Introduction to Python

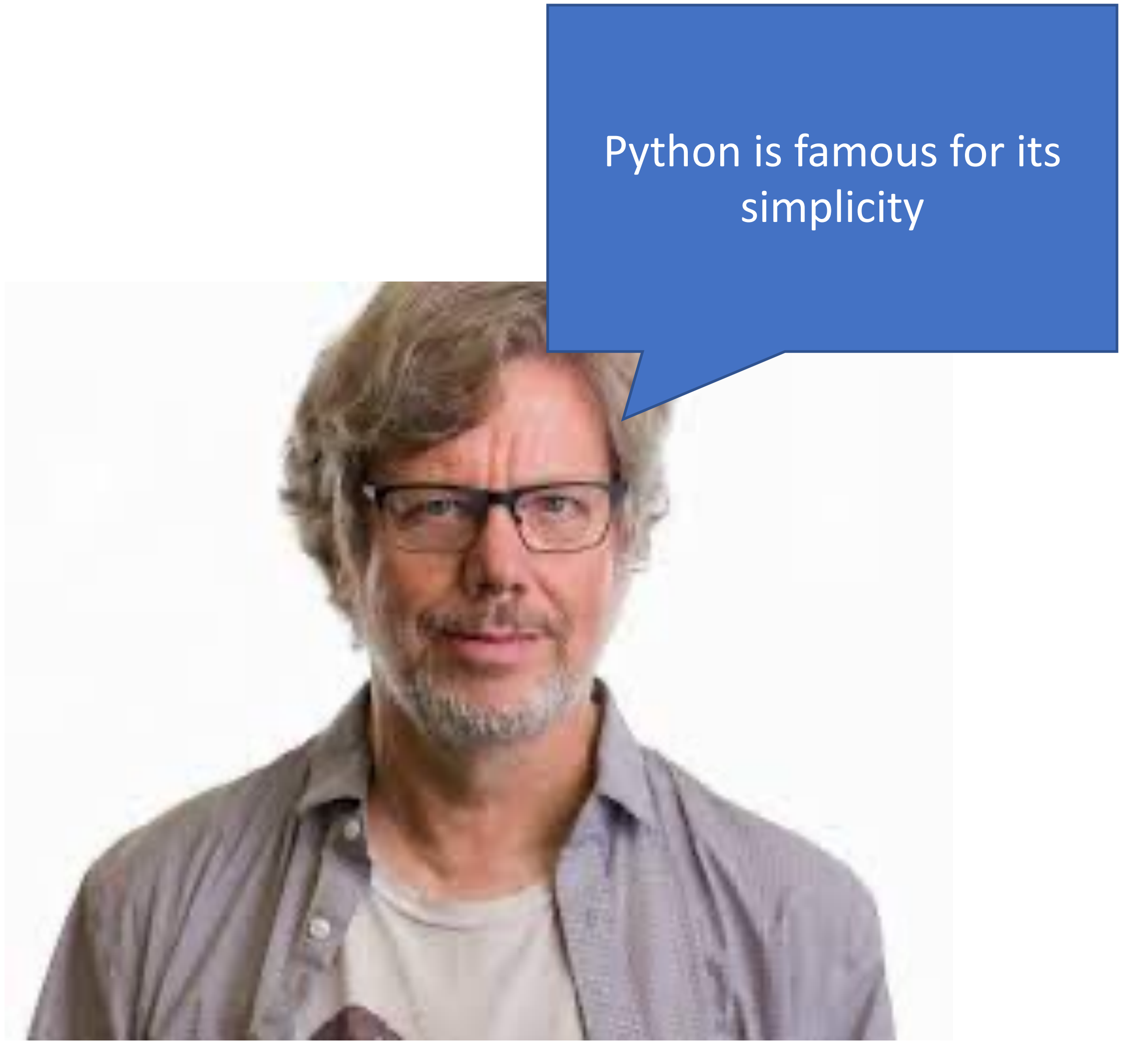
by

RAGE Uday Kiran

Hai!

