1. Excel to Python

Excel to Python
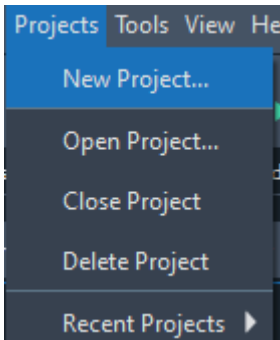
"Code is like humor. When you have to explain it, it's bad." - Cory House

Before we can transition from excel to Python we have to learn some more fundamentals about Python
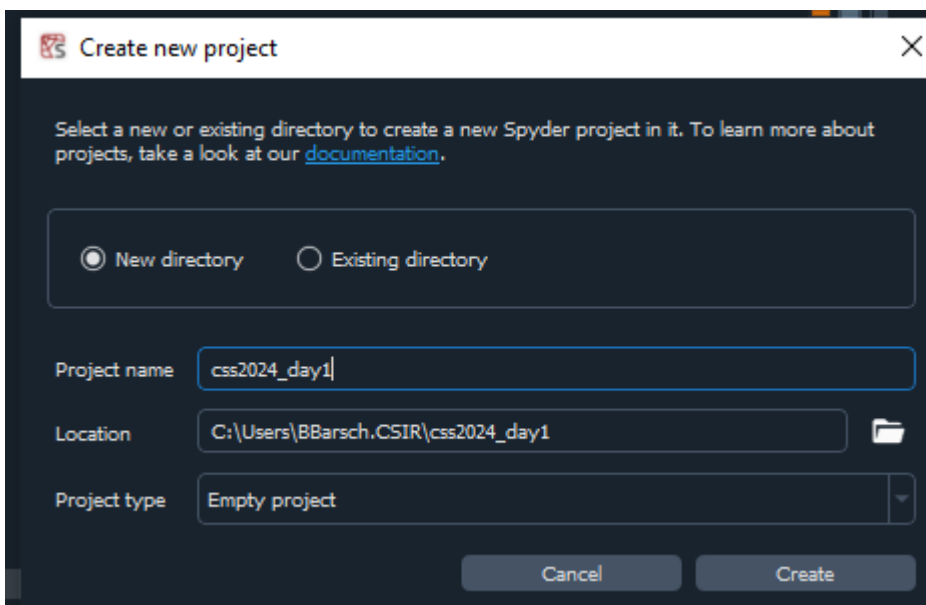
1. Python Scripts

