performed manually by the programmer.**
- Requires using casting functions or methods to convert one type to another.
- May result in loss of precision or data if not done carefully

Type Casting Syntax:

dest_var=dest_type(src_var)

…………………New Program………………

x=5

# y=float(x)

# print(y,type(y))

…………………New Program………………

x="5"

# y=float(x)

# print(y,type(y))

…………………New Program………………

x="5"

# y=int(x)

# print(y,type(y))

…………………New Program………………

# x=2.35

# y=int(x)

# print(y,type(y))

STRING DATA TYPE:

"5" vs 5 in memory

str and int

integers are directly converted to binary and saved

5: 101

ASCII Standard: American Standard Code for Information Interchange

ASCII code were initially developer of 7 bit length and later
it was designed with 8 bit length. ASCII is a standard to represent different standard keys of English Keyboard.

A:	65	a:	97
B:	66	b:	98
C:	67	c:	99
Z:	90	z:	122

0: 48

1: 49

2: 50

3: 51

4: 52

5: 53

9: 57

"5":

Through ASCII numbers we can represent only English Alphabets,
Numbers or some special symbols because ASCII code is of
only 8 bits.

To consider alphabets or special symbols of other languages
of the world, a new standard was created ie Unicode Standard.

16 Bits Combinations: 2 power 16: 65536

Unicode support 16 bits, 32 bits, 64 bits...

Now Python 3 support Unicode

Unicode is a universal character encoding standard designed to represent text in most of the world's writing systems. It assigns a unique code point to every character, regardless of the platform, program, or language, ensuring consistency in text representation.

A: ASCII Code 65, and Unicode 6

ord function returns the Unicode of the character passed

chr function returns the Character of the unicode passed

```
# #New Program
# print(chr(2346))
# print(ord("प"))

" "
```

24-07-2024

Training Day -4

Operators: are special symbols or keywords in programming that perform operations on variables and values. They are fundamental to building expressions and performing computations in a program.Like there are 7 Wonders of World, similarly
there are 7 Types of Operators in Python.

Types of Operators in Python.

1. Arithmatic Operators: 7 Operators

+ Addition
- Subtraction
* Multiplication
/ Division
** Exponent
% Remainder/Modulus
// Floor Division

Comments Comments in Python are used to add explanatory notes or documentation to the code. They are ignored by the Python interpreter and have no effect on the execution of the program. Comments improve code readability and help others (and your future self) understand the purpose of the code.

Single Line: Comment: # character
First Kasam: Always write at least 1 comment in the program ie
type the purpose of the program. Try to use as much comments
as possible.
Multi Line Comment: There is no direct option in python for
multiline comment. Now IDE's provides us some shortcut keys to
make multiple lines as comments but line by line comments
Pycharm Shortcut key: Cntrl + /

There is a way in Python which is not an official way of making
multiline comments but developers use this style frequently.
ie make any set of statments as strings and don't save it in a
variable then it can be seen as multiline comments

```
""""
# #New Program
# a=5
# b=2
# r=a/b
# print(r)
# a=5
```

b=2
# r=a//b
# print(r)
a=5
b=-2
# r=a//b
# print(r)

#New Program
a=5
b=3
# r=a**b   #a to the power b
# print(r)

#New Program
# "cetpa"

#New Program
a=5
b=3
# r=a%b   #a remainder b
# print(r)

""

Relational/Conditional Operators: 6 Types: Returns bool value ie True or False: Generally used to check conditions

==	Equals to Compares and check values are equal or not
!=	Not Equals to
>	Greater Than
<	Less Than
>=	Greater Than Equals to
<=	Less Than Equals to

.......................New Program……………..

x=5
# print(x>5)

.......................New Program……………..

# pwd=input("Enter the Password: ")   #pwd=vikas123
# print(pwd=="Vikas123")

.......................New Program……………..