I am Guido Van Rossum

Creator Python



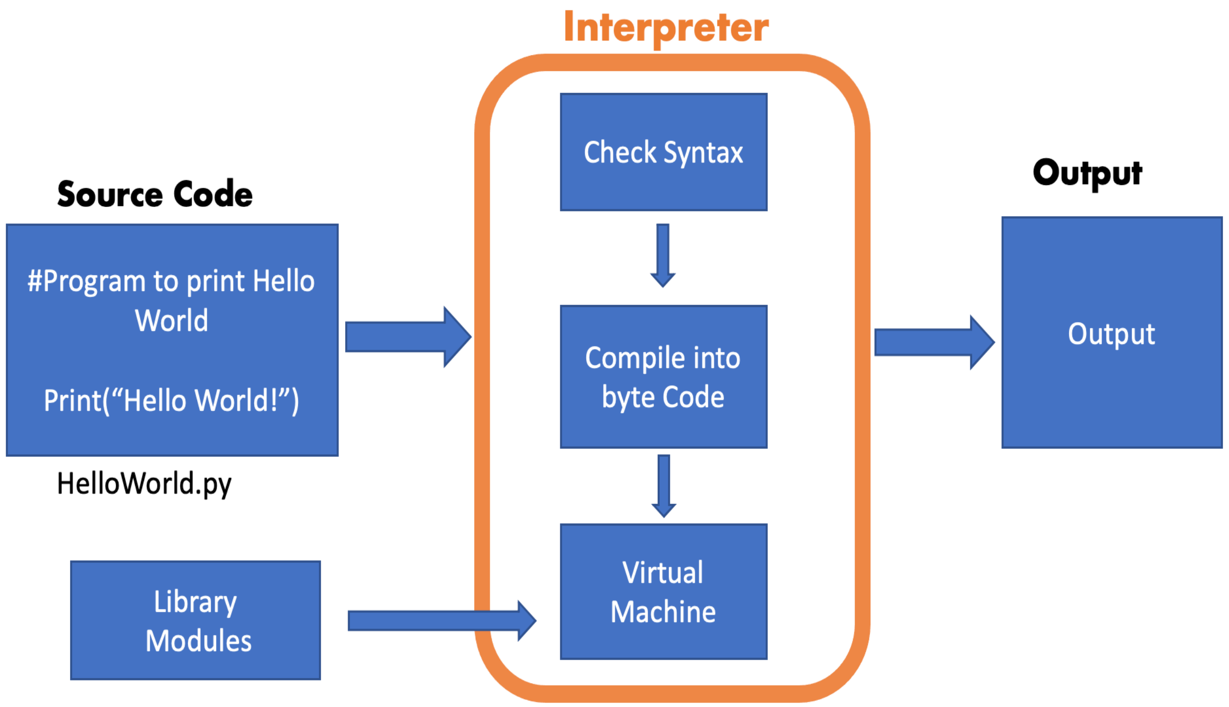
Programming

```
#include <stdio.h>
int main(void)
{
    printf(“Hello World!!\n”);
}
```

Java

```
Class HelloWorld{
    public static void main(Srting args[]){
        System.out.println(“Hello World!!”);
    }
}
```

```
print(“Hello World!!”)
```

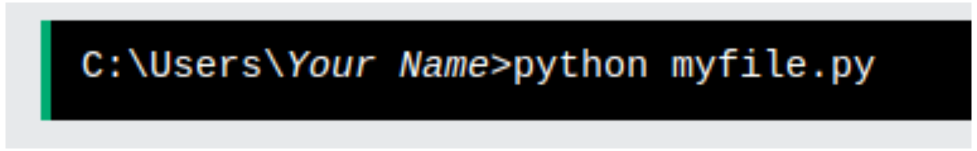


Interpreter Vs. Compiler

Interpreter	Compiler
Translates one statement at a time.	Scans the entire program and translates it as whole into machine code.
Slower than compilers.	Faster than interpreters.
No object code is generated, hence are memory efficient.	Generates object code, requiring more memory.
Example: JavaScript, Python	Example: C, C++, Java

