

Programming for Data Science G (11521 Online & On-campus)

Week 2 Tutorial

Operators and basic Expressions

Objectives

- To use Visual Studio/PyCharm/Spyder to create a new Python project
- To practise basic programs, operators, and expressions
- To implement a short program based on what you have practised

Software

You can skip this step if:

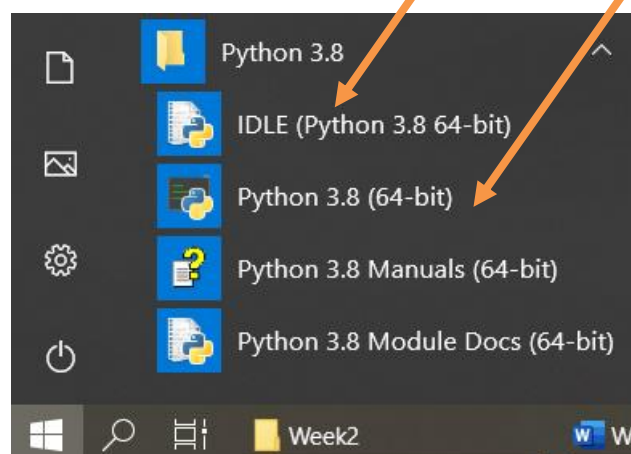
- You already have software installed in your own computer, or
- You are doing this tutorial in **Computer Lab 6B2** at UC, or
- (Recommended, Internet access required) You are using software at UC by logging to **UC Student Virtual Desktop** <https://frame.nutanix.com/university-of-canberra/ditm/uc-remote-access-student/launchpad/uc-virtual-desktop-student>.
Visual Studio Professional 2019, JetBrains PyCharm Professional 2019, Anaconda and Spyder are available at UC.

For your own computer:

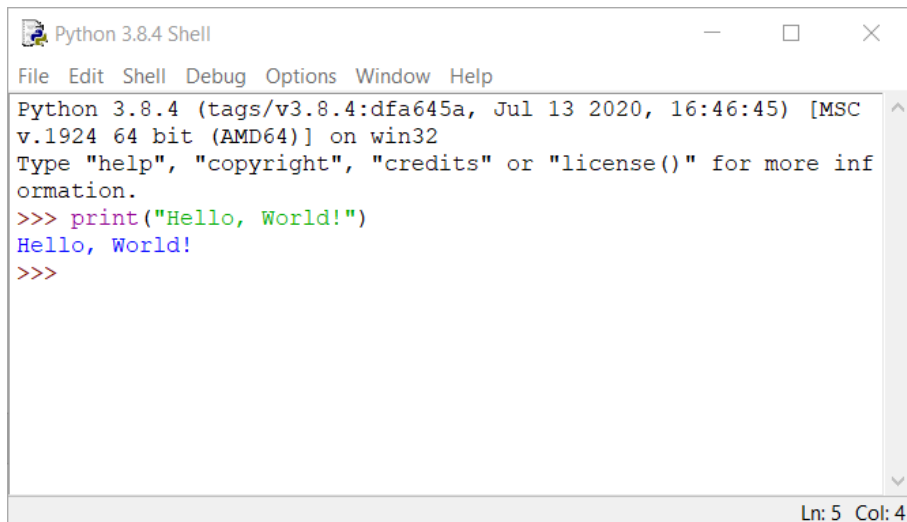
- Download the latest version of Python here <https://www.python.org/>
- Download either **PyCharm Community** (<https://www.jetbrains.com/pycharm/>) or **Visual Studio Community** (<https://visualstudio.microsoft.com/>) or **Anaconda with Spyder** (<https://www.anaconda.com/products/individual#download-section>).

Writing and running a Python program

- You can write and run Python code directly in **Python Shell** or **Python Interpreter** window. Click on the Windows icon in the bottom-left corner to find them (the version number may not be the same in your computer).



