Chapter 1: Introduction to Python

Contents

Syllabus Learning Outcomes

2.1 Coding Standards

Use common coding standards for programming style (which is dependent on programming language used).

- 2.1.1 Use indentation and white space.
- 2.1.2 Use naming conventions (e.g. meaningful identifier names).
- 2.1.3 Write comments (name of programmer, date written, program description and version bookkeeping/control).

References

Computational Fairy Tales: The Importance of Good Comments: A Tale of the Baker's Apprentice (computationaltales.blogspot.com)

https://www.w3schools.com/python/python_syntax.asp

1 Getting started with Python

1.1 Versions used

The latest version of Python for Windows is 3.10.1. However, we are using version 3.6.4 for the A-Level H2 Computing Syllabus 9569.

1.2 Platforms for Python programming

There are many ways to do programming in Python.

1) One way is to use Python's **Integrated Development and Learning Environment** a.k.a. IDLE. It is installed with Python installation.
2) Creating a Python file and run it using IDLE.
3) Jupyter notebook is another platform for Python programming. The version we will be using is 5.5.0.
4) Google Colaboratory. This is a convenient way to do Python programming, as there is no installation of software required. It is a free application available in Google drive.
5) CodeCollab is an online real-time collaborative code editor and compiler. Its web-based application allows users to collaborate in real-time over the internet. It is a powerful editor that supports many other languages on top of Python. https://codecollab.io/

2 Starting Python with Hello World

We will illustrate how to write Python programs using the first three ways mentioned in section 1.2. These three ways are most essential to A-Level H2 computing.

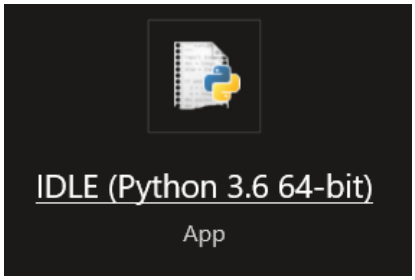
Let's start by printing "Hello World!"

2.1 IDLE

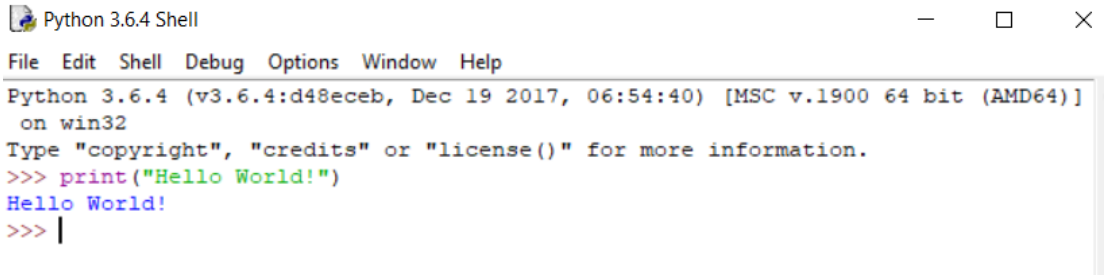
The code is simple as such:

```
print("Hello World!")
```

1. Launch the IDLE app to open a Python shell. Shell is good for experimenting with short expressions or statements to learn new features of the language.



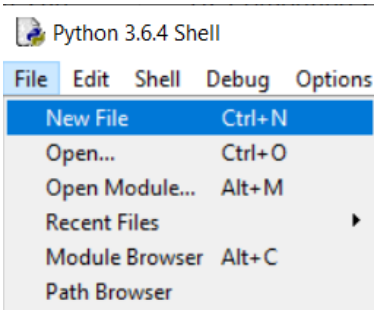
2. Type `print("Hello World!")` in the shell and press Enter.
   The text “Hello World!” will be displayed.



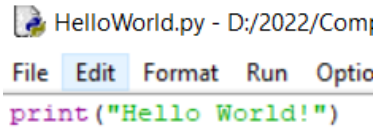
2.2 Python File

When we need to construct complex programs and test them, we can create and save files in program libraries to reuse or share with others.

