Saying Hello to Python

In order to work with Python you have to start a program that understand Python. You can either use a command prompt or IDLE (or IdleX, which is an extended version of IDLE).

## Starting IDLE or IdleX

Just click the program from the Start Menu, or if you can’t find it press the Windows Key, then type IDLE and press return.

## Starting from a Command Prompt

Launch a Command Prompt window. (Windows Key, then enter cmd.exe and press return.) In the boxes below are the input and output from the Python interpreter. Red text is what you type in at the keyboard (input). Blue text is what Python is writing on the screen (output).

C:\users\andy> python

Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>

# Let’s write some Python!

Python is now ready for you to tell it what to do. This is known as interactive mode. Interactive mode allows you to type a command to the Python interpreter and when you press Return it will immediately evaluate, or execute, it. The three greater-than symbols “>>>” means that Python is waiting for you to type something at the interpreter.

Let’s try it out! Type print(“Hello World!”) then press return and you should see the following:

>>> print(“Hello World!”)

Hello World!

Python is *really* picky. You must use exactly the right upper case or lower case letters. You can put extra spaces sometimes if you want to:

>>> print("Hello")

Hello

>>> print ("Hello")

Hello

>>> print( "Hello" )

Hello

>>> print ( "Hello" )

Hello

But if you miss out a parenthesis (bracket) or put an upper-case P in Print then Python prints errors like this:

>>> Print("Hello")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'Print' is not defined

>>> Print "Hello"

File "<stdin>", line 1

Print "Hello"

^

SyntaxError: invalid syntax

These errors can look intimidating and incomprehensible at first, but don’t worry, you won’t be able to damage the equipment by typing the wrong thing. Well, not yet!

Everything in those errors does *mean* something and it can be useful, but for now we’ll ignore it. If you see a message like this and you can’t spot what you’ve done wrong, then just ask.

Our simple print(“Hello”) command to Python invokes a function passing some parameters.

The function is print and there’s just one parameter: the string Hello.

Everything we type at Python is literally a command, as programmers, we’re like an Army Drill Sergeant barking orders at soldiers. Just like soldiers, Python obeys without question!

Let’s try some maths.

>>> print(1 + 2 + 3)

6

>>> print(9-7)

2

>>> print(123456789 + 987654321)

1111111110

>>> print(8 \* 7)

56

>>> print(100 / 5)

20.0

You can do all kinds of basic and advanced mathematics in Python, just by asking it a question.

The print function handled output, but what about input? Input is putting data in to the program, asking a question, for example.

To do this, there’s another function called input. Just like print, it’s a function, but instead of printing things out, this collects input from the user. It takes one parameter, which is some text to print to prompt the user to enter some information.

Let’s see it in action:

>>> input("What is your name? ")

What is your name? Andy

'Andy'

A couple of interesting things happened here:

Notice how we were able to type even though the >>> prompt wasn’t there. This is because it wasn’t the Python interpreter waiting for input, but it was our call to the input() function.

Also, Python decided to print out the name Andy without being asked to. It even put single quotes around it, for some reason. It did this because we asked Python to ask for input, but we didn’t tell it what to do with it. When you don’t tell Python what to do with something you’ve asked it to capture, it will just print it out for you.

Let’s get Python to greet us by our name. So instead of saying Hello, it will say Hello Andy. To do this we need to save the captured input in a variable. Variable is a word that means “something that can change”.

To save the captured input, we’re going to use some more mathematical notation:

>>> name = input("What is your name? ")

What is your name? Andy

Notice this time it didn’t print out the name. That’s because Python has taken the input and put it in the variable called *name*. You can think of it as if Python had Andy written down on a bit of paper, which it then put in a box called name. Now, every time we use the word name in our program, it will substitute it for the item in the box.

Now, let’s tell Python to print out our greeting along with the contents of the name variable:

>>> print("Hello", name)

Hello Andy

Again, Python’s done some clever stuff here.

1. We’ve called the print function, as before.
2. There’s now two parameters to the print function. Both the parameters are between the parenthesis. Each parameter is separated by a comma.
3. The first parameter Hello stays in double quotes, but the second one doesn’t.

Whenever Python finds a word it will look to see if it’s a function or a variable first. The double quotes around Hello are telling Python “I don’t want you to look for a function or variable, I want you to take everything between the quotes as the literal data.

Play around with this and you’ll see what’s going on:

>>> print("Hello", "name")

Hello name

By putting double quotes around name, we’ve told Python not to look for a variable called name, but to literally print the letters n, a, m and e.

>>> print(Hello, name)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'Hello' is not defined

Uh oh, a weird error! Just look at the last bit though. Python is telling us that it doesn’t know of anything called Hello, either a variable or function.

See if you can work out what’s going on here to make it work:

>>> Hello = "Good-bye"

>>> print(Hello, name)

Good-bye Andy

What we did here was create a new variable called Hello, just like we did with name, but set the value of Hello to “Good-bye”.

Don’t worry if you didn’t quite follow it, the key lesson here is that Python is pretty stupid. It will do *whatever* you tell it to do, no matter how mad it is!

You might be wondering if you can respond to an input function using a number. The answer is that you can, but before you can do maths on it, you must tell Python that it’s a number.

>>> word = input(“Enter a number: “)

Enter a number: 5

>>> word + 2

Traceback (most recent call last):

File "<pyshell#46>", line 1, in <module>

word + 2

TypeError: Can't convert 'int' object to str implicitly

Here, Python is telling us that even though we typed “5”, it didn’t realise that it’s a number (there is a good reason for this, but that’s for another lesson!). To tell Python that it’s a number, we use a third function, called int.

>>> word = input("Enter a number: ")

Enter a number: 5

>>> number = int(word)

>>> number + 2

7

It’s a bit of a pain having to use two variables each time you want to input a number, so luckily Python allows you to combine functions together, like this:

>>> number = int(input("Enter a number: "))

Enter a number: 5

>>> number + 1

6

Read the top line carefully and make sure you understand it. The *output* from the input function is being passed in to the int function straight away, without going in to a variable first. Then the output from int is being saved in the number variable.

Finally, let’s merge some maths, variables and printing out (I’ve put some blank lines in to make it more readable, you don’t have to type those), also put in any numbers you like.

>>> this\_year = int(input("What is this year? "))

What is this year? 2016

>>>

>>> print("This year is", this\_year)

This year is 2016

>>>

>>> birth\_year = int(input("What year were you born? "))

What year were you born? 1885

>>>

>>> print("I was born in", birth\_year)

I was born in 1885

>>>

>>> print("I am", this\_year - birth\_year, "years old this year!")

I am 131 years old this year!I am 131 years old this year!

>>>

>>> print("I was born in", birth\_year, "and this year is", this\_year, "which means I am", this\_year - birth\_year, "years old this year!")

I was born in 1885 and this year is 2016 which means I am 131 years old this year!

Using input and print you can create your own questionnaires, that include maths, in Python!

# Exercise

Create a simple questionnaire that asks for 2 numbers and then prints out the answer to:

1. Adding the numbers.
2. Subtracting one number from the other.
3. Multiplying the two numbers together.
4. Dividing one number by the other.

Here is some suggested output from your program:

Enter first number: 4

Enter second number: 5

Adding 4 and 5 equals 9

Subtracting 4 from 5 equals 1

Multiplying 4 and 5 equals 20

Dividing 4 by 5 equals 0.8