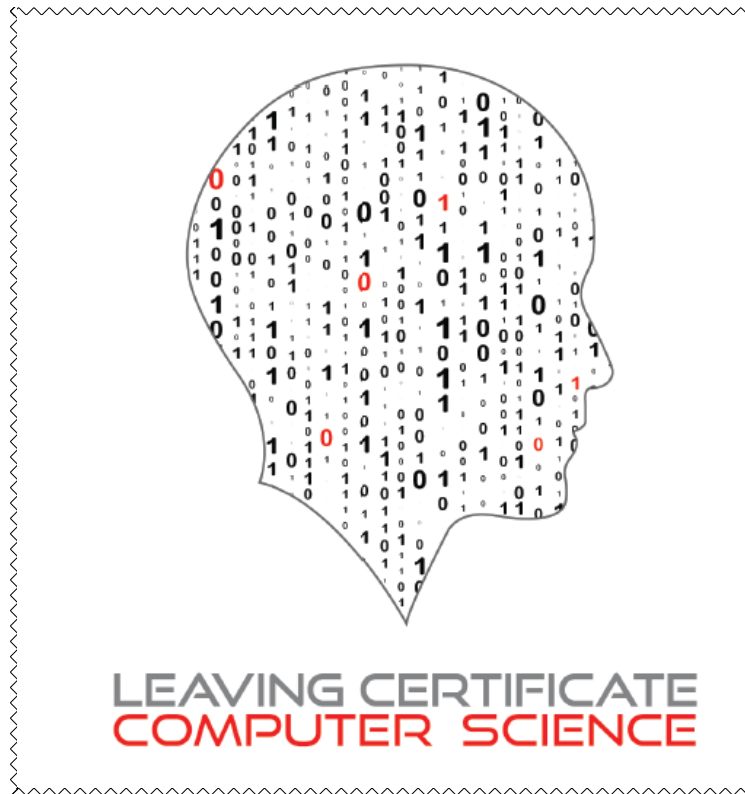


LC CS Python

Student Manual



Section 1

Getting Started

Name: _____

Installation and Setup

All of the examples in this manual were tested using the *Thonny* IDE (Integrated Development Environment). You should download *Thonny* by searching for it in your web browser and following the steps to download it. As we progress, you will have to install certain language extensions such as `pygame` and `plotly`.

- In handout, do **Task 1**.

Throughout this course we will be creating new files, typing in and testing Python code which is provided. The aim is to get to the point where we can write our own code.

- In handout, do **Task 2**.

A point well worth noting at an early stage is the difference between *programmers* and *end-users*.

- Programmers usually work as part of a team. They write and test the code that makes up a computer system. As a programmer, you should bear the needs of the end-user in mind i.e. see the system from the perspective of the end-user.
- An end-user is the person (or organisation) for whom a software system is developed. End-users are the customers and, very often, do not know how to program.

Language Syntax

You are probably already aware that natural languages such as English, French, German, Polish etc. have their own rules. These rules make up the language *grammar*. The syntax of a language is that part of the grammar which tells you how sentences are constructed – syntax is mostly concerned with legitimate words, symbols and the order in which they are used.

In a similar way, all programming languages (e.g. Python, Java, JavaScript, C++, PHP, Perl etc.) have their own syntax – this is called the *language syntax*.

One important aspect of Python's syntax is its vocabulary i.e. the words and symbols that Python understands.

Words can be keywords or commands. The list of all of Python's 33 keywords is given below – only some of these will be needed for LCCS.

False break else if not while

None	class	except	import	for	with
True	continue	finally	in	pass	yield
and	Def	for	is	raise	
as	Del	from	lambda	return	
assert	elif	global	nonlocal	Try	

Python 3.6.2 keywords

Programmers can add to Python's vocabulary by defining their own words.

The most common kinds of symbols in Python are operators – these can be arithmetic or relational.

Python also understands white spaces (e.g. spaces, tabs, newlines), numbers and strings (anything enclosed in quotation marks) – more on these later.

All programs must adhere to the syntax of the programming language in which they are written. When a program does not conform to the language's syntax it is said to contain a *syntax error*. Such programs are said to be *syntactically incorrect*.

When you try to run a program that has a syntax error, Python displays a syntax error message.

Comments are a way to tell Python to ignore syntax. They are used by programmers to make it easy for another programmer to understand their code. Comments in Python start with the hash character, #, and extend to the end of the physical line. When Python comes across the hash character, #, it ignores the rest of the text on that line.

Basic Python Syntax

We will now take a look at some of the basic syntax rules of Python and illustrate what happens when these rules are broken.

Syntax Check: Python is case sensitive. This means that Python treats upper and lower case letters differently. For example, Python understands `print` but does not understand `Print`

Try running the following:

```
Print("Hello World")
```

You will see a message like this:

```
Traceback (most recent call last):  
  File "C:/PDST/Python Workshop/src/hello1.py", line 1, in <module>  
    Print("Hello World")  
NameError: name 'Print' is not defined
```

This is Python's way of telling the programmer that the program contains a syntax error.

