

Information Analysis and Visualisation

Python Programming Language

University of Greenwich
Konstantin Kapinchev

Information Analysis and Visualisation

Python Programming Language

- **Historical Overview**

- The Python programming language was introduced by Guido van Rossum¹ in the late 1980s and early 1990s
- The initial aim was to develop a language, which combines the positive characteristics of the general-purpose languages, such as C², and a shell script³ (as used in a POSIX environment)
- The latest stable release of the language is version 3.11.1 (2022)

¹<https://www.youtube.com/watch?v=J0Aq44Pze-w>

²The main purpose of the C programming language was to allow OS to be developed by using high level language, as opposed to an assembly language. It is the language used in all UNIX-like OS, including Linux and MacOS

³Initially POSIX OS used Command Line Interface (CLI) with typing commands being the main interaction between the user and the computer

Information Analysis and Visualisation

Python Programming Language

- **"Hello World" program in Python and other languages**
 - The purpose of the "Hello World" programs:
 - Test the translator (compiler/interpreter)
 - Test the output of the environment
 - Introduce a new programming language to learners
 - Illustrate basic functionality

Information Analysis and Visualisation

Python Programming Language

- **"Hello World" Examples**

- These are the "Hello World" programs written in 3 different programming languages, C++, Java and Python

```
// C++
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {  
    cout << "Hello World!";  
    return 0;  
}
```

```
// Java
```

```
public class Main { public static void main(String[] args) {  
    System.out.println("Hello World");  
}
```

```
# Python
```

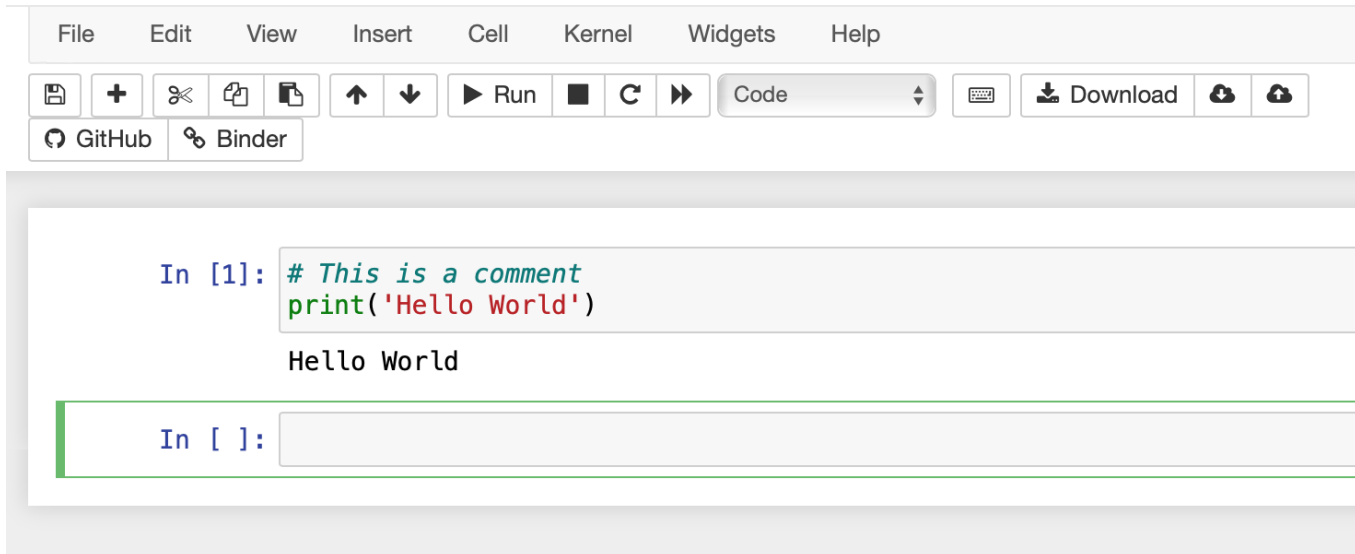
```
print('Hello World')
```

Information Analysis and Visualisation

Python Programming Language

- **Input and Output Functions**

- The 'print' function is the basic output function in Python
- When invoked, the print function will display the content of its parameters on the console



The screenshot shows a Jupyter Notebook interface. At the top is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, running, and other actions. There are also buttons for 'Code', 'Download', and social media links for GitHub and Binder. The main area contains a code cell with the following text:

```
In [1]: # This is a comment  
print('Hello World')
```

Below the code cell, the output 'Hello World' is displayed. At the bottom, there is an empty code cell with the prompt 'In []:'.