We are going to write a recipe for instructions in Python, this type of file is known as a Python script.
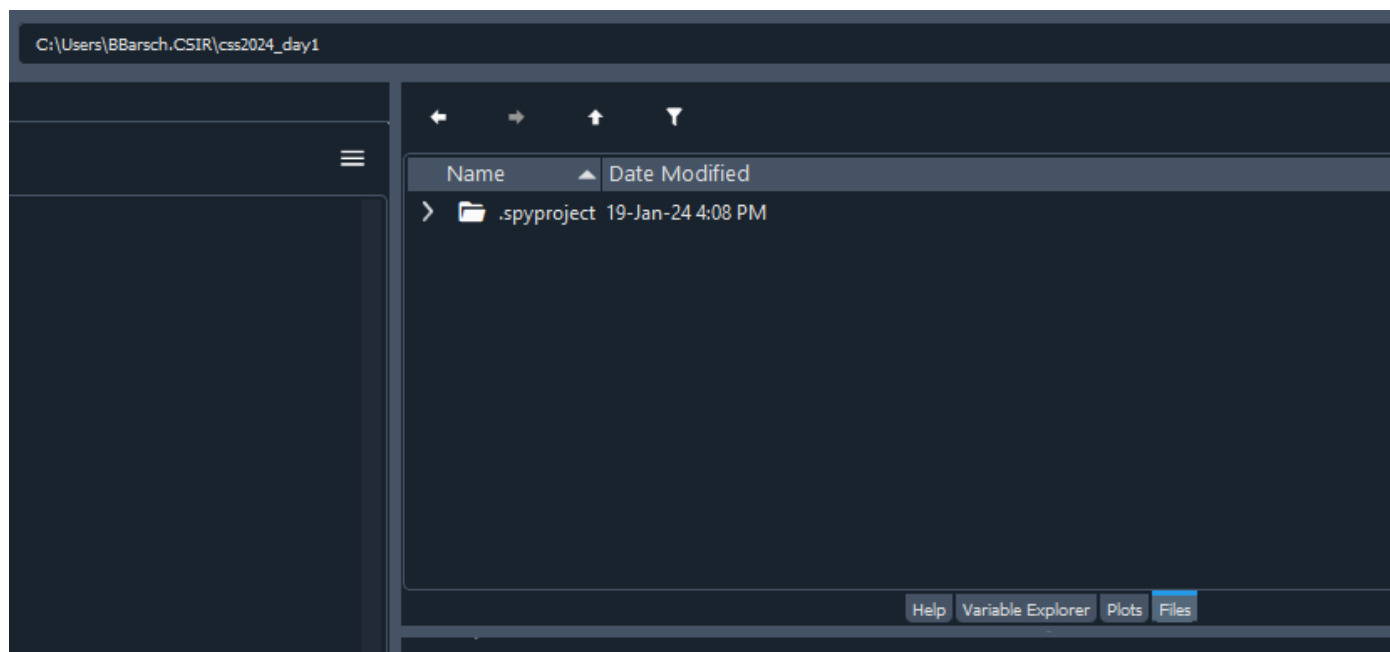
The first thing we will do is to create a Project directory to keep all our files:
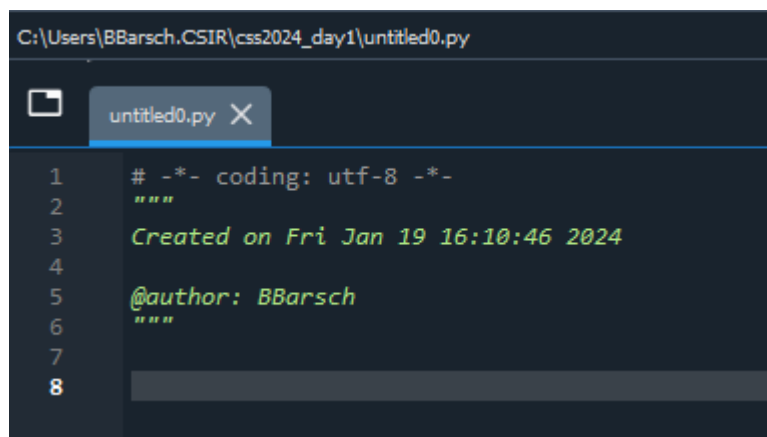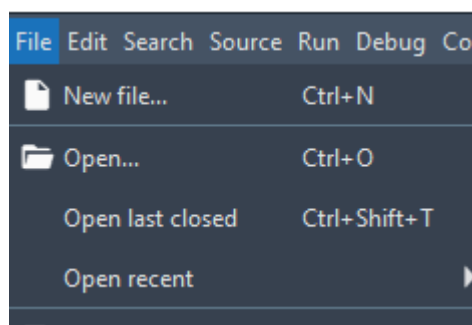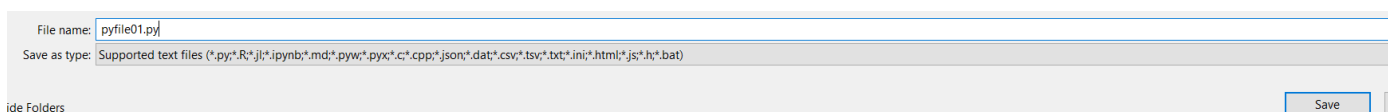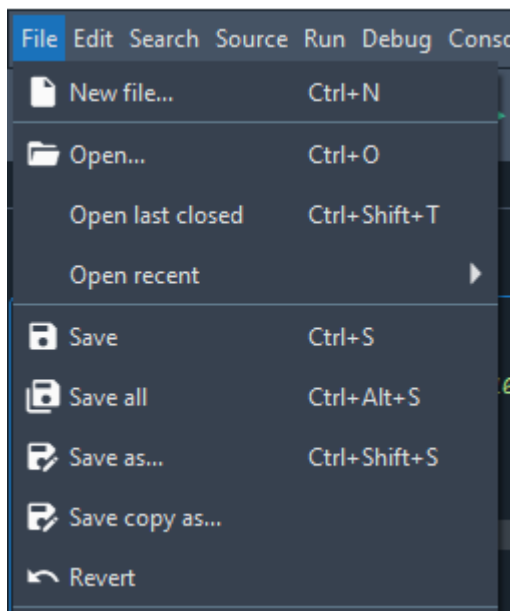