myfile.py

```
# Sample Program
a = 10
b = 5
print("Total Sum: ", a+b)
```

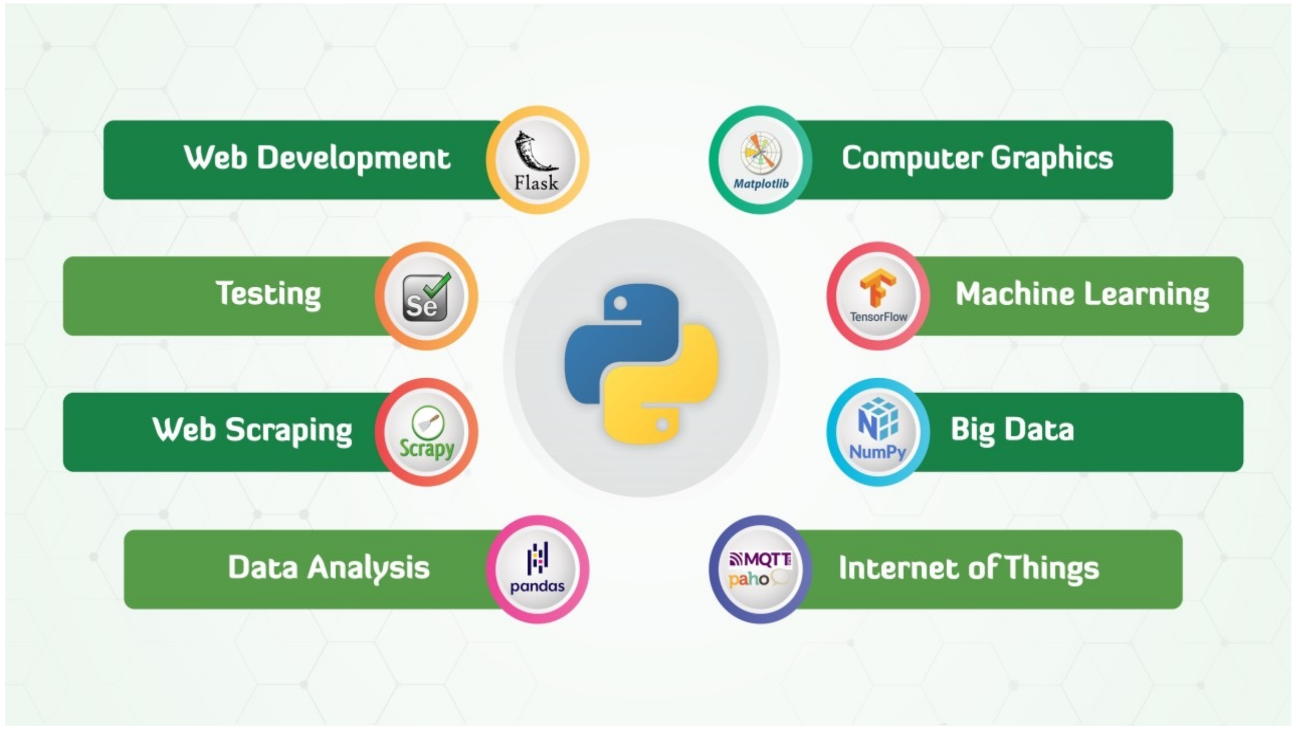


```
C:\Users\Your Name>python myfile.py
```

Output:

```
Total Sum: 15
```

Python has many applications



All of you please learn Python

All of you must learn Python

I will take leave now.

Your professor will handle the rest.

