

## INTRODUCTION

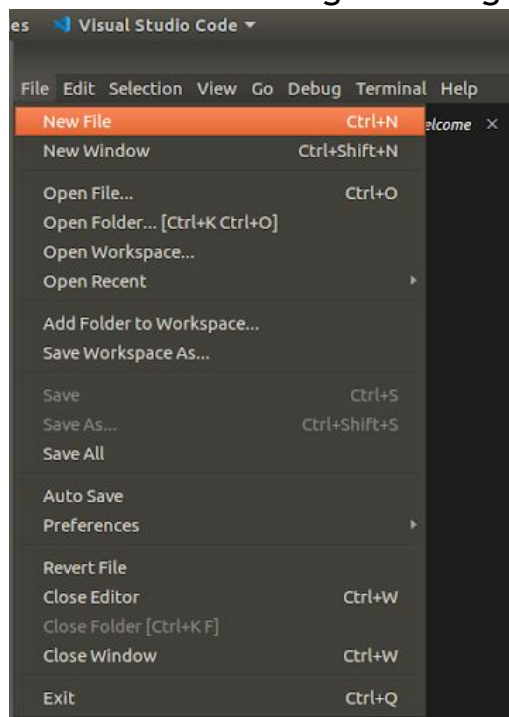
Printing “Hello World” is traditionally the first thing anyone does when learning a new programming language.

In this project guide we’ll take you through using the print statement in Python, along with loops and variables to be able to make your programmes talk. And to print out pictures of cake!

```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$  
^ ^ ^ ^ ^ ^  
I I I I I I  
#####  
-----  
#####  
-----  
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

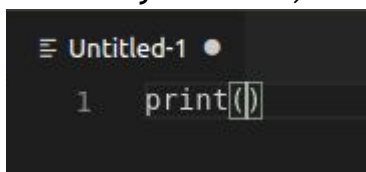
## Step One: Getting started

- ✓ Open VS Code from the icon in your side bar.
- ✓ We need a new file to begin editing. Start one from the 'File'



menu.

- ✓ In your new, untitled file, begin typing print(

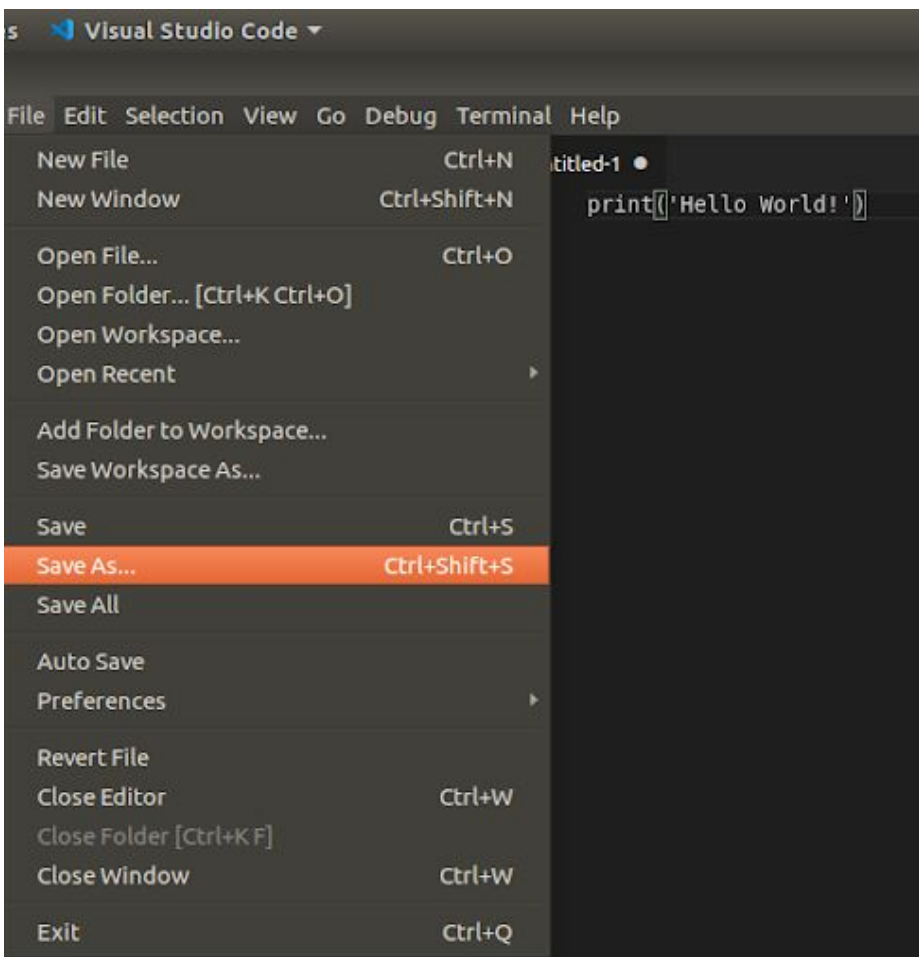


- ✓ Notice how even though you only typed print( the programme has added the second )

- ✓ Add the rest of our print statement to say 'Hello World!'

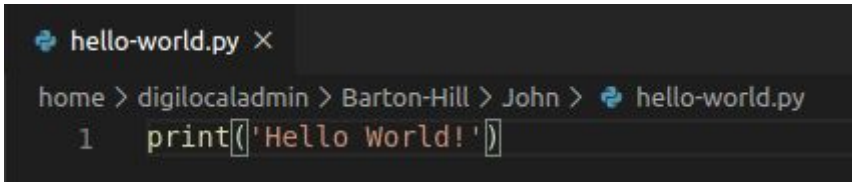
```
Untitled-1 •
1  print('Hello World!')
```

- ✓ We haven't yet told VS Code that we're using python. VS Code can work with lots of different languages. The easiest way to let it know we're using python is to save the file with a .py ending, so let's do that now.



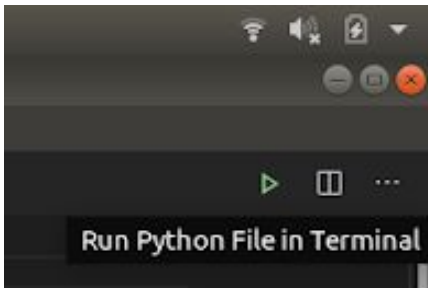
- ✓ Call your programme something like hello-world.py it's a good idea not to use spaces in file names for programming as it can cause problems later on.

- ✓ You'll notice some important changes to VS Code. First your editing window now has the filename, and a python logo, the full directory and filename is shown just above the editing window, and your code has been coloured in to show important features.

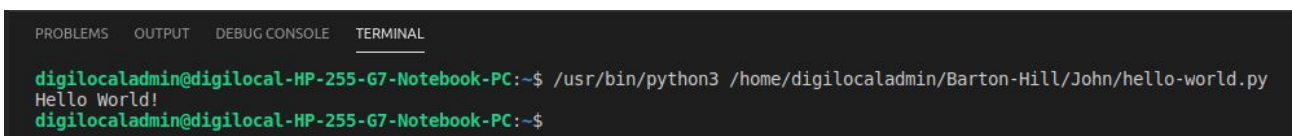