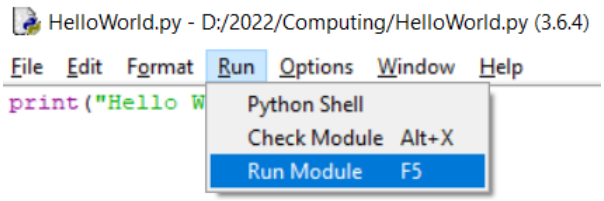
1. Launch IDLE to open a Python shell. Cilck on the File tab and open a new file in the shell.



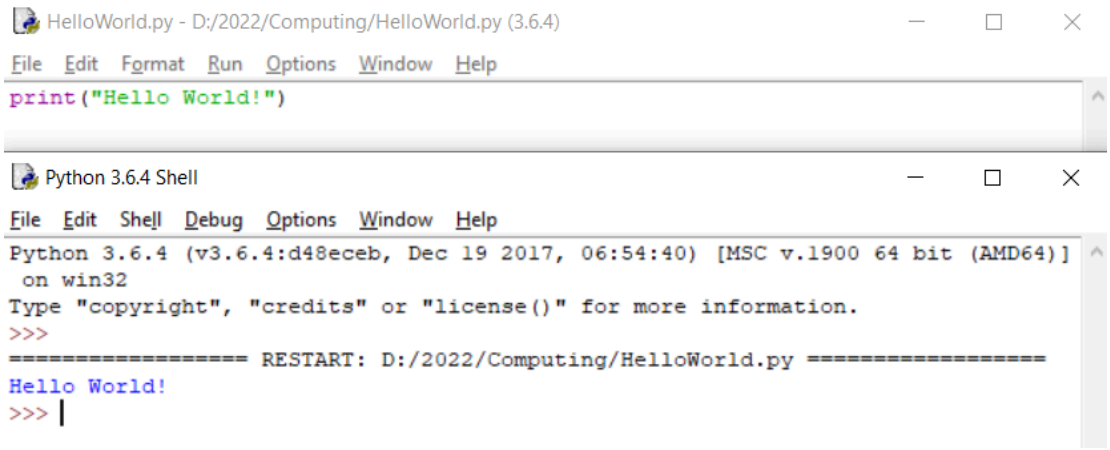
2. Enter your code into the file and save the file using .py extention.



3. Run the Python file by clicking on the Run tab followed by Run Module or using the F5 key.


HelloWorld.py - D:/2022/Computing/HelloWorld.py (3.6.4)
File Edit Format Run Options Window Help
print("Hello W
Python Shell
Check Module Alt+X
Run Module F5

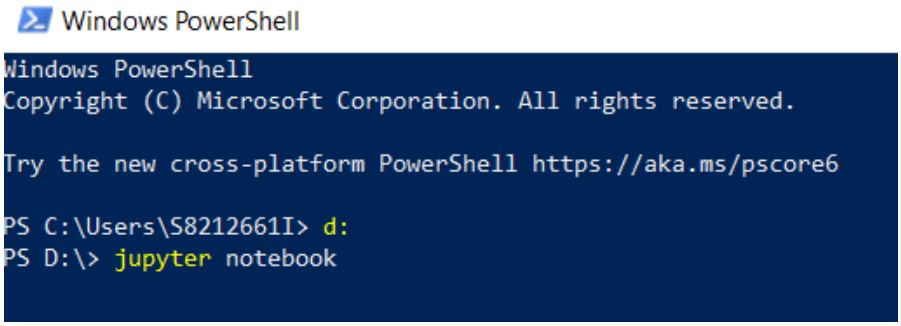
The output will be displayed in the Python shell.



