INST0004 Programming 2

Lecture 01: Introduction to Python Programming

Module Leader:
Dr Daniel Onah

2023-24


UCL

UCL

Copyright Note

Important information to adhere to

Learning Outcomes

The learning outcomes for the lecture

The objective of this week's lecture is to introduce the concept of object-oriented programming in Python. At the end of the lecture, you should be able to:

- the basic concepts of object-oriented programming
- understand classes creation
- understand objects creation
- concepts and types of constructors
- concepts of inheritance
- concepts of polymorphism
- various Python methods to achieve object-oriented programming

Module Structure

The module structure for the 10-week lecture

- Week 1 Lecture 1 - Introduction to Python Programming
- Week 2 Lecture 2 - Programming with Variables, Numbers and Strings
- Week 3 Lecture 3 - Control Flow Structures - LOOPS, IF, ELSE and NESTED
- Week 4 Lecture 4 - Object Oriented Programming
- Week 5 Lecture 5 - Building Class Structures, Algorithms and Pseudocode
- Week 6 Lecture 6 - Recursion in Python
- Week 7 Lecture 7 - Classes and Objects
- Week 8 Lecture 8 - Inheritance and Polymorphism
- Week 9 Lecture 9 - Sets, Sorting and Searching
- Week 10 Lecture 10 - Files and Exceptions Handling

Vivas, Research Seminars & Workshops

Weekly assessment, seminars & workshops

- Week 1 - Welcome Introduction to the Module
- Week 2 - TS1 Mock viva
- Week 3 - TS2 Viva
- Week 4 - TS3 Viva
- Week 5 - TS4 Viva
- Reading Week
- Week 6 - **Research Seminar & workshop** in Artificial Intelligence Area: **Natural Language Processing**
- Week 7 - TS6 Viva
- Week 8 - **Research Seminar & workshop** in Artificial Intelligence Area: **Data Science & Mining**
- Week 9 - TS8 Viva
- Week 10 - **Research Seminar & workshop** in Artificial Intelligence Area: **Machine Learning & Deep Learning**

Module Overview

Introducing the module overview

The module overview will be similar to last term

- Weekly Lectures and formative lab exercises
- Weekly vivas for 6 weeks (week 1 Mock viva)

Do engage with the course - we like questions!!!

- In face-to-face plenary sessions
- Group in-class practical sessions
- Moodle discussion forums (research this yourself first)
- We encourage asking your module related questions via Moodle forums so peers could also benefit from the response(s)
- ... occasionally e-mail if you feel it needs to be private and urgent
- DO NOT SEND ME DIRECT MESSAGE VIA MOODLE. I MIGHT NOT BE ABLE TO SEE THIS.

Reference Texts ...

Essential and Recommended Textbooks

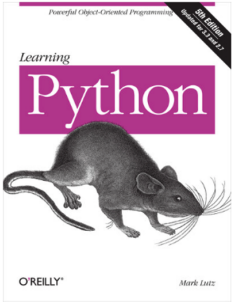
This module would not be taught directly from ANY ONE particular textbook ...

- But these books form good reference texts.
- They are all or some in the UCL library (maybe earlier versions)
- They are or mostly available on e-readers or electronic versions in the UCL library
- You can check the module reading list for other textbooks.

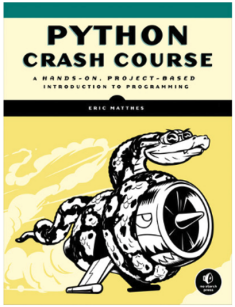
Reference Texts ...

Essential and Recommended Textbooks

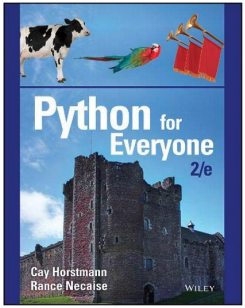
ISBN: 978-1449-35573-9



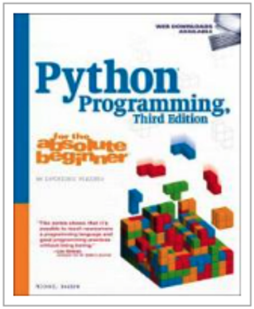
ISBN-10: 1-59327-928-0



ISBN: 978-1-119-05655-3



ISBN-13: 978-1-4354-5500-9



So You Like Programming!!!

A skill-set achieve in programming.

Obviously to succeed in this module and if you want to pursue a career in software development and programming related roles, you should be interested in programming!