```
hello-world.py X
home > digilocaladmin > Barton-Hill > John > hello-world.py
1 print('Hello World!')
```

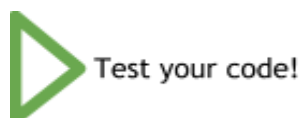
- ✓ You also have a small green 'play' triangle in the top right of VS Code.



- ✓ If you click this you should see your code run in the window at the bottom of VS Code.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$ /usr/bin/python3 /home/digilocaladmin/Barton-Hill/John/hello-world.py
Hello World!
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```



- ✓ You can also right-click anywhere in your editor window and select the 'Run python file in terminal' option.

## Step Two: Joining things up

- ✓ Quite often in a programme, we want to join two (or more) things together when we print something out. Python gives us lots of options for this.
- ✓ Add a new line of code, and begin typing `print(` again. Notice the large window pop up with helpful information about how and what you can print.

```
1 print('Hello World!')
2 print()
```

`print(*values: object, sep: Text=..., end: Text=..., file: Optional[_Writer]=..., flush: bool=...) -> None`

param \*values: object

`print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)`  
Prints the values to a stream, or to `sys.stdout` by default.

Optional keyword arguments:

`file`: a file-like object (stream); defaults to the current `sys.stdout`.

`sep`: string inserted between values, default a space.

`end`: string appended after the last value, default a newline.

`flush`: whether to forcibly flush the stream.

- ✓ We can ignore most of this for now, but later on you may find this useful. For now, let's complete our line of code.

```
1 print('Hello World!')
2 print(['Hello', 'World'])
```



Test your code!

- ✓ You should see the output below.

```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$  
Hello World!  
Hello World  
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

- ✓ The comma is used to tell python we're printing two separate things. In this case they were text strings. Python helpfully put a space between them.
- ✓ We can also tell python to join our strings together. This is called concatenation in programming. It's just a long word for 'joined together'. Change your comma to a +.

```
1 print('Hello World!')  
2 print(['Hello'+'World'])
```



Test your code!

- ✓ Now that we've told python these two strings should be joined together, it's removed the space between them.

```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$  
Hello World!  
HelloWorld  
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

- ✓ We can also perform simple operations inside a print statement. Not the medical kind, the mathematical kind!
- ✓ Add the code below to your programme, what do you think will be the output?