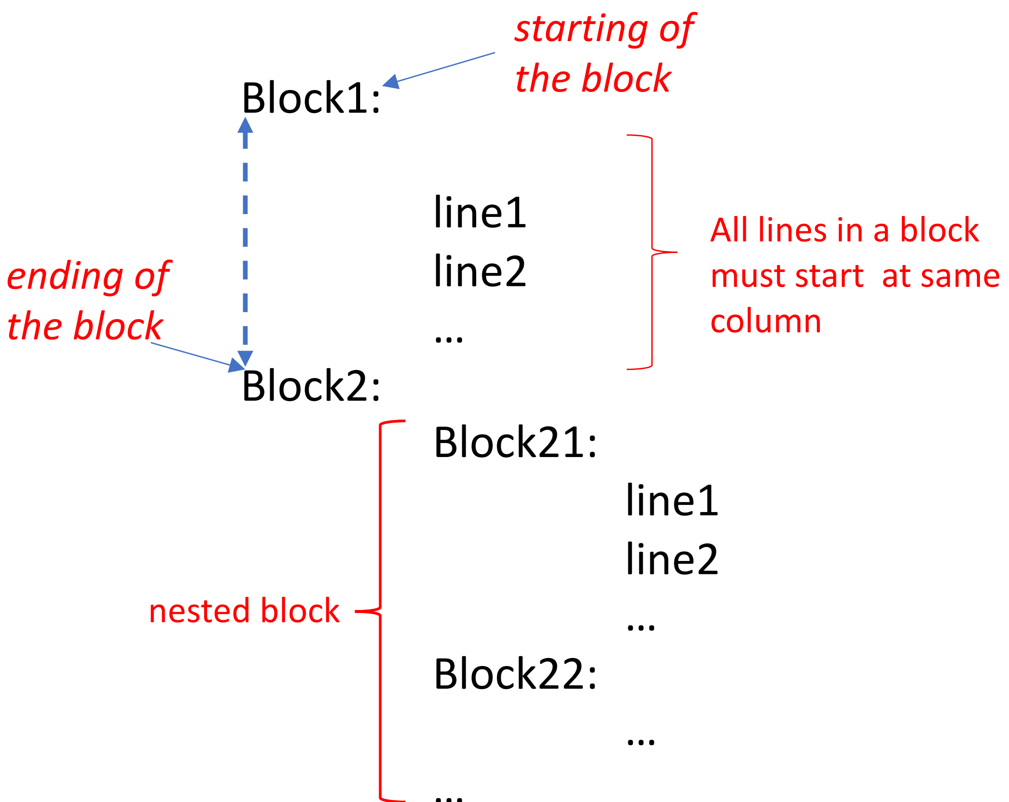
Bye

Outline

- Indentation
- Variables and Casting
- Conditions and Loops
- Functions and
- Arguments

Indentation

- Vertical arrangement of code.
- Python uses indentation to indicate a block of code

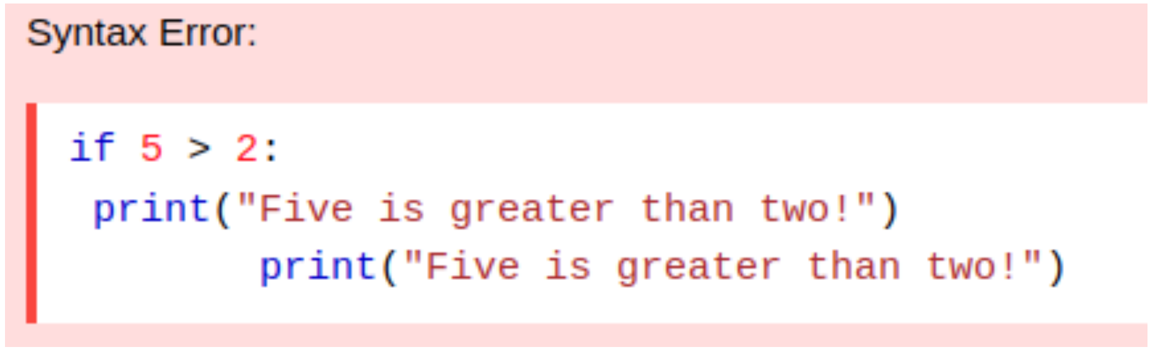


```
Block1:

        line1
        line2
        …
Block2:
        Block21:
                line1
                line2
                …
        Block22:
                …
        …
```

Example With Proper Syntax

```
if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
        print("Five is greater than two!")
```