Modern Application Programming Language ...

Everything done in Programming.

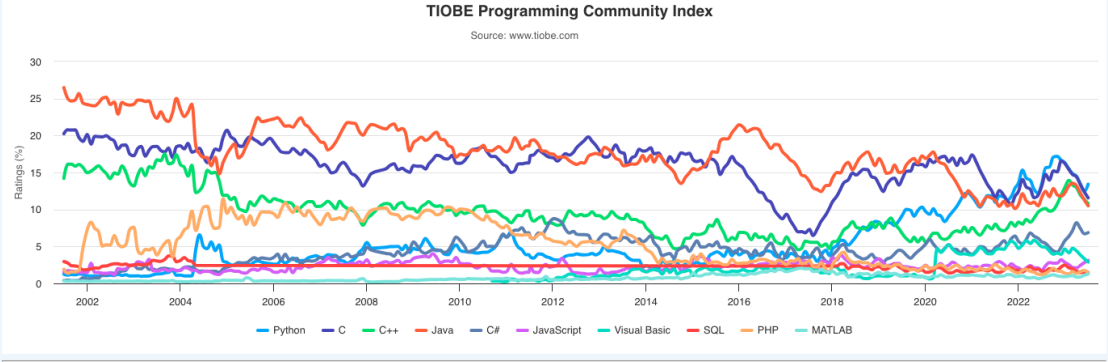
Commercial software development today is:

- **Developed under heavy time constraints:** A language therefore needs to promote reuse and integration of software.
- **Developed in teams:** A language needs to promote high levels of modularity so they could be developed in parallel.
- **Mission critical for their customers:** Languages should trap faults at compile time and runtime.
- **Run on many platforms** (iOS, macOS, Windows, Linux, Ubuntu, Android, Web ...).
- Programming software should be platform independent.

Today's modern programming & software development is written by a lot of people, collaborating and exchanging good coding ideas.

Popularity of General Purpose Languages ...

Programming language ranking & Ratings



But Do You Like Python?

Significance of Python

What do you like about it?

- Concise: Syntax lets you define quite a lot without the code feeling too bloated.
- Established: Python has been used for decades, is well understood by programmers the world over.
- Efficient: Your program takes little space on disk and in the computers memory, and has little computational overhead.
- Close to the hardware: Very good for OS development, device, drivers, embedded devices, web technologies and scripting etc.
- **Python gives a lot of power to the programmer ...**

About Python ...

About Python programming

Python is a modern, platform independent, object oriented programming language. Python is a modern, platform independent, object oriented programming language. Overarching design goal: simplicity and reuse.

About Python ...

About Python programming

Python is a modern, platform independent, object oriented programming language.

- Objects are at the heart of language. An object is a set of data (variables) and set or operations that operate on that data (methods).
- Objects are the unit of modularity Object Oriented (OO) programs.
- The OO vision is to split your program into many independent objects – often as many as you can*

Lots of small objects ...

- Development parallelism (teams!)
- Ease of reuse (the smaller it is, the less specialized it will be).
- OO idolizes the vision: “write no new code”.
- *except when that’s too many.

Our Implementation Could Be Buggy ...

Program constructs and implementation

"As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought it would be. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."

— Maurice Wilkes

Machine Wrong Figures! ...

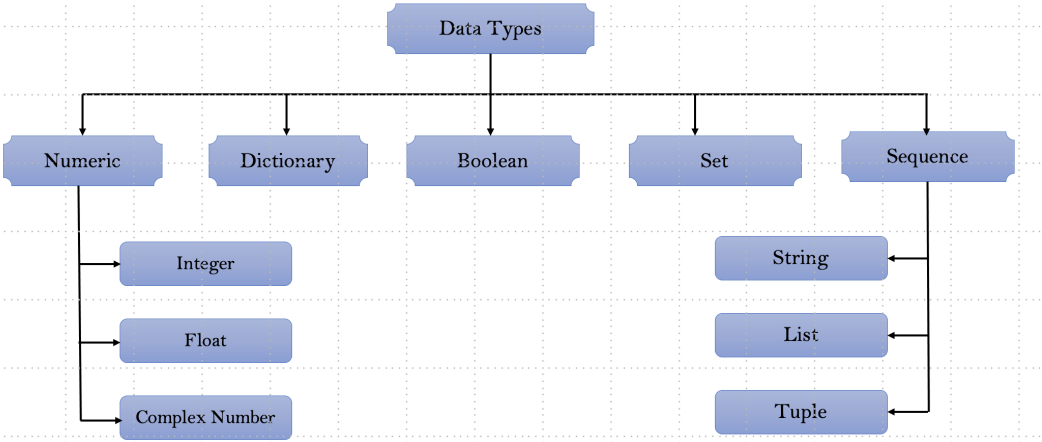
Program constructs and implementation

“ On two occasions I have been asked, ‘Pray, Mr. Babbage, if you put into the machine wrong figures, **will the right answers come out**?’ I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.”