2.3 Using Jupyter Notebook

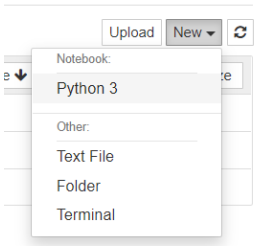
1. Open up Windows Powershell

2. Change to d: drive (or the drive with your working files). Type jupyter notebook and press Enter

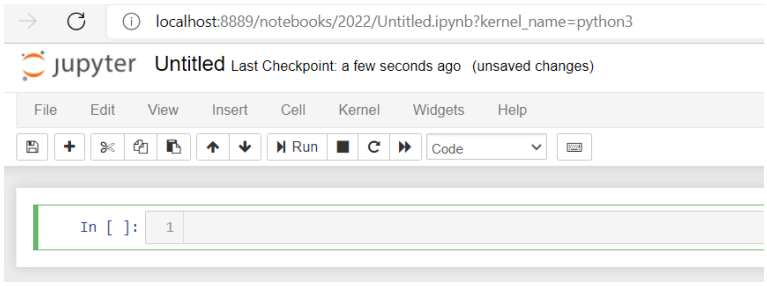


3. Jupyter notebook window will open, displaying the folders/files in the drive.

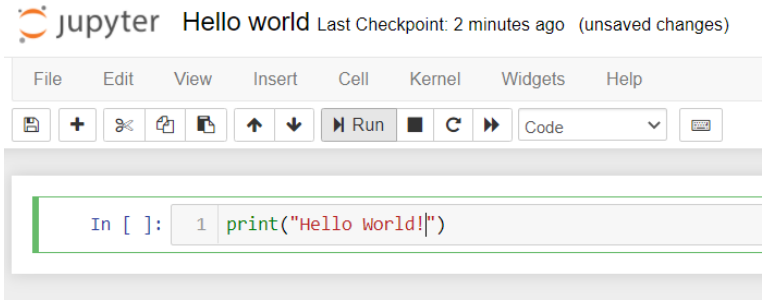
4. Go to a folder of your choice and start a new jupyter notebook file by clicking on the "new" tab on the top right corner.



5. Click on "Python 3" to start a new notebook file.



6. Change the title of the file to "Hello world". Type `print("Hello World!")` and click "Run".



7. "Hello World!" will be displayed.

```
In [1]: 1 print("Hello World!")
Hello World!
```

3 Python Syntax

3.1 Indentation and white space

Indentation refers to the spaces at the beginning of a code line. Tab or 4 spaces per indentation level is commonly used.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

```
1 # Example of using indentation
2
3 if 5 > 2:
4   print("Five is greater than two!")
```

```
Five is greater than two!
```

The example above shows that `print("Five is greater than two!")` is a block of code under the condition `if 5 > 2`.

Python will give an error if indentation is skipped.

```
1 # Example of using indentation
2
3 if 5 > 2:
4 print("Five is greater than two!")
```

```
  File "<ipython-input-5-848779db54fd>", line 4
    print("Five is greater than two!")
        ^
IndentationError: expected an indented block
```

The number of spaces must be the same in the same block of code. Otherwise, Python will give you an error:

```
# Example of using indentation

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

```
  File "<ipython-input-7-604e66f0a2e8>", line 5
    print("Two is smaller than five!")
    ^
IndentationError: unexpected indent
```

That being said, extraneous spaces, tabs and empty lines generally have no effect to the program (except in string literals and for indentation). They should be used adequately and sparingly for readability of the program code.

3.2 Naming conventions

Meaningful identifier names should be used to represent a variable, function, class, module and any other object.

Look at the code below. Are the variable names meaningful?

```
x = "Spiderman"
y = 3

print("{} is {} years old".format(x,y))
```

```
Spiderman is 3 years old
```

Can you suggest variable names that are more meaningful than x and y?

Python Keywords

Python keywords are special words which are reserved and have specific meanings. Python has a set of keywords that cannot be used as variables in programs.

```
from keyword import kwlist

for i in range(0,len(kwlist),3):
    print("{}\t{}\t{}\t".format(kwlist[i], kwlist[i+1], kwlist[i+2]))
```