Syntax Error:

```
if 5 > 2:
 print("Five is greater than two!")
        print("Five is greater than two!")
```

Variables and Casting

- Variables are created by assigning a value.
- No need to declare the type of the variable

```
x = 5
y = "Hello, World!"
```

- Variables conversions/casting can be done implicitly or explicitly

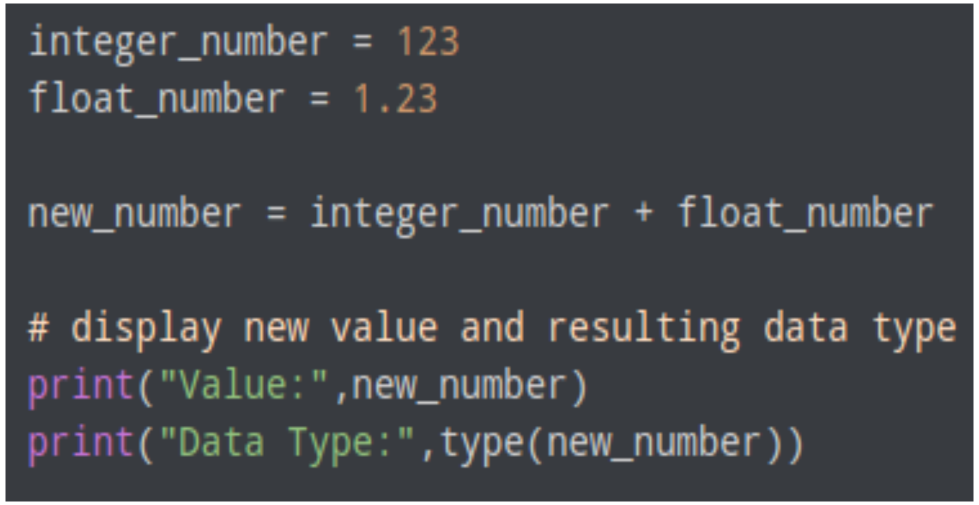
$$\begin{tikzcd} & \text{Number} \\ \text{Number} \arrow[r] & \textit{to} \arrow[u] \arrow[d] \\ & \text{String} \end{tikzcd}$$

implicit casting

$$\begin{tikzcd} \text{String} \arrow[r] & \textit{to} \arrow[r] & \text{Number} \end{tikzcd}$$

explicit casting

Implicit casting

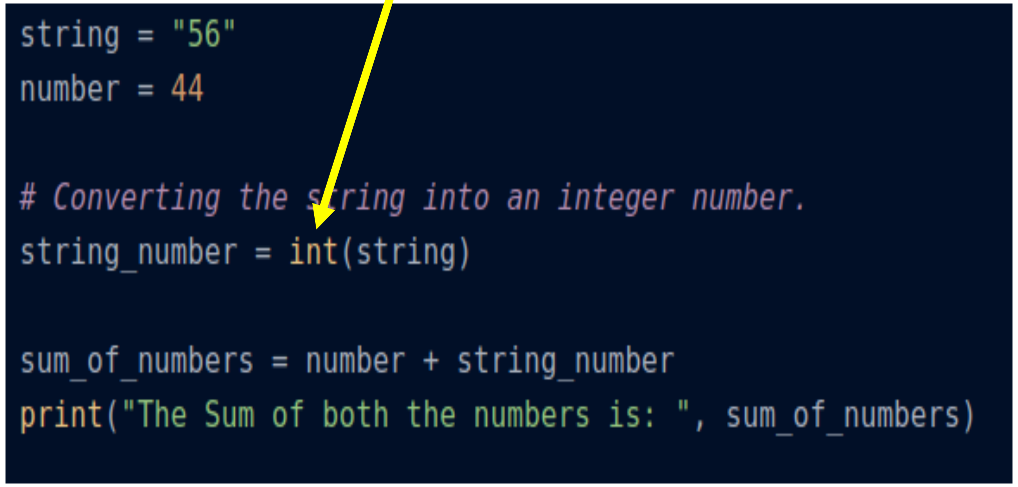


```
integer_number = 123
float_number = 1.23

new_number = integer_number + float_number

# display new value and resulting data type
print("Value:",new_number)
print("Data Type:",type(new_number))
```

