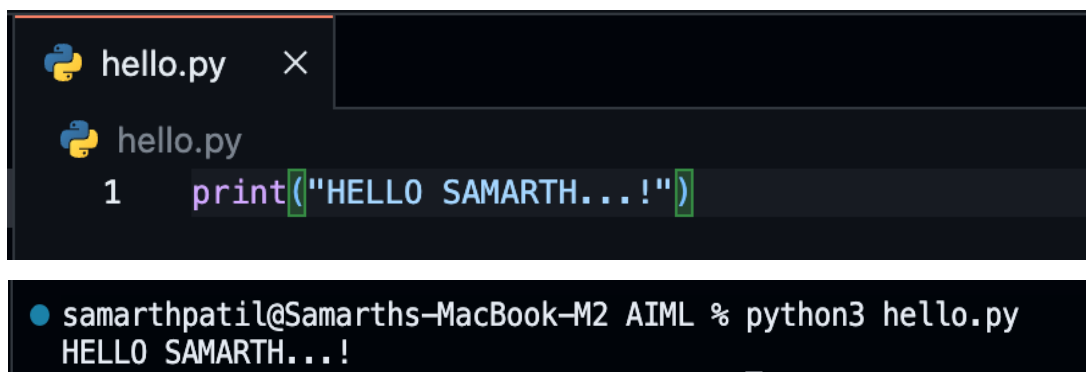


# DAY – 02

In this day – 2 module you will get understanding of python syntax, variables, comments, etc. most important Data Types & Operators section. Keeping this aside focus on concepts. After several days of practice, you will get clear understanding of everything.

Syntax: - python syntax does not follow any traditional format or simply it does not need any pre req defined structure.

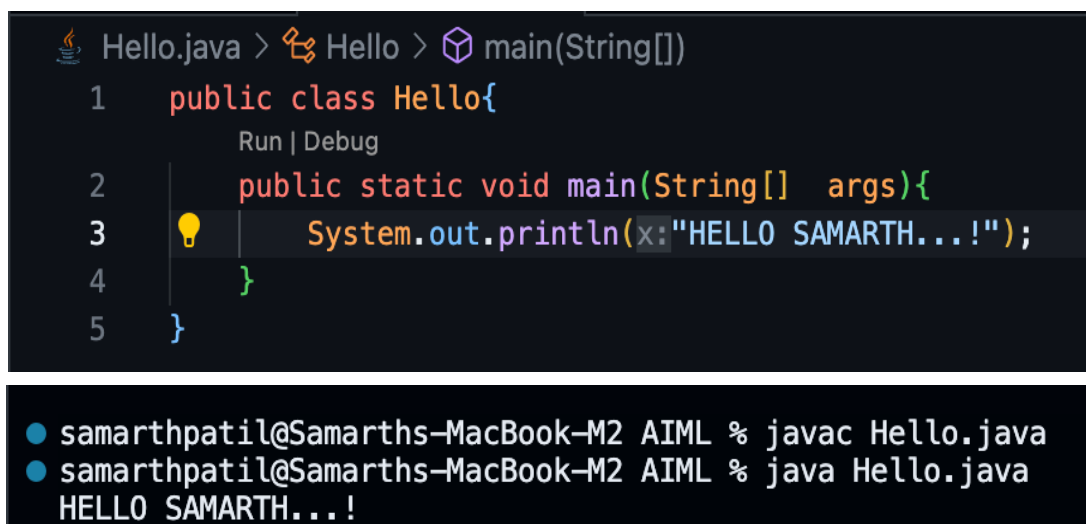


```
hello.py x
hello.py
1 print("HELLO SAMARTH...!")

● samarthpatil@Samarths-MacBook-M2 AIML % python3 hello.py
HELLO SAMARTH...!
```

As you can see here we don't have to put any format while printing the O/P – HELLO SAMARTH.

HERE I WILL SHOW WHY I AM SAYING FORMAT OR ANY STRUCTURE WHICH PYTHON DOES NOT NEED AND MAKES IT EASY TO LEARN.



```
Hello.java > Hello > main(String[])
1 public class Hello{
    Run | Debug
2     public static void main(String[] args){
3         System.out.println(x:"HELLO SAMARTH...!");
4     }
5 }
```

```
● samarthpatil@Samarths-MacBook-M2 AIML % javac Hello.java
● samarthpatil@Samarths-MacBook-M2 AIML % java Hello.java
HELLO SAMARTH...!
```

Now you can clearly understand the difference. Where we have to put a format to print just HELLO SAMARTH.

That's why python is easy to learn and grasp.

Don't worry about the understanding. No one is perfect or that much proficient in the beginning. Just go through the process and practice hard.

Now let's focus or start the main course. We have now clear understanding of python, what python makes easy, in which fields python is used. Let's dive deeper.

---

1. Python data types: - it has several data types

a. Numeric – int, float, complex.

```
p = 5 # int
q = 3.14 # float
r = 3 + 4j # complex
```

b. String – sequence of character

```
text = "SAMARTH"
print(text)
```

c. List – ordered | mutable

```
list = [31, 22 , "SAMARTH"]
print(list)
```

d. Tuple – ordered | mutable

```
tuple = (31, 22 , "SAMARTH")
print(tuple)
```

e. Set – unordered | unique collection

```
set = {1, 2, 3, 3, 4, 5, 6}
print(set)
```

f. Dictionaries – key-value pair

```
dict = {"name" : "SAMARTH" , "number" : 8766794922}
print(dict)
```

## 2. Type conversion: -

```
a = 5  
b = str(a)  
c = float(b)
```

Here the data type is changed to string first and to float second.

## 3. Operators: -

### a. Arithmetic

| OPERATOR | DESCRIPTION    | EXAMPLE (a=5 : b=5) | RESULT |
|----------|----------------|---------------------|--------|
| +        | Addition       | a+b                 | 10     |
| -        | Subtraction    | a-b                 | 0      |
| *        | Multiplication | a*b                 | 25     |
| /        | Division       | a/b                 | 1      |
| //       | Floor division | a//b                | 1      |
| %        | Modulus        | a%b                 | 1      |
| **       | Exponentiation | a**b                | 3125   |

### b. Logical

| operator | description                                     | Example x=T   y=F | result |
|----------|-------------------------------------------------|-------------------|--------|
| and      | Returns True if both conditions are True.       | x and y           | False  |
| or       | Returns True if at least one condition is True. | x or y            | True   |
| not      | Negates the value.                              | Not x             | false  |

### c. Comparison

| operator | Description              | Example a = 5 : b = 1 | result |
|----------|--------------------------|-----------------------|--------|
| ==       | Equal to                 | a==b                  | False  |
| !=       | Not equal to             | a!=b                  | True   |
| >        | Greater than             | a>b                   | True   |
| <        | Less than                | a<b                   | False  |
| >=       | Greater than or equal to | a>=b                  | True   |
| <=       | Less than or equal to    | a<=b                  | False  |