```
False   None    True
and     as      assert
break   class   continue
def     del     elif
else    except  finally
for     from    global
if      import  in
is      lambda  nonlocal
not     or      pass
raise   return  try
while   with    yield
```

Guidelines for Creating Identifiers

- Identifier name can contain following characters
  - a sequence of letters either in lowercase (a to z) or uppercase (A to Z)
  - digits (0 to 9)
  - underscore (_)
- Identifier name cannot begin with digits
- Special characters are not allowed
- Identifier names are case-sensitive (age and Age are different variables)
- Avoid using Python reserved keywords as identifier names

Best Practices

- Modules should have short, all-lowercase names. Underscores can be used in the module name if it improves readability.
- Class names should normally use the CapWords convention. Eg. MyVariableName
- Function, variable, and method names should be lowercase, with words separated by underscores as necessary to improve readability. Eg. my_variable_name
- Constants are usually defined on a module level and written in all capital letters with underscores separating words. Eg. MY_VARIABLE_NAME
- Declare private identifiers by using the single-underscore `_` as their first letter.
- Don't use double-underscores `__` as the leading and trailing character in an identifier as Python built-in types use this notation.

Which of the following do you think will give an error?

```
1) message = 'hello'
2) _message = 'hello'
3) 1message = 'hello'
4) message? = 'hello'
5) mes-sage = 'hello'
```

3.3 Write comments

Why writing comments is important? Read Appendix C.

Comments are additional text within code that provide explanation. Comments can greatly improve the readability of code by providing insight or summarization. For example, a clear comment before a large block of numeric statements might simply explain "Solve a*x^2 + b*x + c = 0 for x", allowing the user to understand exactly what the following block of code is meant to do.

Comments are also used as part of documentation to help oneself or others understand the program code after a period of time.

Comments start with a #, and Python will render the rest of the line as a comment:

```
1 # This is a comment
2 print("Hello World!")
```

```
Hello World!
```

Compare the two block of codes below:

Code example 1

```
with open("COURSE_INTAKE_BY_YEAR.CSV") as f:
    reader = csv.reader(f)
    header = next(reader)
    intake_data = [row for row in reader]
```

Code example 2