Explicit casting



```
string = "56"
number = 44

# Converting the string into an integer number.
string_number = int(string)

sum_of_numbers = number + string_number
print("The Sum of both the numbers is: ", sum_of_numbers)
```

Conditions and Loops

- Python supports the usual logical conditions from mathematics

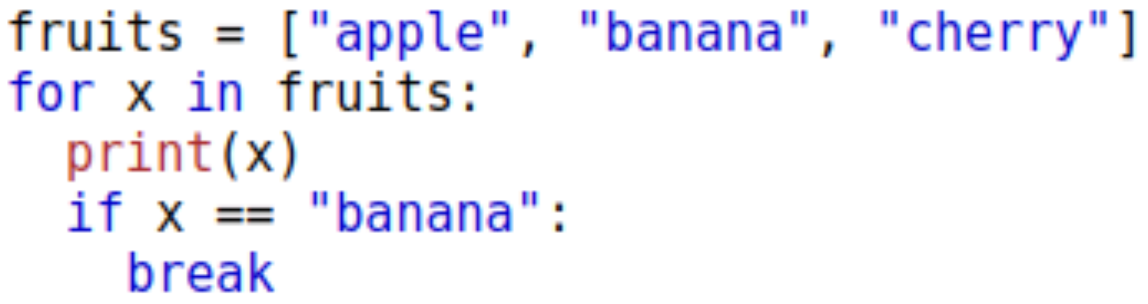
- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Conditions and Loops

- A for loop is used for iterating over a sequence(that is either a list, tuple, a dictionary, a set, or a string) .Works more like an iterator.

- break Statement : With the break statement we can stop the loop before it has looped through all the items.



```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
Output :   apple
           banana
```

Functions

- A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function.

- A function is defined using the ***def*** keyword.

- Call a function using its name and arguments in parenthesis.

Example :

declarating a function

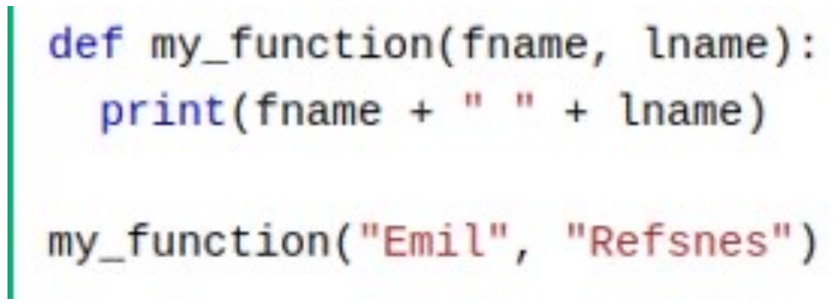
```
def my_function():
    print("Hello from function")

my_function()
```

calling a function

Arguments

- Information can be passed into function as arguments.
- We can add as many arguments as we want
- Arguments are separated with a comma



```python
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

- ***<u>We can set default value to an argument</u>***

default value

```
def my_function(fname=“RAGE”, lname)
            print(fname + ” “ + lname)


my_function(“Uday Kiran”)
my_function(“Musashi”, “Ito”)
```

Output

```
RAGE Uday Kiran
Musashi Ito
```

Arguments

- Arbitrary Arguments - If you don't know how many arguments that will be passed into your function. (add a * before the parameter name in the function definition.)

- Keyword Arguments - You can also send arguments with the key=value syntax. This way the order of the arguments does not matter.

- Arbitrary keyword Arguments - add two asterik ** before the parameter name in the function definition. This way the function will receive a dictionary of arguments, and can access the items accordingly.

Arbitrary Arguments

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

Keyword Arguments

```
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

Arbitrary Keyword Arguments

```
def my_function(**kid):
  print("His last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes")
```

The END