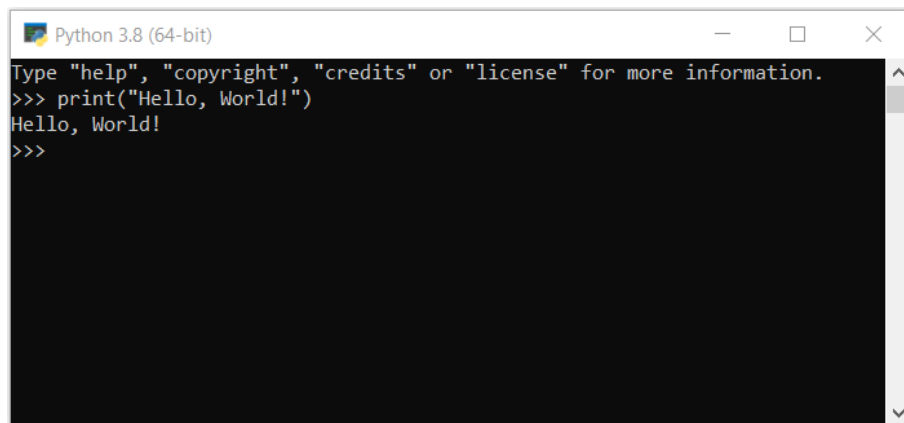
- For example, type the first Python code: `print("Hello, World!")` and press Enter key, Python executes this line and outputs to the shell.



```
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:46:45) [MSC
v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
ormation.
>>> print("Hello, World!")
Hello, World!
>>>
```

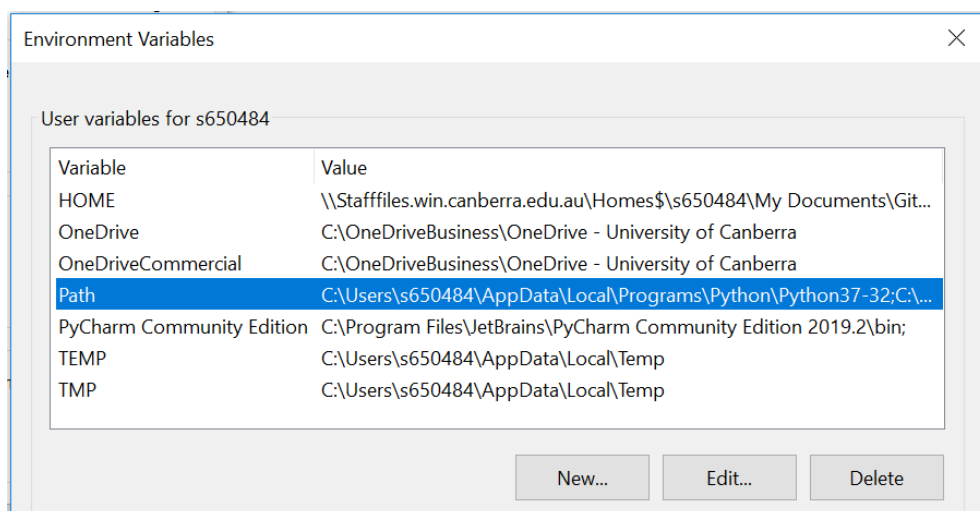
Ln: 5 Col: 4

- The same output if you use Python Interpreter:



```
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>>
```

- If you want to write Python code in a file, this file should have **.py** extension and you can run your program in this file from the **Command Prompt** window (you need to set a path to Python when you install Python as seen below). However, you cannot do this in computers in the lab since you cannot modify environment variables. Below is an example from my laptop.



- For example, you create a file **week2.py** in the **PFDS** folder in C:\ drive and edit this file to add `print("Hello, World!")` and save. Now you can type command **python week2.py** to run your program as seen below.

```

C:\>cd PFDS

C:\PFDS>python week2.py
Hello, World!

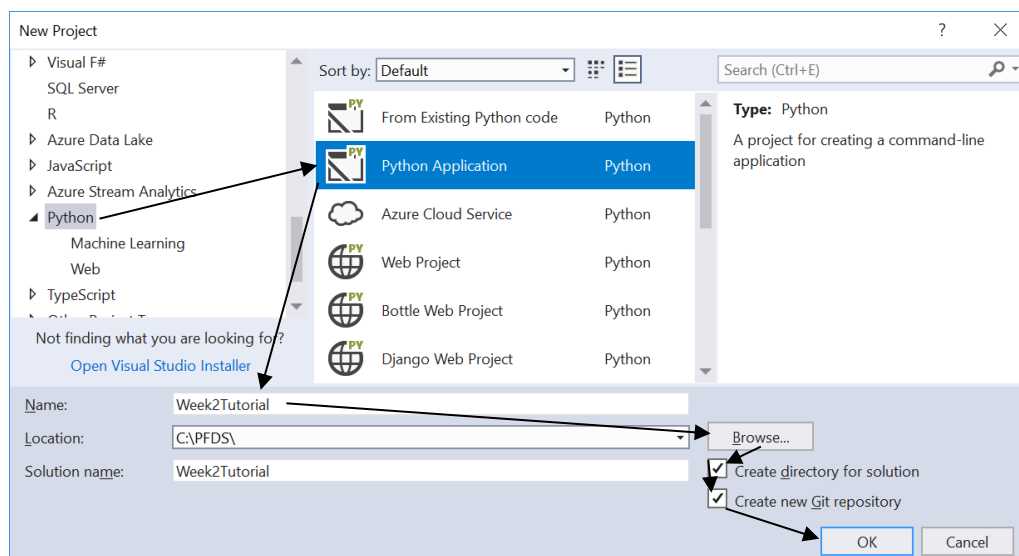
C:\PFDS>