```
1 print('Hello World!')  
2 print('Hello'+'World')  
3 print(['Hello '*5])
```



# Hello World!

MAKING OUR DIGITAL FUTURE



Test your code!



- ✓ You should have 5 hello's.

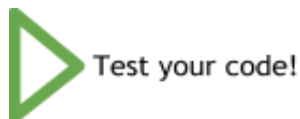
```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$  
Hello World!  
HelloWorld  
Hello Hello Hello Hello Hello  
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

- ✓ If they're all clumped together try editing your code to include the space.

- ✓ We can try other mathematical operators.

```
1 print('Hello World!')  
2 print('Hello'+'World')  
3 print('Hello '*5)  
4 print(['World'-5])
```

- ✓ What do you think will happen here?



- ✓ We have an error!

```
Hello World!  
HelloWorld  
Hello Hello Hello Hello Hello  
Traceback (most recent call last):  
  File "/home/digilocaladmin/Barton-Hill/John/hello-world.py", line 4, in <module>  
    print('World'-5)  
TypeError: unsupported operand type(s) for -: 'str' and 'int'  
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

- ✓ Python has told us that something went wrong on line 4, and it has printed out the bit that went wrong.
- ✓ It's also told us what when wrong, but you need to understand a little about computers to make sense of the message. Basically, you can't



take a number away from a word. 'int' is short for integer, or a whole number, and 'str' is short for string, or a list of letters.

## Step Three: Longer text outputs

- ✓ So far we have printed single words, and how to repeat them. What if we want to print a longer message, such as this paragraph or some instructions for a game?
- ✓ Until now we've used the single quote mark to contain our printed string. If we use 3 quote marks, python will print out longer passages, including line breaks. Enter the code below.

```
1 print('Hello World!')
2 print('Hello'+ 'World')
3 print('Hello '*5)
4 print("""Here are the instructions for my game:
5 1) you can type across
6 2) many lines but always finish
7 3) with 3 quote marks, ok""")
```



Test your code!

- ✓ You should get something like the output below (depending on what you typed).

```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
Hello World!
HelloWorld
Hello Hello Hello Hello Hello
Here are the instructions for my game:
1) you can type across
2) many lines but always finish
3) with 3 quote marks, ok
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

- Using this new print statement, we can create ASCII art, try making the dog and tower below, or something of your own design!

```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
Here is a dog:
0 _ _ /
  II II
and a tower
  / \
 /   \
I # # I
I # # I
I     I
I   H I
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

- Note: for the backslashes ‘\’ you will need a space after them, otherwise python will think they are special characters and won’t print things out properly.

## Step Four: Loopy text!

- So we can print simple strings, and we can print multiple strings, and we can print long strings. But what if we want print lots of strings? We can use loops for this.
- We’ll print an ASCII cake. Either delete your code or start a new file:

```
home > digilocaladmin > Barton-Hill > John > + ascii-cake.py > ...
1 print('^ ' * 6)
2 print('I '*6)
```

- Notice that python doesn’t mind spaces within the lines of code. Both the above lines work perfectly, but it’s a little easier to read the upper one with extra spaces. So let’s tidy our code up to put a space between everything.

```
home > digilocaladmin > Barton-Hill > John > + ascii-cake.py > ...
1  print('^ ' * 6)
2  print('I ' * 6)
```

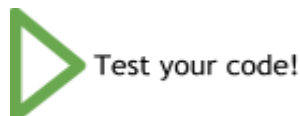
- Next we add our loop. This is a simple repeat loop like we have used in Scratch.

```
home > digilocaladmin > Barton-Hill > John > + ascii-cake.py > ...
1  print('^ ' * 6)
2  print('I ' * 6)
3  for i in range(2):
```

- We've actually created a new variable called 'i' to keep track of how many loops we've done. And we are going to loop through twice.

```
home > digilocaladmin > Barton-Hill > John > + ascii-cake.py > ...
1  print('^ ' * 6)
2  print('I ' * 6)
3  for i in range(2):
4      print('#' * 11)
5      print['-' * 11]
```

- Helpfully, VS Code shows that the last two print statements are inside a loop by putting a short line next to them as a group.



- We should have a small cake!

```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
^ ^ ^ ^ ^ ^
I I I I I I
#####
-----
#####
-----
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

- ✓ We can add another line to our loop to see what's actually happening.
- ✓ We've also added a line at the end to show that we've finished the loop and are back in the main programme.

```
1 print('^' * 6)
2 print('I ' * 6)
3 for i in range(2):
4     print('#' * 11)
5     print('-' * 11)
6     print(i)
7 print(['Cake!'])
```



Test your code!

- ✓ Notice that the computer begins counting at 0, not at 1. This becomes quite important later!

```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
^ ^ ^ ^ ^ ^
I I I I I I
#####
-----
0
#####
-----
1
Cake!
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

- ✓ Adding print statements is a great way to check that your programme is behaving as you expect it to.



# Hello World!

MAKING OUR DIGITAL FUTURE

## Challenge time!

What other patterns and designs can you make?