Introduction to Python

Overview

- Python 3.
    - Better than Python 2, but Python 2 still in common use for various reasons (compatibility, stubbornness, etc).
    - Most of what you learn here will be transferrable to Python 2 if necessary.
    - We will say no more about this.
- What is a Program?
    - List of instructions that a computer can understand.
- What is a Programming Language?
    - Describes what instructions we can use, how to use them, and what they mean.
    - Lots of languages exist, all with different aims.
        - Specific use-cases vs general; mathematical elegance; ease-of-use; etc.
    - We will use Python because it's general purpose, easy to learn, but still a capable and powerful language.
- Interpreted vs Compiled
    - How do we run our Python code?
        - Feed code to Python interpreter - a program that will run Python code!
    - This makes Python an *interpreted* language.
        - For people to run your program, they need the interpreter and the code.
    - *Compiled* languages:
        - e.g. C++
            - (if anyone asks, Java is a special case - compiles to Java bytecode, which then runs on Java virtual machine - so it's a sort of hybrid)
        - Feed code to compiler - a program that turns code into executables / machine code / something the computer can understand.
        - For people to run your program, only need compiled version - no need for source code or compiler.
    - Compiled languages generally run faster than interpreted languages.
        - This will not matter for our purposes.
- Ok, great, but how do we actually run our Python code? You didn't answer that!
    - Two methods:
    - Command line and text editor.
        - Write code in text editor of choice (e.g. Notepad, Notepad++, gedit, vim, emacs, nano, etc - *but not Microsoft Word*! A word processor is not a text editor).
        - Run python from command line.
        - But this is likely to be confusing if you don't know how to use the command line, and is fiddly either way.
    - Alternative method: use an Integrated Development Environment (IDE).
        - One program containing everything you need to code: text editor + interpreter, with a handy run button!

        - Text editor may provide additional features useful to programming, such as syntax highlighting, code completion, etc.
    - We will be using IDLE, a simple IDE for Python.
        - Make sure everyone has it installed.
        - Show how to open IDLE.
        - Make sure everyone has right version - version number will be displayed!
            - Precise minor version does not matter too much - important thing is that major version is 3!

Hello World

- We are now in a position to write our very first Python code!
- Observe interactive terminal.
    - Executes individual commands immediately.
    - Good for testing what a command does!
- First command we will learn: `print`.
    - Very important command for all our programs: shows output!
    - Syntax: `print(`*`expression`*`)`.
        - We will cover expressions in detail shortly.
        - For now, we will just use text.
        - Text goes in quotes - single or double quotes are both valid, but we recommend double quotes (for style reasons we might go into later).
- So, let's try this in the interactive terminal!
    - (Demonstrate following, and have students do it also.)
    - `print("Hello World!")`
    - Outputs Hello World!
    - Yay!
    - (Make sure everyone managed this without problems, to ensure they have things set up properly.)
- We want to write programs with multiple instructions, though!
    - IDLE: File->New. Opens a blank Python program.
        - (For command line users, just open a text editor.)
    - Write same instruction again into this window.
    - Save as `helloworld.py` (program must be saved for both IDLE and command line users).
        - .py is the extension for Python programs.
    - IDLE: Run->Run Module, or just press F5.
        - (Command line: `cd` to appropriate directory, then `python3 helloworld.py`.)
        - Program runs in terminal window.
- We can add further lines to our program.
    - Demonstrate, and have students try, adding more print commands with more text.
    - (If anyone asks, the newlines can be removed with `print("message", end="")`, but this is fiddly and relies on stuff we haven't taught, so don't bring this up if no-one asks.)

        - (The print buffer is only flushed with newlines, or on other occasions at the interpreter's whim. To ensure flushing, use: `print("message", end="", flush=True)`.)

Comments

- We may want to annotate our code to remind ourselves what it does.
- We can do this with comments.
- Comments start with a hash character.
- Everything after the hash character, up to the end of the line, is ignored by the interpreter - we can put whatever we want!
- Using comments is good, as it makes our code more readable.
    - Also, if you come back to code many weeks / months later, you may not remember how you made it work.
    - Commenting is good practice!
        - You don't have to describe what every line does, though… that should be evident from the code. Only explain what is generally going on.
- Demonstrate having a comment both on its own line, or after one of the print statements.

Next lesson: Expressions and Variables