```

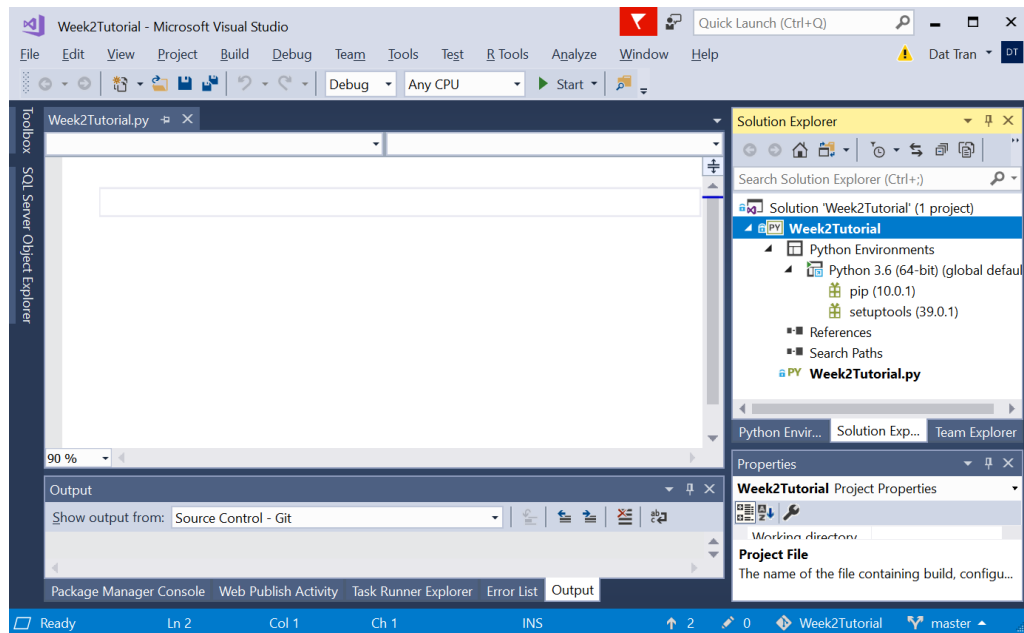
- You can continue editing this week2.py file with any text editor, save the file then run it on this Command Prompt window. However, it would be easier and faster if you use **PyCharm** or **Visual Studio** or **Spyder** to develop your Python projects. These tools have intelligent code editor that helps you save time for writing Python.

Use Visual Studio to Create a New Python Project (you can do this step with PyCharm or Spyder)

- If you haven't installed Visual Studio (or PyCharm or Spyder) on your own computer, please login to UC Student Virtual Desktop <https://frame.nutanix.com/university-of-canberra/ditm/uc-remote-access-student/launchpad/uc-virtual-desktop-student>
- Open Visual Studio. Click the link *Continue without code* in bottom-right corner)
- Click on File > New > Project
- On the New Project dialog, select **Python** and **Python Application**. Enter **Week2Tutorial** to the **Name** text box. Click **Browse** button and browse to a folder in your drive. Select **Create directory for solution** and **Create new Git repository**. Finally, click OK button to create this new project.



- The following project opens in Visual Studio. Have a look at Solution Explorer, Python Environment, and Team Explorer.



- The **Week2Tutorial.py** file is empty and opening for editing. You will write Python code in this file.

For instructions to create a python project in Spyder, follow this: [Link](#).