- Charles Babbage. (*A Mathematician, Philosopher, Inventor ...*)

Python Data Types ...

What are the data types in Python?



Reminder - Executing a Python Program

About Python programming

Create a text file with your favourite editor **(VSCode, Sublime, Notepad++)**.

- Your source code filename should be saved in a file extension ending with (.py)

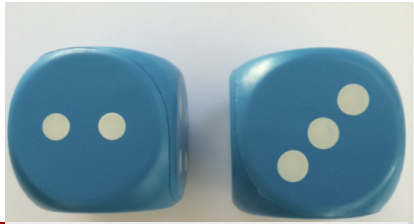
Open a command line prompt

- shell in Unix
- terminal in MacOS
- cmd in Windows
- Change directory to the location of your file, then execute your program source code
- Now start and run a Java virtual machine that interprets your program: **python3 filename.py**
- start from page 20

Important Facts

Useful points to note

- Statements instructs the computer program to perform the actions.
- A string in **double quotes** is sometimes called a character string or a string literal
- Standard output function known as print() displays characters or results of the program in the command window.
- print() function for example, written as print("\n") displays its argument in command window follow by a newline character to position the output cursor to the beginning of the next line.



Important Facts

Useful points to note

- If an if statement's condition is true, **its body executes**.
- If the condition is false, its body will not execute.
- Every program consist of combination of sequence, selection and iteration statements.
- To perform a floating-point operation with integer, cast the integer to type double.
- Python knows how to perform calculations using operand of the same type. Promotion is used to perform operations of unidentical operand types.
- Type Casting is by placing parenthesis around the name type e.g. float(value).

Equality & Relational Operators

Decision making

- Conditions in if statement can be formed by using the equality operators e.g. == and !=
- The relational operators e.g. >, <, >= and <=
- Both equality operators have the same level of precedence, which is lower than the relational operators.
- The equality operators associate from left to right.
- The relational operators all have the same level of precedence and associate from left to right.
- The assignment operator associate from right to left.

If Statement

Condition within the body

- If the condition of an if statement is true, the corresponding body statement displays an appropriate line of text.
- Each If statement contains a single body statement holding an indented line of text to display.
- Each body statement in an If statements are **indented** in blocks.
- These indentation is used in **if statement** to create a compound statement or block of codes.

Good Programming Style

Good programming practice

- Indent the statement(s) in the body of your class, conditional statements to enhance readability.
- Follow through the Python convention in your code development (classes, functions, variables etc) and **you will surely progress to be a good programmer**

White Space

Program ignore white spaces occasionally

- The Python interpreter normally ignores white spaces during program execution. E.g. **blank lines, tab, space characters**
- White space makes programs easier to read
- Programs may be split into several lines and may be spaced according to your preferences without affecting a program's meaning.
- Note that it is incorrect to split identifiers and strings. Ideally, statements should be kept small , but this is not always possible.

Preventing Error

Python programming is prone to indentation and other errors

- A lengthy statement can be spread over several lines.
- If a single statement must be split across lines, choose natural breaking points, such as after a **comma** in a *comma-separated lines*, or after an **operator** in a *lengthy expression*.
- If a statement is split across two or more lines, *indent all subsequent lines until the end of the statement*.

Assignment Operator

Associativity from right to left

The assignment operator associate from right **to** left. An assignment expression's value is whatever was assigned to the **variable** on the = operator's **left side**. e.g. x = 7

- The value of the above expression is 7.
- So for an expression such as: x = y = 0
- This expression is evaluated as if it has been written as: x = (y=0)
- Which first assigns the value **0** to the variable y
- Then assign the result of the assignment **0** to x.

Precedence & Associativity of Operators

What are the precedence and associativity of operators in Python?