# a,b=5,7
# r=a==b # print(r)

......................New Program…………….
```
# a,b=5,5
# r=a>=b
# print(r)
```

Logical Operators: Generally used to check multiple conditions.
and : When both the inputs are True, output is True
or : When at least one of the input is True, output will be True
not : Toggle the output: Input True then output False and vice versa

What are False values in python:
False
0
None
All empty values
Rest all are True values
```
s=""        #Empty String,
u=[]        #Empty List
```

If inputs are not boolean value:
and: if first input is False then output is first input else output will be second input
or: If first input is True then output is first input else output will be second input.
......................New Program…………….
```
# user="Vikas"
# pwd="Kalra"
# res=(user=="Vikas" and pwd == "Kalra")  #True and True
# print(res)
```
......................New Program ....# a,b=5,7
```
# r= a or b      #First input is True output will be first
# print(r)
```
......................New Program ....# a,b="",7
```
# r= a or b      #First input is False output will be second
# print(r)
```

Bitwise Operators: 6 in numbers: Works on bits
Bits are checked at same position level or index level.

& : and: If both bits are 1, output will be 1
| : or: If at least 1 of the bit is 1, output will be 1
~ : not: Toggle
^ : xor: If both bits are different then output will be 1

<< : shift left with 0 filling:
>> : shift right with upper bit filling
a << b means a will be shifted to left, b times
Shift Left: All the bits will be shifted to left, MSB (Most significant bit ie left most bit) will be discarded and LSB will be filled with 0.

Decimal
00
01
2
9
10
11
19
20

0000	: 0
0001	: 1
0010	: 2
0011	: 3
0100	: 4
0101	: 5

.......................New Program……………..

```
# a=5         #  0101
# b=3         #  0011
# r= a & b    #  0001     #r=1
# print(r)
```

.......................New Program……………..# a=5   # 0101

```
# b=3         #  0011
# r= a ^ b    #  0110     #6
# print(r)
```

All variables in python are of reference type ie when we assign one variable to another variable in python then address of RHS variable is assigned to LHS variable.

Assignment Operators: 1 (Equals to) + 7 (Arithmatic operators) +5 (Bitwise operator) + 1 (walrus)= 14

=	Equals to
+=	a+=b means a=a+b
-=	
*=	
/=	
**=	a**=b means a=a**b

%=
//=   a//=b means a=a//b
&=   a&=b means a=a & b
|=
^=   a ^= b means a = a ^ b
<<=
>>=   a >>= b means a = a >> b
:= Walrus operator: used to assign a value to a variable inside an expression

.......................New Program……………..

```
# a=5      #
# b=a
# print(id(a))
# print(id(b))
```

.......................New Program ....# a,b=5,7

```
# a+=b        #a=a+b
# print(a)
```

.......................New Program ....# a=5

```
# a+=1        #a=a+1
# print(a)
```

.......................New Program ....# a,b=5,3

```
# a**=b       #a=a**b
# print(a)
```

.......................New Program……………..

```
# print(a=5)      #TypeError: 'a' is an invalid keyword argument for print()

# #New Program
# print(a:=5)

# #New Program
# r=(a:=5+5)
# print(r)
```

25-07-2024 Training Day- 5

Binary: 0 and 1. False and True

```
# #New Program
# print(True and True)
# print(True and False)
# print(True or False)
```

Bitwise Operators: Work on bits: bit by bit

1: Convert your data into binary numbers

```
"""
# a=3         #0011
# b=2         #0010
# y = a & b  #0010  #2
# print(y)
```

………..New Program……..

```
# a,b,c,d=2,3,4,5
# y=a and b and c and d
# print(y)
```

………..New Program……..

```
# a,b,c,d=0,False,4,5
# y=a or b or c or d
# print(y)
```

Membership Operators: 2, returns True or False, bool

in

not in

check whether element or substring is part of main string or iterator check the element in an iterator, so they find elements only from iterators not from single elements data types