Comments, Standard Input and Output

- **Example 1:** Enter the following to Week2Tutorial.py (Note: **all lines must have the same indentation**)

```
"""
Programming for Data Science
Week 2 Tutorial
"""

#Output
print("Hello, World!")

#Input
x = input("Enter a number: ")

print("You have entered " + x)
```

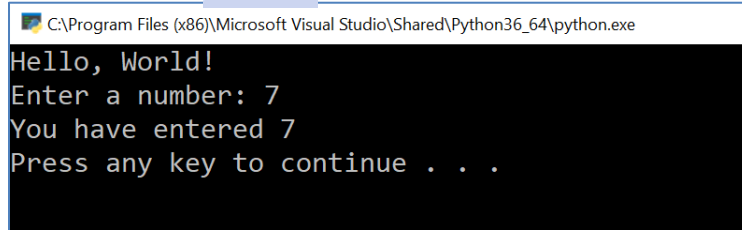
The first 4 lines show how to write a multi-line comment. The next line **#Output** is a single line comment. Python will ignore all comments.

The `print("Hello, World!")` function will output **Hello, World!** when you run the program.

The next line **#Input** is a single comment. The following line `x = input("Enter a number: ")` is to prompt the user to enter a number and then gets the number and assigns it to variable `x`.

The last line `print("You have entered " + x)` is to output the string and the number.

To run this program, click  Start on Toolbar. The output is as follows:



Constants and Strings

Literal Constants: 5, 3.4, 'data science'

Integers: 2, 5, 23456, -23

Floats (floating point numbers): 1.3, -2.678

String in single quote: 'data science'

String in double quote: "data science"

Strings in double quotes work exactly the same way as strings in single quotes

String in triple quote: for multi-line strings

```
'''This is a multi-line string.
```

```
    'Single quote in triple quote'
```

```
    "Double quote in triple quote"
```

```
'''
```

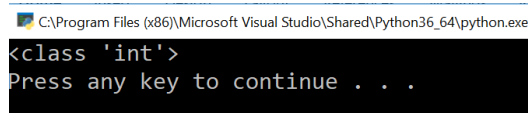
Raw strings: specified by prefixing r or R to the string: `print(r"Data Science")`

Variables

You use variables when you need to store a data value. For example, in the Python statement: `x = 7`, `x` is a variable, and 7 is a data value that is assigned to `x`. Python does not need a command like `int x = 7`; to declare a variable as seen in Java or C#.

- **Example 2:** Enter the following Python code to `Week2Tutorial.py` and run it

```
x = 5 #x is of type int
print(type(x)) #output type of x
```



The `type(x)` function returns type of `x`.

- **Example 3:** Enter the following Python code to `Week2Tutorial.py` and run it

```
y = 7.9 #y is of type float
print(type(y))
```

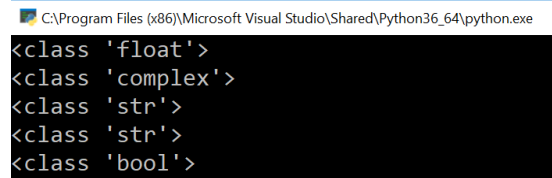
```
z = 1 + 3j #z is of type complex and j is imaginary
print(type(z))
```

```
u = "Data Science" #u is of type str (string)
print(type(u))
```

```
v = 'Data Science' #v is of type str (string), too
print(type(u))
```

```
t = True #t is of type bool
```

```
print(type(t))
```



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
<class 'float'>
<class 'complex'>
<class 'str'>
<class 'str'>
<class 'bool'>
```

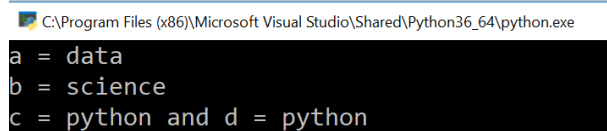
Note "Data Science" is the same as 'Data Science' in Python. Both are strings.

- **Example 4: Multiple variables.** You can assign data values to multiple variables.

Enter the following Python code to Week2Tutorial.py and run it.

```
a, b = 'data', 'science'
print("a = " + a)
print("b = " + b)
```

```
c = d = 'python'
print('c = ' + c + ' and d = ' + d)
```



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
a = data
b = science
c = python and d = python
```

Rules for variable names:

- Variables should have more descriptive names, e.g., day, age, line, etc.
- A variable name cannot start with a digit. It must start with an alphabet letter or the underscore (_) character
- A variable name is case sensitive and can contain a-z, A-Z, 0-9, and _ only.
- Examples of **invalid** variable name: 7day, #myId, first name, last-name.
- Variables firstName and firstname are different (case sensitive).