Information Analysis and Visualisation

Python Programming Language

- **Input and Output Functions**

- Function 'print' can be used with multiple parameters, mixing variables and textual or numerical values

```
In [2]: c = input('Type a text: ')\n        print('You just typed', c)
```

```
Type a text: Hello\nYou just typed Hello
```

```
In [ ]:
```

```
In [3]: a = input('Enter a:')\n        b = input('Enter b:')\n        c = a + b\n        print('The answer is:', c)
```

```
Enter a:5\nEnter b:10\nThe answer is: 510
```

```
In [ ]:
```

Information Analysis and Visualisation

Python Programming Language

- **Input and Output Functions**

- Textual and numeric data

```
In [3]: a = input('Enter a:')  
        b = input('Enter b:')  
        c = a + b  
        print('The answer is:', c)
```

```
Enter a:5  
Enter b:10  
The answer is: 510
```

```
In [ ]:
```

```
In [15]: a = int(input('Enter a: '))  
         b = int(input('Enter b: '))  
         c = a + b  
         print(a, '+', b, '=', c)
```

```
Enter a: 5  
Enter b: 10  
5 + 10 = 15
```

```
In [ ]:
```

Information Analysis and Visualisation

Python Programming Language

- **Input and Output Functions**

- Formatting numerical output

```
In [1]: a = int(input('Enter a: '))
        b = int(input('Enter b: '))
        c = a / b
        print(a, 'divided by', b, 'is', c)
```

```
Enter a: 10
Enter b: 3
10 divided by 3 is 3.3333333333333335
```

```
In [ ]:
```

```
In [2]: a = int(input('Enter a: '))
        b = int(input('Enter b: '))
        c = round(a / b, 2)
        print(a, 'divided by', b, 'is', c)
```

```
Enter a: 10
Enter b: 3
10 divided by 3 is 3.33
```

```
In [ ]:
```


Information Analysis and Visualisation

Python Programming Language

- **Libraries**

- Libraries extend the functionalities of the programming languages
- In most cases implemented as collection of classes and functions
- Can be developed by third parties
- Using third party libraries has advantages and disadvantages

Information Analysis and Visualisation

Python Programming Language

- **Libraries**

- Example:
 - Function 'sqrt' for calculating the square root of a real number

```
a = int(input('Enter a: '))
b = sqrt(a)
print('The square root of', a, 'is', b)
```

Enter a: 16

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-3-9e7c3f41425f> in <module>
      1 a = int(input('Enter a: '))
----> 2 b = sqrt(a)
      3 print('The square root of', a, 'is', b)

NameError: name 'sqrt' is not defined
```

Note:

By checking the Python documentation, we can find out which library defines the function we want to use:

<https://docs.python.org/3/library/math.html>

Information Analysis and Visualisation

Python Programming Language

- Libraries

- Example:
 - Function 'sqrt' for calculating the square root of a real number

```
a = int(input('Enter a: '))
b = sqrt(a)
print('The square root of', a, 'is', b)
```

Enter a: 16

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-3-9e7c3f41425f> in <module>
      1 a = int(input('Enter a: '))
----> 2 b = sqrt(a)
      3 print('The square root of', a, 'is', b)
```

NameError: name 'sqrt' is not defined

```
import math
a = int(input('Enter a: '))
b = math.sqrt(a)
print('The square root of', a, 'is', b)
```

Enter a: 16

The square root of 16 is 4.0

Information Analysis and Visualisation

Python Programming Language

- **Conditional Statement**

- Is the square root function defined for every real¹ number?

```
import math
a = float(input('Enter a: '))
b = math.sqrt(a)
print('The square root of', a, 'is', b)
```