Operators	Associativity	Type
* / %	Left to right	multiplicative
+ -	Left to right	additive
< <= > >=	Left to right	relational
= !=	Left to right	Equality
=	Right to left	assignment

Syntax Error

What are syntax errors?

Syntax errors are commonly found in programming languages.

- A syntax error occurs when the interpreter encounters code that violets **Python's** language rules.
- Its similar to a grammar error in a natural language. The *ability to identify and correct them*, make you a better programmer.

Good Programming Practice

Conventional best practice

When writing expression containing many operators:

- Refer to the operator precedence
- Confirm that the operations are perform in the order you expect

If in a complex expression and you are uncertain about the order of evaluation:

- Use parenthesis to force the order, *exactly as in algebraic expressions*.

Python Classes

What are Classes in Python?

A class is a blueprint for which many instances of objects can be created. *Python programs are made up of one or more classes*. They are Python's unit of modularity **...** they define objects.

- Remember in object oriented programming vision – so programs normally have loads of classes!
- A class keyword defines a unique name for the class you're writing.
- Python indentation encapsulate the block of code written as part of a class, function and in general Python program.
- *A function are modular blocks of code that have names, parameters and return types*.

Variable declaration

Brief points about variables?

In Python, variables can be defined anywhere.

- You don't need to declare your variable at the start of a function
- Can even be declared in the middle of a code block, but ...
- Are only in scope as defined by the code block.

Methods

Class methods?

In Python, class method have the following features:.

- Method have a scope where it is define
- Tab to indent the lines (indentation)
- Method must be defined inside a class
- Methods can be overloaded
- Local scope variables are access within a class method

Method Overloading

What is method overloading?

Method overloading is phenomenon that **two or more methods** with the same name within a class can behave differently depending on what they are operating on. All methods with the same name can be invoke within a class. All methods declared within a class with the same name , must have different parameter list and they will behave differently, that's what differentiate them. **Note:** *The concept of method overloading is more efficient with other object oriented programming languages such as* ***Java****.*

Important Facts

Essential facts?

- Program statements instructs the computer program to perform the actions
- A string in double quotes is sometimes called a character string or a string literal
- Standard output object methods such as **print()** displays characters or results of the program in the command window
- Method such as *print("\n")* displays its argument in command window follow by a newline character to position the output cursor to the beginning of the next line.

Polymorphism

What is polymorphism?

Polymor – Takes multiple forms. Polymorphism is a class that takes multiple forms. Its the process of treating two **different types of object** as if they are **the same** in the same way using interfaces. Method overloading is actually one example of **polymorphism**. It is an important feature of object-oriented programming languages. This refers, in general, to the phenomenon of **having methods and operators with the same name** performing different functions.

While Loops

Brief points about while Loops?

While loops are identical. Let's look at the **syntax**:

Syntax

```
#!/usr/bin/env python
"""
A simple while loop construct
"""
while (boolean_condition):
        #code to execute goes here
```

While Loops

Brief points about while Loops?

Let's look at a simple example of a while loop construct:

```
1 #!/usr/bin/env python
2 """
3 A simple while loop construct
4 """
5 i = 0
6 while(i < 10):
7       print("INST0004 Programming 2")
8       i = i + 1
```

For Loops

Brief points about for Loops?

For loops also work much the same ...

- Except we can now create loop variables as we need them ... which is safer.
- The loop variable is only in scope within the loop.

Let's look at the **syntax**

Syntax

```
#!/usr/bin/env python
"""
A simple for loop construct
"""
for (init_statement; boolean_condition; modifier_statement):
```

For Loops

Brief points about for Loops?

Let's look at an example:

```
#!/usr/bin/env python
"""
A simple for loop construct
"""
for x in range(1, 100):
    # body of loop define here.
```

Strings

Brief points about Strings?

Strings however are very different. In Python, sequence of characters are used to represent a string.

- Python has a specific String type that's safer and more expressive.
- Note: String has a capitalized first letter unlike primitive types.
- **ALSO** when copying and pasting, be careful that the quotes are correct! Always try to type your code in.
- Strings can be concatenated to form other strings ... Uses the + operator.

```
1 #!/usr/bin/env python
2 """
3 A simple of a string construct
4 """
5 h = "Hello"
6 w = "World"
7 print(h + w);
```

What's wrong with the output here though?

Strings

Brief points about Strings?

Strings however are very different. In Python, sequence of characters are used to represent a string.