Type conversion

You can use int(), float(), bool(), complex(), and str() to convert type of a variable.

- **Example 5:** Enter the following Python code to Week2Tutorial.py and run it.

```
a = 5 #a = 5
print(a)
b = float(a) #b = 5.0
print(b)
c = bool(b) #c = True
print(c)
d = str(c) #d = 'True'
print(d)
e = complex(a) #e = 5 + 0j
print(e)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
5
5.0
True
True
(5+0j)
Press any key to continue . . .
```

Output data values

If data value is a string, you can use the operator `+` to concatenate it with other strings. For example, the following will output **Unit name = Data Science** since `name` is of type `str`.

```
name = 'Data Science'
print("Unit name = " + name)
```

If data value is not string, you can use `str()` function to convert it to string then use the `+` operator for concatenation. For example,

```
myId = 123456
print('My ID is ' + str(myId))
```

Alternatively, you can use **f-strings** (available with Python 3.6 or newer) if data values are not strings. For example, the following code will output **My ID is 123456** and note `myId` is of type `int`.

```
myId = 123456
print(f'My ID is {myId}')
```

- **Example 6:** Enter the following Python code to `Week2Tutorial.py` and run it.

```
name = 'Data Science'
print("Unit name = " + name)
myId = 123456
print('My ID is ' + str(myId))
print(f'My ID is {myId}')
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Unit name = Data Science
My ID is 123456
My ID is 123456
Press any key to continue . . .
```

- **Example 7: Multi-line string.** Use new line character `\n` or triple quote (`"""` or `'''`). Enter the following Python code to `Week2Tutorial.py` and run it.

```
print('Programming for Data Science\nWeek 2 Tutorial')
```

```
print("""Programming for Data Science
Week 2 Tutorial
""")
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Programming for Data Science
Week 2 Tutorial
Programming for Data Science
Week 2 Tutorial
```

Operators

- **Example 8:** Enter the following Python code to Week2Tutorial.py and run it.

```
s1 = 'abc'
s2 = 'def'
print(f's1 = {s1}')
print(f's2 = {s2}')
print(f's1 + s2 = {s1 + s2}')
print(f's1 * 2 = {s1 * 2}')
print('\n')
x1 = 5
x2 = 2
print(f'x1 = {x1}, x2 = {x2}')
x3 = x1 + x2
print(f'x1 + x2 = {x3}')
x3 = x1 - x2
print(f'x1 - x2 = {x3}')
x3 = x1 * x2
print(f'x1 * x2 = {x3}')
x3 = x1 ** x2 #power
print(f'x1 ** x2 = {x3}')
x3 = x1 / x2
print(f'x1 / x2 = {x3}')
x3 = x1 // x2 #divide and floor
print(f'x1 // x2 = {x3}')
x3 = x1 % x2 #modulo
print(f'x1 % x2 = {x3}')
```

C:\Program Files (x86)\Microsoft Visual Studio\Sha

```
s1 = abc
s2 = def
s1 + s2 = abcdef
s1 * 2 = abcabc
```

```
x1 = 5, x2 = 2
x1 + x2 = 7
x1 - x2 = 3
x1 * x2 = 10
x1 ** x2 = 25
x1 / x2 = 2.5
x1 // x2 = 2
x1 % x2 = 1
```

- **Example 9:** Enter the following Python code to Week2Tutorial.py and run it.

```
w = 5
print(f'w = {w}')
w += 2
print(f'w += 2 -> w = {w}')
w -= 2
print(f'w -= 2 -> w = {w}')
w *= 2
print(f'w *= 2 -> w = {w}')
w /= 2
print(f'w /= 2 -> w = {w}')
w %= 2
print(f'w %= 2 -> w = {w}')
```

```
w = 5
w += 2 -> w = 7
w -= 2 -> w = 5
w *= 2 -> w = 10
w /= 2 -> w = 5.0
w %= 2 -> w = 1.0
```

Question: Write a Python program that calculates the earnings of a carpenter. The base salary is \$1200 a week. For every door that the carpenter sells, he/she will get \$15 extra as bonus. In addition, at the end of each month, the carpenter gets 2% commission on the month's sale. Calculate the carpenter's total earnings in this month if he/she sells 35 doors and each door costs \$150.