Enter a: -25

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-2-2f5cf49484f8> in <module>
      1 import math
      2 a = float(input('Enter a: '))
----> 3 b = math.sqrt(a)
      4 print('The square root of', a, 'is', b)
```

ValueError: math domain error

¹Changing int to float to allow input of real numbers

Note:

Mathematicians extended the definition of the square root operation in order to work with negative values¹. In this case, square root of a negative value is a complex number in the form of $a+bi$, where i is the imaginary unit².

¹This is reflected in the `cmath` library

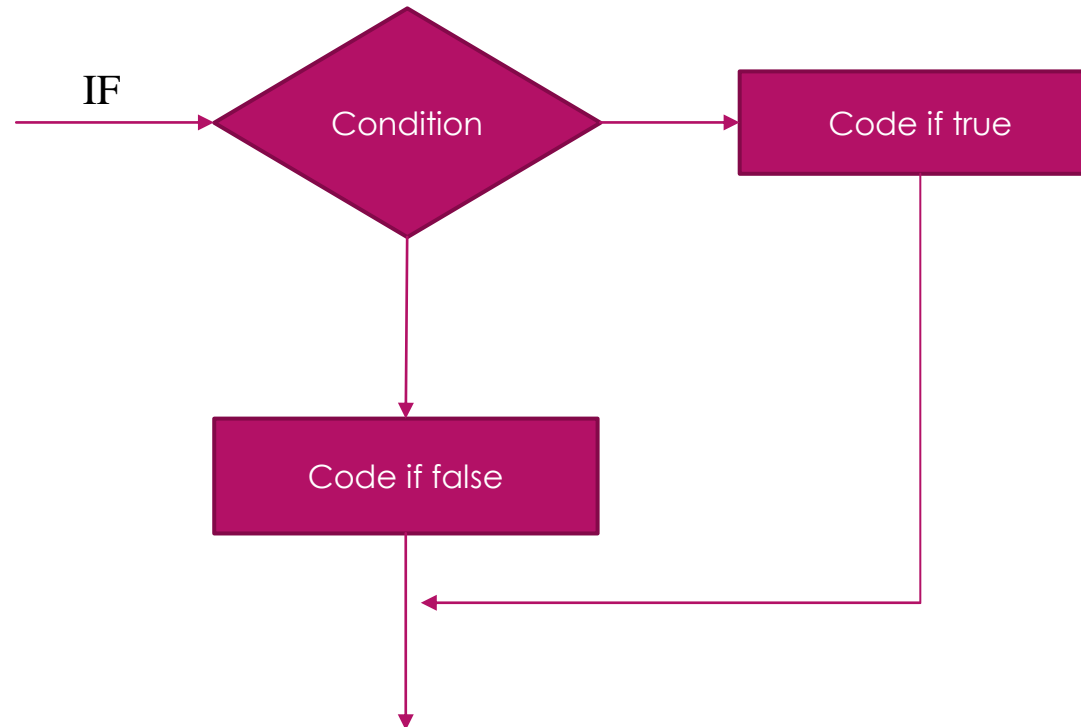
²Some use j instead of i

Information Analysis and Visualisation

Python Programming Language

- **Conditional Statement**

- Block diagram of IF statement



Information Analysis and Visualisation

Python Programming Language

- **Conditional Statement**

- Adding the conditional statement 'if' solves the problem

```
import math
a = float(input('Enter a: '))
if a >= 0:
    b = math.sqrt(a)
    print('The square root of', a, 'is', b)
else:
    print('Negative numbers do not have real square roots')
```

Enter a: 25

The square root of 25.0 is 5.0

Note:

Two possible outputs depending on the value of variable a

Information Analysis and Visualisation

Python Programming Language

- **Conditional Statement**

- Adding the conditional statement 'if' solves the problem

```
import math
a = float(input('Enter a: '))
if a >= 0:
    b = math.sqrt(a)
    print('The square root of', a, 'is', b)
else:
    print('Negative numbers do not have real square roots')
```

Enter a: -25
Negative numbers do not have real square roots

Note:

Result when variable 'a' has a negative value

Information Analysis and Visualisation

Python Programming Language

- **Conditional Statement**

- Adding the conditional statement 'if' solves the problem