- What's wrong with the previous code output here though?
- The String construct below is Better!

```
#!/usr/bin/env python
"""
A simple of a string construct
"""
h = "Hello"
w = "World"
print(h + " " + w)
```

Strings

Brief points about Strings?

Strings also now know how big they are ...

- len() method provides this:

```
#!/usr/bin/env python
"""
A simple of a string construct with len() method
"""
name="Dan"
if(len(name)  < 4):
     print("Wow! this a really short name.")
```

Strings

Brief points about Strings?

Strings also now know how big they are ...

- But take care while comparing strings ...
- Use the "==" operator to check for the equality of two strings

```
#!/usr/bin/env python
"""
A simple of a string construct with "==" operator
"""
name1 = "Dan"
name2 = "Dan"
if (name1==name2):
    print("Thats my name!")
```

Strings Escape Character

Escape sequence in String.

Strings also now know how big they are ... We can use several escape sequence with String literals.

- **\t** Insert a tab in the text at this point
- **\b** Insert a backspace in the text at this point
- **\n** Insert a newline in the text at this point
- **\r** Insert a carriage return in the text at this point
- **\f** Insert a form feed in the text at this point
- **\'** Insert a single quote character in the text at this point
- **\"** Insert a double quote character in the text at this point
- **** Insert a backslash character in the text at this point

Strings escape character

Escape character in Strings.

Let's look at an example of an escape character.

```
#!/usr/bin/env python
"""
A simple a string construct with escape character
"""
h = "Hello"
w = "World"
print(h + "\t " + w + "\n" + "Greeting by Danny")
```

Python String format() method

Escape character in Strings using format() method.

Let's look at an example of an escape character with String format() method.

```
#!/usr/bin/env python
"""
A simple a string construct with escape character and format() method
"""
h = "Hello"
w = "World"
print('\t{}  \t  {} , \t  said Danny'.format(h, w))
```

Number Guesser

Number guesser program

```
#!/usr/bin/env python
"""
A simple number guesser program
"""
import random
def main():
    myNumber = random.randint(1, 100)
    print("The number is between 1 and 100.")
    while True:
        guess = int(input("Enter your guess: "))
        if(guess == myNumber):
            print("Correct. Well done!")
            break
        else:
            print("Wrong. Try Again!")
main()# if this is not called program will not work
```

Maximum Number

Maximum number program

```
#!/usr/bin/env python
"""
A program to display the maximum of three numbers
"""
def main():
    num1 = int(input("Enter first number: "))
    num2 = int(input("Enter second number: "))
    num3 = int(input("Enter third number: "))
    maxOf2 = findMaximum(num1, num2)
    maxOf3 = findMaximum(maxOf2, num3)
    print("The maximum number is: ", maxOf3)
def findMaximum(num1, num2):
    if(num1 > num2):
        return num1
    else:
        return num2
main()
```

Key Points

Key points to remember about input & output...

Input and Output

```
name = input("Your name: ")
age = int(input("Your age: "))
weight = float(input("Your weight: "))
print("You are", age, "years old.")

print("Your age: ", end="")
```

No newline after output

Field width

Precision

```
print("%-20s Age:%3d %8.2f kg" % (name, age, weight))
```

Left-justified string

Integer

Floating-point number

Python Important Facts

Import points to note

Python is designed to be:

- *General purpose*
- *Simple, safe and open*
- *Object Oriented*
- *Ideal for reuse and extensibility*
- Python programming is universal


Summary

Summary

Let's revise the concepts of today's lecture

In this lecture we discuss the following:

- In this section, you learned the important features of Python programming including:
- understanding variables, operators and precedence
- classes and object constructs
- basic methods overloading and ploymorphism
- **conditional statements** and **loops**
- taking user **input data** and **displaying output**
- performing program structure and making decisions
- introduce you to many basic concepts and good Python programming constructs.
- finally we demonstrated a few program executions in Python.

Further Reading

chapters to find further reading on the concepts

You can read further this week's lecture from the following chapters:

- Python for Everyone (3/e) : **By Cay Horstmann & Rance Necaise** - *Chapter 1 Introduction*
- Learning Python (5th Edition): **By Mark Lutz** - *Part 1 Getting Started - Chapter 1 A Python Q&A Session*
- Python Crash Course - A Hands on, Project based Introduction to Programming (2nd Edition): **By Eric Matthes** - *Part 1 - Chapter 1 Getting Started*

UCL

Next Lecture 2

In week 2

Lecture 2: Programming with Variables, Numbers and Strings