Python keywords and commands must all be typed in lower case.



- In handout, do **Task 3**.

Syntax Check: Use opening and closing brackets after the `print` command when you want to display output. For every opening bracket there needs to be a matching closing bracket.

Try running the following:

```
print "Hello World")
```

You will see a syntax error displayed in a message box like this:



- In handout, do **Task 4**.

Syntax Check: When you want to display text using the `print` command, the text must to be enclosed inside matching quotation marks. If either, or both, quotation marks are missing a syntax error message is displayed.

Experiment: Try each of the following lines separately.

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

Quotation marks can be single (`'`) or double (`"`) – it does not matter as long as they match.



What one question do you have so far?



KEY POINT: The technical word for text is *string*. A string is any text enclosed inside quotation marks.

Python is not too fussy about what you type inside quotation marks. Outside quotation marks, Python is very limited in what it understands. One thing Python understands outside quotation marks is number. Numbers do not have to be enclosed inside quotations.

Each of the following lines are syntactically correct. (Try them!)

```
print("What is your age? ")
print(21)
print("My age is 21")
print("My age is", 21)
print("I am", 18, "and my friend is", 21)
print("The print command", "can handle", "more than 1 string.")
```

Notice from the last three examples how `print` allows strings and numbers to be separated by commas.

- In handout, do **Task 5**.

Escape Sequences

Let's say we wanted to display the following text exactly – including the quotation marks.

In the words of Nelson Mandela, "Education is the most powerful weapon which we can use to change the world"

The line below does not work because in the 'eyes' of Python the second quotation closes the first and the remainder of the line is not understood.


```
print("In the words of Nelson Mandela, "Education is the most
powerful weapon which we can use to change the world")
```

To fix the syntax error we *escape* the second quotation using the backslash character, `\`, as follows:

```
print("In the words of Nelson Mandela, \"Education is the most  
powerful weapon which we can use to change the world\"")
```

In the above example the use of `\` tells Python include the double quotes as part of the string (as opposed to treating it as the closing quote).

The backslash character introduces an *escape sequence* in a string. Some common escape sequence characters are illustrated in the table below:

 Python <i>First 5 escape sequences</i>	Escape Sequence	Meaning
	<code>\n</code>	Newline
	<code>\t</code>	Tab
	<code>\'</code>	Single Quote
	<code>\"</code>	Double Quote
	<code>\\</code>	Backslash

- In handout, do **Task 6**.



What was the main thing you learned in this section about escape sequences?

Flow of Control

The *flow of control* refers to the order in which the lines of a computer program are run by the computer. Normally lines are executed in the same sequence in which they appear. This type of flow is called sequential. We use the following four-line program to illustrate this concept.

```
1. print("As I was going out one day")  
2. print("My head fell off and rolled away,")  
3. print("But when I saw that it was gone,")  
4. print("I picked it up and put it on.")
```

When this program is run, execution starts at line 1 which causes the string, *As I was going out one day*, to be displayed on the output console. Execution then moves sequentially through lines 2, 3 and 4 and finally, the program ends as there are no more lines to execute.

Indentation

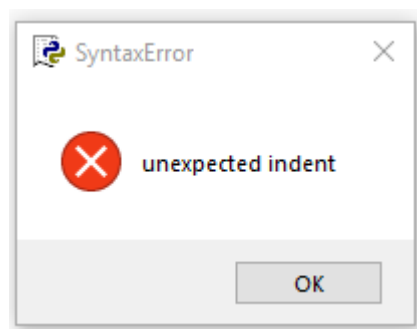
Indentation refers to the empty space(s) at the beginning of a line of code.

Python is very fussy about indentation - try to run the following:

```
1. print("As I was going out one day")
2.  print("My head fell off and rolled away,")
3. print("But when I saw that it was gone,")
4. print("I picked it up and put it on.")
```

Notice that the second line contains a leading space. This is an *indentation error*.

The following syntax error is displayed when a program contains an indentation error.



Syntax Check: Every line of Python code must be properly indented.

Proper indentation means logically related lines (called blocks of code) appear at the same level of indentation.

In the above example indentation is not needed but – as we will see later – it is sometimes necessary to indent code.

- In handout, do **Task 7**.



Programming Exercises

1. Write a program to display your own name and address.
2. Re-arrange the lines of code below into a program that displays the pattern shown on the right. Note that you can use any line as often as you like, but you won't need to use every line.

```
print("##  ##  ##")
print("#####  #####  #####")
print("  ##  ##  ##")
print("  ##  ##  ##")
print("  ##  ##  ##")
print("  ##  ##  ##")
```

```
#####  #####  #####
  ##  ##  ##
#####  #####  #####
  ##  ##  ##  ##
#####  #####  #####
```

3. Reflect on what you have learned about Python so far. On your whiteboard, write down *three* things that Python likes and *three* things that Python does not like.