```
with open("COURSE_INTAKE_BY_YEAR.CSV") as f: # "with open" ensures automatic file closure
    reader = csv.reader(f)
    header = next(reader) # Process 1st row as the list header
    intake_data = [row for row in reader] # Iteratively add each subsequent row of data as a sub-list
```

Code example 1 may be obvious to you, but may not be obvious to others. It may be obvious to you now, but does not mean that you will remember it.

Code example 2 is obvious to you and others. It is obvious to you now and in future when you do not remember it.

Be convinced and write comments in your codes to allow yourself and the user to understand exactly what your code meant to do now as well as in the future.

Appendix A – Tips for using IDLE

IDLE is Python's Integrated Development and Learning Environment. It is installed with Python installation.

What does it provide?

- Cross-platform, available on Windows, Unix and MacOS.
- Interactive Python interpreter (shell window)
- Multi-window text editor with syntax highlighting, code completion
- Debugger with breakpoints, stepping, viewing of variables

Read-Eval-Print-Loop (REPL)

The default window of IDLE is an interactive Read-Eval-Print-Loop (REPL) environment, where user can type command directly. The interpreter will

- Reads the command entered by user
- Evaluate and execute the command
- Print the output (if any) to the console
- Loop back and repeat the process

Autocomplete

REPL has autocompletion feature. Use TAB key to activate autocompletion.

Type pr and press TAB key

Copy a line

Use UP arrow key to move up to previous command, press Enter key.

The command at the cursor will be copied

Execute command with conditional statements, loops etc

Type command and press Enter key twice to execute

Appendix B – Jupyter keyboard shortcuts

Reference: https://towardsdatascience.com/jypyter-notebook-shortcuts-bf0101a98330

First, we need to know that they are 2 modes in the Jupyter Notebook app: command mode and edit mode. The left side of the cell is blue when it is in command mode. It is in green when it is in edit mode.

Shortcuts in both modes:

Shift + Enter: run the current cell, select below
Ctrl + Enter: run selected cells
Alt + Enter: run the current cell, insert below
Ctrl + S: save and checkpoint
While in command mode (press Esc to activate):

Enter: take you into edit mode
H: show all shortcuts
Up: select cell above
Down: select cell below
Shift + Up: extend selected cells above
Shift + Down: extend selected cells below
A: insert cell above
B: insert cell below
X: cut selected cells
C: copy selected cells
V: paste cells below
Shift + V: paste cells above
D, D (press the key twice): delete selected cells
Z: undo cell deletion
S: Save and Checkpoint
Y: change the cell type to Code
M: change the cell type to Markdown
P: open the command palette.
While in edit mode (pressEnter to activate)

Esc: take you into command mode
Tab: code completion or indent
Shift + Tab: tooltip
Ctrl +]: indent
Ctrl + [: dedent
Ctrl + A: select all
Ctrl + Z: undo
Ctrl + Shift + Z or Ctrl + Y: redo
Ctrl + Home: go to cell start
Ctrl + End: go to cell end
Ctrl + Left: go one word left

2022/VJC/JC1/H2Computing/9569

Ctrl + Right: go one word right
Ctrl + Shift: + P open the command palette
Down: move cursor down
Up: move cursor up

Appendix C – The Importance of Good Comments: A Tale of the Baker's Apprentice

As part of his world renowned teaching program, the famous baker Breadtista insisted that each apprentice thoroughly document his or her work. Thus, apprentices created a reference that would live beyond their apprenticeship. All recipes, tips, and tricks would be clearly documented. Breadista strongly believed that such a resource provided his apprentices with a significant advantage.

As Breadtista was fond of pointing out, the key to a great recipe was capturing the perfect amount of information. Where other bakers wrote recipe books filled with lists of bland instructions, such as: "Add two eggs and blend well", Breadtista insisted on commenting his recipes. These comments added context and explanation.

Here is an excerpt from Breadtista's famous eight-chili strawberry muffins:
Step 5: Cut the strawberries into cubes of 1cm per side. [This size allows the flavors to blend while preserving the fruit's texture.]
Step 6: Add the strawberries to the dough.
Step 7: Gently stir until the strawberries are mixed into the dough.
Since comments were not part of the instructions themselves, Breadtista clearly denoted his comments with braces [] to avoid confusing other bakers.

Further, Breadtista insisted that all comments provide meaningful information to the recipe itself. He hated recipe books by famous castle chefs that contained thirty page digressions of the author's lifelong quest for the perfect pancake. It was especially egregious if the recipe then produced chewy pancakes. Pancakes were not meant to be chewy, and comments were not meant to be life stories.

Despite his firm belief in the importance of good comments, Breadtista found it difficult to explain this philosophy to each new apprentice. Every year, he heard the same arguments:

- "Why should I comment? The only people reading this are bakers, and they should already know this."
- "I won't forget why I did that."
- "Isn't that step obvious?"

And every year, he repeated the same answers:

- “Just because it is obvious to you, does not mean that it will be obvious to others."
- "Just because it is obvious to you now, as you are writing the recipe, does not mean that you will remember it."
- "If it is so obvious, why did I have to draw you an example on the chalk board? Twice?"

Each year, after lecturing on the importance of clear comments, Breadtista would have at least one apprentice who took the idea too far. For example, one year he had an apprentice who wrote the following instructions for making simple bread:

Step 1: Measure out 1/4 of an ounce of yeast into a small bowl. [We put it into a bowl so that we can add water in step 2 to activate the yeast.]
Step 2: Add two table spoons of warm water to the bowl with the yeast. [We add the water to the bowl in order to activate the yeast.]
Step 3: Wait 5 minutes. [We wait 5 minutes in order to allow the yeast to activate in the water].
Breadtista never made it past step 85 in that recipe. He "accidentally" spilled a full pitcher of water over the apprentice's parchment so as to spare anyone else from reading it.

It took ten years of trial and error, but eventually Breadtista found the perfect way to reinforce the importance of GOOD comments. The first three weeks of each apprenticeship, Breadtista would make each new apprentice work from the worst notes of previous students. Although it meant that he would now hear cries of "WHO WROTE THIS?" and "BUT WHAT DOES IT MEAN?" from the kitchen, this process saved him from having to repeat the same reasoning over and over.

Nothing reinforces the concept of understandable, well documented recipes more than having to read other people recipes.

Computer science concepts as told through fairy tales.
By Jeremy Kubica