👥

HARSHITH’S DATA SCIENCE BOOK

Python:

Python is a high-level, object-oriented, interpreted language that can be applied to various fields such as web development, machine learning, data analysis, etc.

Why Python:

- Python is known for its straightforward syntax and ease of use, making it a popular choice for beginners.
- With over 127,000 libraries available, it offers a vast range of possibilities for developers to work with.
- Python is widely used across various domains such as web development, machine learning, artificial intelligence, and more.
- Python is a Dynamic Language.

Variables:

Variables are named places in memory where values can be stored for later use.

```
a=10
b="Harshith"
```

In the above statement "a = 10 and b = "Harshith", "a" and "b" are examples of variables that are assigned the values 10 and "Harshith", respectively.

Rules of Declaring a Variable:

There are certain rules while declaring variables such as:

- Variable names shouldn’t start with a Number.

- There shouldn’t be gaps while declaring a variable.
- Python, variables are case-sensitive, meaning that "Name" and "name" are considered two different variables.

```
3name=10   #This is not a correct way
Id no=1739 #This is not a correct way

#Instead we can use the below methods

name3=10   #This is a correct way
Id_no=1739 #This is a correct way
```

Data Types:

Data types are the classification of data based on the type of value they hold. Some common data types in programming include:

1. Integer: a whole number
2. Float: a number with a decimal point
3. String: a sequence of characters (e.g. "Hello World")
4. Boolean: a data type that can have only two values, either True or False
5. Complex: which are in the format of a+bj
6. List: an ordered collection of values
7. Tuple: an ordered, immutable collection of values
8. Dictionary: an unordered collection of key-value pairs
9. set: an unordered collection of values

```
#Examples of declaring various basic Data types

#Integers
a=10
b=11339

#Float
a=1.22
nass=0.40
g=1.00

#Strings
a="ahhssh"
b="asdfghjkl"

#Boolean
a=True
b=False
```

```
#complex
a=2+3j
b=44-9j
```

Data Type Check:

In python, we can check the Data Type of the data using the inbuilt function type()

```
ip: type(123)
op: int

ip: type("ASDFFF")
op: str

ip: type(1.00)
op: float

ip: type(True)
op: boolean

ip: type(1+2j)
op: complex
```

Type Casting:

Type casting is the technique by which we can change the data type of the data from one data type to another.

For eg: if a=123 and it is an integer, we can easily change its Data Type by using float(a) or str(a). Here float value will become 123.00 and if it's changed to string it will become "123" which is not a number but a character in the form of a number.

```
ip: a=123
    type(a)
op: int

ip: a=str(a)
    type(a)
op: str

# And we can do this as vice versa to change strings(in the form of numbers) to float and int.
# And also float to int and str.

ip: int(123.45)
op: 123
```

Print Function:

print() is a built-in function in Python used to print desired output.

```
ip: print("asdfghjkl")
op: asdfghjkl

ip: a=12
    print(a)
op: 12

ip: print(1+2)
op: 3
```

Functionalities of Print() function:

- **\n** can be used to print text in separate lines.
- **sep** can be used to separate two output terms.

```
ip: print("asdfghjkl
    asdghhh)
op: error

ip: print("asdfghjkl\nasdghhh")
op: asdfghjkl
    asdfhhh

ip: a=12
    b="pen"
    print(a,b)
op: 12,pen

ip: print(a,b,sep='-')
op: 12-pen
```

Types of printing variables along with a text using Print():

- Using string concatenation (Adding the variable with text)
- Using comma
- fstring
- Formatting
- Placeholders using Format Function

Using String concatenation:

Just use the symbol ‘+’ between two strings to add them.

```
ip: # Adding two variables in Print using concatenation
    # As this method adds only strings, we first need to typecast every variable as a string
    a=10
    b="harshith"
    print(str(a)+' '+b)
op: 10 harshith
```

Using comma:

Just use the comma symbol between two variables to add them(variables can be of any Data Types)

```
ip: # Add them using a comma
    print(a,b)
op: 10 harshith
```

fstring:

Add variables in between text in angular braces {} and use f at the point before the coats(”,’) of print statement.

```
ip: # let's think name=Harshith, age=21 and degree =Btech
    # print My name is Harshith and My age is 21 and I had done my BTech degree
    name="Harshith"
    age=21
    degree="Btech"
    print(f"My name is {name} and age is {age} and I had done {degree}")
op: My name is Harshith and age is 21 and I had done Btech
```

Format Function:

Same as fstring but there is no need to give f and also no need to give the variable name in angular {} braces. Instead, we can use .format() function and pass the variables or parameters in it.

```
ip: print("My name is {} and age is {} and I had done my {}".format(name,age,degree))
op: My name is Harshith and age is 21 and I had done Btech

# Note: The variables in the format function should be in the same order as the places
#       you want them to fit in angular braces.
```

Placeholders using Format Function:

we can use placeholders in angular braces so that there will be no need for particular order to be followed in format function.

```
ip: print("My name is {n} and age is {a} and I had done my {d}".format(a=age,d=degree
                                                                    ,n=name))
op: My name is Harshith and age is 21 and I had done my Btech
```

Operators in Python:

Python has several types of operators:

1. Arithmetic Operators: +, -, *, /, %, **, //
2. Comparison Operators: ==, !=, >, <, >=, <=
3. Assignment Operators: =, +=, -=, *=, /=, %=, **=, //=
4. Logical Operators: and, or, not
5. Membership Operators: in, not in
6. Identity Operators: is, is not

Control Statements in Python:

Python has several control statements like

If-Else: Executes a block of code based on whether a condition is true or false.

```
ip: a=22
    # using "IF" the code will check if the condition is greater than 18 or not.
    if a>18:
      # if the statement is true it will execute the below line of code
      print("You are major")
    else:
      # if the "IF" statement is false, then the Else code will be executed.
      print("You are minor")
op: You are major
```

Elif: Will be used along with If-Else where we have to check more than 2 conditions.

```
ip: a=77
    # using "IF" the code will check if the condition is greater than 18 or not.
    if a>18 and a<60:
      # if the statement is true it will execute the below line of code
      print("You are major")
    elif a>60:
      print("You are old")
    else:
```

```
      print("You are minor")
op: You are old
```

For Loop: Executes a block of code repeatedly for each item in a sequence (list, tuple, etc.).

For eg: if name=”Harshith” and I want to traverse each of the elements of the value of the variable name, Then I have to travel through H A R S H I T H one by one. we can do this by using For loop. so we can define for-loop as the one which can traverse through all the elements of a given value be it string or list or dictionary or etc.

```
ip: name="HARSHITH"
    # Here i is the temporary variable to which we are assigning each element of the name
                                                          # "HARSHITH" one after another.
    for i in name:
      #Now we are going to print each value of i
      print(i)
op: H
    A
    R
    S
    H
    I
    T
    H
```

We can also use if-else statements in the for loop as below

```
#Program to replace all the letters "H" to "*" using if statement in for loop.
ip: name="HARSHITH"
    for i in name:
      # I will add an if statement here.
      if i=="H":
        print(*)
      else:
        print(i)
op: *
    A
    R
    S
    *
    I
    T
    *
```

Instead of Printing each character in a different line, we can print them on the same line using the end property in the print statement.

```
#In this code I used end="" empty string to be placed between each printing Element.
ip: name="HARSHITH"
    for i in name:
      if i=="H":
        print(*,end="")
      else:
        print(i,end="")
op: *ARS*IT*
```

Break: Terminates a loop prematurely.

we can use the Break statement to end a loop when a certain condition occurs.

```
#In this code I will stop executing code if there is letter "R" occurs.
ip: name="HARSHITH"
    for i in name:
      if i=="R":
        break
      print(i,end="")
op: HA
```

Continue: Skips the current iteration of a loop and moves to the next one.

we can use the Continue to skip the loop for one time to execute when a certain condition occurs.

```
#In this code I will skip executing code if there is letter "H" occurs.
ip: name="HARSHITH"
    for i in name:
      if i=="H":
        continue
      print(i,end="")
op: ARSIT
```

Pass: Does nothing, used as a placeholder for future code.

```
#In this code pass Does nothing, used as a placeholder for future code
ip: name="HARSHITH"
    for i in name:
      if i=="H":
        pass
      print(i,end="")
op: HARSHITH
```

While Loop: Executes a block of code repeatedly as long as a given condition is true.

For Eg: If there is a Government employee who aged 25. if we want to write a code to notify him when it's time for him to Retire we can do it using While loop by printing that it's

time for his retirement once he reaches 62 by using While Loop.

```
ip: # While loop
    age=25
    experience=0
    # when age becomes 62 this while loop will stop
    # ! is not symbol. != this refer not equal to 62 in the below loop.
    while age!=62:
      # Here we are increasing the age by 1 each time while the loop is running.
      age=age+1
      #Let's increase experience every year
      experience=expericene+1
    else:
      print("it's time to Retire! You had now {} years of experience.format(experience))
op: it's time to retire you had now 37 years of experience.
```