………..New Program……..

```
# s="CETPA InfoTech"
# r="T" in s
# print(r)
```

………..New Program……..

```
# r=5 in 456      #TypeError: argument of type 'int' is not iterable
# print(r)
```

………..New Program……..

```
# r="5" in "456"
# print(r)

"""
```

7. Identity Operators: 2 in numbers: Returns True or False:
Checks whether arguments have same identity ie address or not
is
is not
works like id function
identity operators checks whether arguments are reprsenting
the same identity/same entity or not
……….New Program……..
```
# a=5    #a address 1000
# b=a    #b address 1000
# print(a is b)
# print(a is not b)
```
……….New Program……..
```
# a=5     #a address is 1000
# b=a     #b address 1000
# print(id(a))
# print(id(b))
# print(a is b)  #Checks the addresses
# print(a==b)     #Checks the values
```

"""
Python Drawbacks:
1. Python is a very slow language.
2. Python consumes hell lot of memory.

100 dollar: 10k Man hours
300 Man hours: Cost drastically reduce
For companies these days, manpower cost matters much more than hardware infra cost.

Why Python is slow?
Because python support dynamic data type definition and dynamic memory allocation.

In Python: We need not to define the data type
……….New Program……..
```
# a=5
# print(a, type(a))
# a=2.5
# print(a, type(a))
# a="CETPA"
# print(a, type(a)
"""
```

If we want to define the data type intentionally before passing the value

```
"""
```

………..New Program……..

```
# a=int()
# print(a,type(a))
# a=float()
# print(a,type(a))
# a=list()
# print(a,type(a))
```

Python Supports Dynamic Data Type Definition: How memory is assigned to variables in Python a=5 :

How this statement will execute in the background

Step 1: It will be checked that how much memory is required to store 5 in RAM.

Step 2: Your python program will request the OS to allocate that much memory.

Step 3: OS will search in the RAM, where that much memory is available in RAM to store 5.

Step 4: OS will occupy the free memory available will allocate to python program.

Step 5: Now 5 will be stored at that location and a will start representing that location.

Say in statement, say a=5 is representing 1000 location.

Now Python supports dynamic memory allocation, once we assign a new value to same variable of same data type or another data type, all above steps will be repeated.

a=7

Python fundamental rule, whenever we assign any new value to a variables, it will be stored at a new location which actually means a new variable will be created

Benefit of Dynamic Memory Allocation:

1. We can pass any type of value to a variable
2. We can pass any big value for a variable in Python

………..New Program……..

```
# a=5     #Location 1000 location
# print(id(a))
# a=7     #Location 2000 location
# print(id(a))
a=5     #1000 location
```

b=7   #1010 location

a="Welcome to CETPA"

"""

Variable: Is a data storing element, whose value vaies in the program

a=5   #5 will be stored at 1000 location and will refer to 1000

a=7   #a will refer to 2000 location

When we run any application then in the background multiple small processes execute. In those processes of python, one process is of garbage collector. Whenever we assign a value to a variable then automatically a reference counter is assigned to the address associated with that variable. Reference counter represents the count ie how many variables are representing to a memory address.

a=5   #a address 1000, ref_counter of 1000 location will become 1

b=a   #b address 1000, ref_counter of 1000 location will become 2

#So here 1000 location is referred by 2 variables ie a and b that

#why the ref_counter value of 1000 location will become 2.

a=7   #a address 2000, ref_counter_2000=1

#ref_counter_1000=1 because now a is referring to 2000

#location and 1000 location is referred only by b ie 1 variable.

Reference counter tells that how much variables are referring to a particular memory location. Each memory location will have a different reference counter.

1. All variables in python are of ref type.If we assign one variable to another then address of RHS variable is assigned to LHS variable so variables are of reference type.
2. Python supports dynamic memory allocation, means whenever we assign a new value to a variable then it will be stored at a new location.
3. Whenever any location is not referred by any variable in the program then that location is automatically made free by the garbage collector.

"""