Module 2; Notebook 1 - Hello World

January 4, 2022

1 Module 2; Notebook 1 - Hello World

A "Hello, World!" program generally is a computer program that outputs or displays the message "Hello, World!"- it is often used to illustrate the basic syntax of a programming language.

It is often the first program written by people learning to code and that's exactly how we'll use it throughout this notebook!

1.1 The Python Syntax

Python follows many of the syntax rules of other programming languages such as Java or C, however there are a few differences. In this section we'll work through the syntax of types, defining variables as well as best practices.

1.1.1 Data types

Python has a variety of variable types ranging from a string to an integer. We'll use the examples below to explore them!

[ ]:
```
type("a")
```

[ ]:
```
type("a4")
```

[ ]:
```
type(4)
```

[ ]:
```
type(4.2)
```

[ ]:
```
type(True)
```

As shown by the examples above, to find the type of something we can call the `type(<argument>)` function on it

Python has many different data types but, for the purpose of this course, we'll stick to the following:

- Text Type: str
- Numeric Types: int & float
- Boolean Type: bool

1.1.2 Operators

An operator is a character that instructs the Python compiler to perform some special operation. For example:

- The opening and closing parentheses `()` are function invocation operators (when placed to the right of a function name).
- The equal sign `=` is the assignment operator, used to assign a value to a variable.
- The plus sign `+` is the string concatenation operator. As you'll see in the next unit, it's also the addition operator when you're working with numbers.
- The number sign `#` is the comment operator, which instructs the compiler to ignore all code or text that comes after it on a line of a code file. We'll explore this later in this notebook.

Python has dozens of operators that perform mathematical, logical, and relational operations in your code.

1.1.3 Variables

Next, we'll take a look at variables - a variable being a named unit of data that is assigned a value. We can call variables to access the data that is assigned to it. We'll explore the declaration of variable as well as the best practices around using them.

[ ]:
```
my_age = 19
type(my_age)
```

As shown above, we have assigned the value of `19` to a variable called `my_age`. Unlike some other programming languages, Python uses dynamic typing meaning we don't need to specify our data type before the variable instantiation e.g. `int my_age = 19`

A couple of notes for best practice here - be sure to always name your variables something relevant and memorable (my_age is more intuative than num!) & common practice is to define variable names either using camel case or using `_` between words.

[ ]:
```
my_age = 19
your_age = my_age
print(your_age)
```

New varibles can be declared using the values of other variables without having any impact on the value. See the example above to see how we can declare a variable using the value of another!

Constants A constant is like a variable but its values won't change during the program. Constants are instantiated at the top of the program in capital letters like below:

[ ]:
```
PI = 3.14 # Constant definition

radius = 4
area = PI * (radius ** 2) # ** is the operator for 'to the power of'
print(area)
```

1.1.4 Comments

One of the most important things in programming, is ensuring your code is clear and easy to understand - the tip earlier about well named variables is one way to aid this. Here, we'll look at another way to improve code readability - using comments!

Comments are a vital way to help explain your code to others in your team or even yourself when you come back to old code. They can be used to explain individual functions (sections of code with a particular functionality) or just a single line where it may be difficult to see what is going on underneath the hood.

[ ]:
```
my_age = 19 #Assigning the value 19 to the variable `my_age`
```

Yes the above example is borderline redundant, however it emphasises the point of how comments can be used to aid understanding of code. If you struggle to explain your code with a simple comment, you may need to re-evaluate and simplify it.

1.1.5 Input / Output

Often, programs interact with the user through either writing something to the terminal or asking for an input. We'll be looking at the way to ingest an input from the console and then printing it back.

The `print()` function is used to output thigns to the console and has been teased a few times over the course of the first couple of notebooks, but this time we'll walkthrough using it!

[ ]:
```
print("This will be printed below")
```

[ ]:
```
print(4)
```

The print function outputs the argument to the console, it can print integers, float, strings etc.

As for passing in an input into a Python program, that can be done using the `input()` function - as shown below!

[ ]:
```
print("Whats your age?")
user_age = input()
```

[ ]:
```
print(user_age)
```

[ ]:
```
number = input("Please input a random number!") #Asks the user for a number and␣
↪saves it in the variable named number
```

That draws us to a close for the basics - to consolidate what you've learnt, spend a few minutes in the Python IDLE (this is the interactive development environment installed with Python - can be found by searching your PC) playing around with the functions and ideas that have been introduced!