```
import math
a = float(input('Enter a: '))
if a >= 0:
    b = math.sqrt(a)
    print('The square root of', a, 'is', b)
else:
    print('Negative numbers do not have real square roots')
```

Enter a: -25

Negative numbers do not have real square roots

Notes:

The keywords 'if' and 'else' are followed by colons.

The indentation of one tab defines the scope of the 'if' statement. This syntax is characteristics of Python.

Also note the notation '>=' for 'greater than or equal to'.

Information Analysis and Visualisation

Python Programming Language

- **Subroutines**

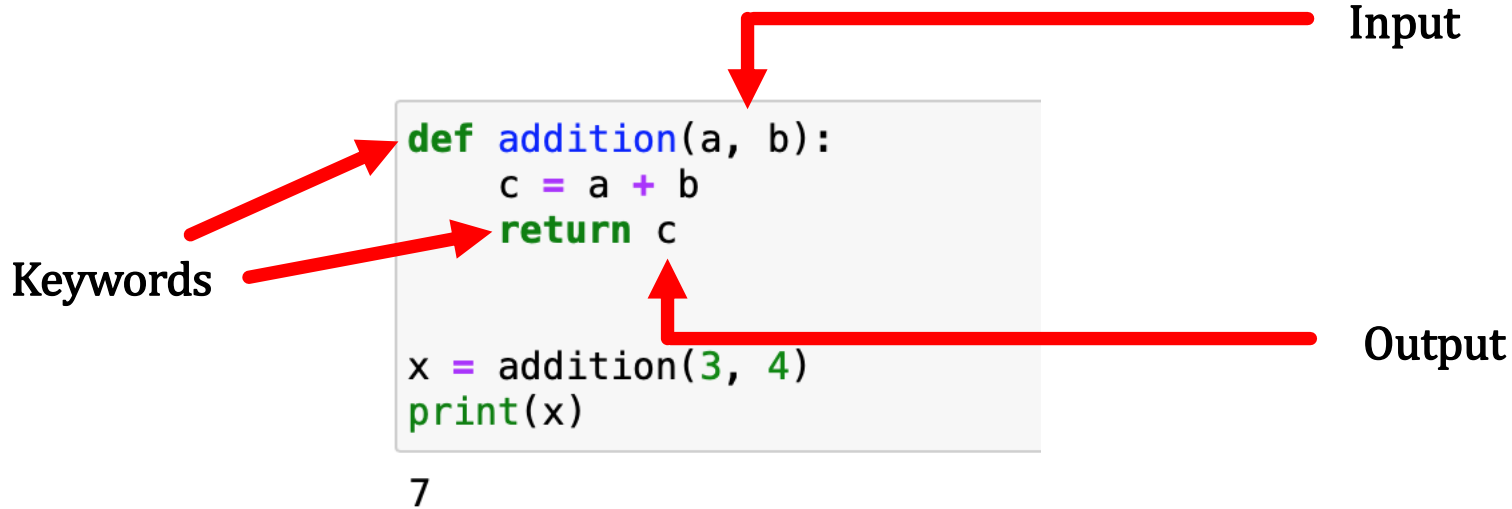
- In practice programs tend to have significant number of lines and statements
- There is a need to organise the programs into logically separated blocks
- Ideally, those blocks would be independent from particular programs, in order to be re-used in other programs
- Most programming languages have their own approach regarding subroutines, although most of them follow some common concepts and principles
- In general, these logically separated blocks are denoted as 'subroutines'
- Most programming languages, Python included, support subroutines

Information Analysis and Visualisation

Python Programming Language

- **Subroutines**

- In Python subroutines are called functions



Note:

The indentation on the second and third lines defines the block of the subroutine, or function.

We observed the same use of indentation as part of the 'if' statement.

Information Analysis and Visualisation

Python Programming Language

- **Subroutines**

- A Python example:

```
def addition(a, b):  
    c = a + b  
    return c
```

```
x = addition(3, 4)  
print(x)
```

7

```
def addition(a, b):  
    c = a + b  
    return c
```

```
x = 10  
y = 20  
c = addition(x, y)  
print(x, '+', y, '=', c)
```

10 + 20 = 30

Note:

Subroutines can be used in multiple situations, including on the right-hand side of the equal sign.

Their parameters can be variables or specific values.