Choose a name and location:

You will see now the folder location and your files in the current folder, which only has the default project file `.spyproject...`.
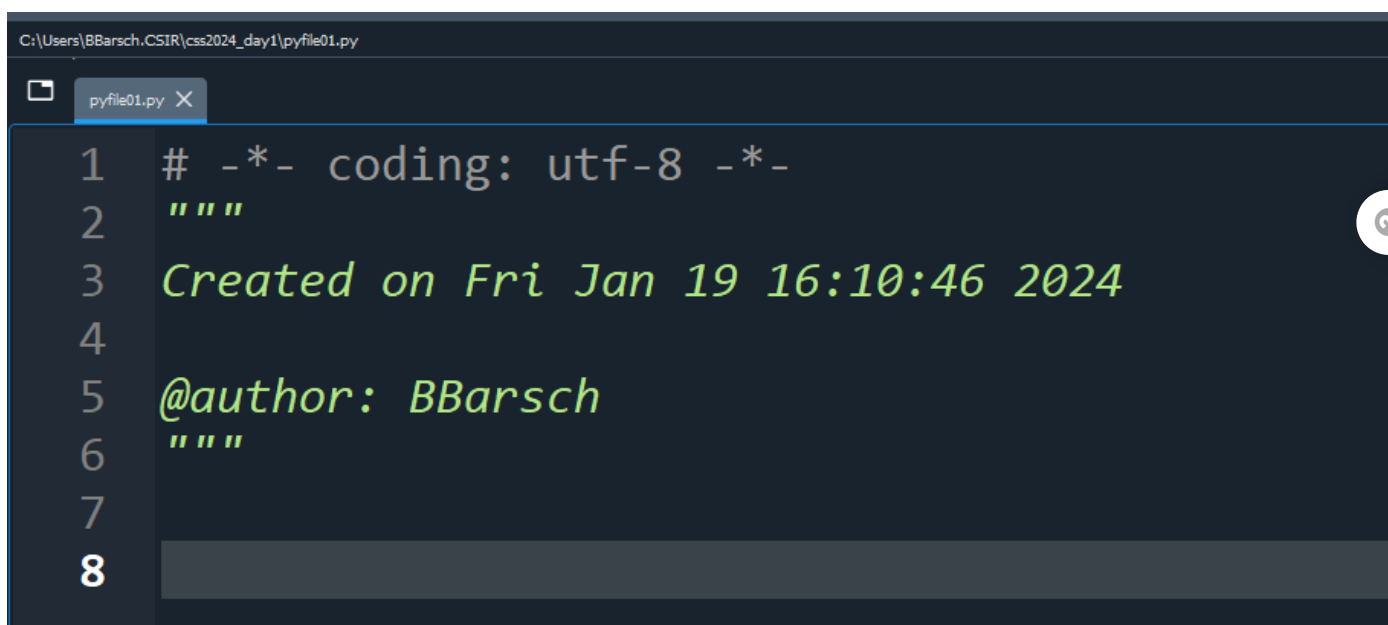
On the left-hand side of your screen you will see your text editor. Click on File -> New File:

It will now create a temporary file called "untitled0.py", if you save it will ask you for a name like pyfile01.py

File	Edit	Search	Source	Run	Debug	Cons

New file...	Ctrl+N	
Open...	Ctrl+O	
Open last closed	Ctrl+Shift+T	
Open recent	▶	
Save	Ctrl+S	
Save all	Ctrl+Alt+S	
Save as...	Ctrl+Shift+S	
Save copy as...		
Revert		

File name: pyfile01.py

Save as type: Supported text files (*.py;*.R;*.jl;*.ipynb;*.md;*.pyw;*.pyx;*.c;*.cpp;*.json;*.dat;*.csv;*.tsv;*.txt;*.ini;*.html;*.js;*.h;*.bat)

ide Folders   Save

It will be saved in your project:

C:\Users\BBarsch.CSIR\css2024_day1\pyfile01.py

pyfile01.py ✕

```python
# -*- coding: utf-8 -*-
"""

Created on Fri Jan 19 16:10:46 2024


@author: BBarsch
"""


```

And now listed as well here on the right-hand side:
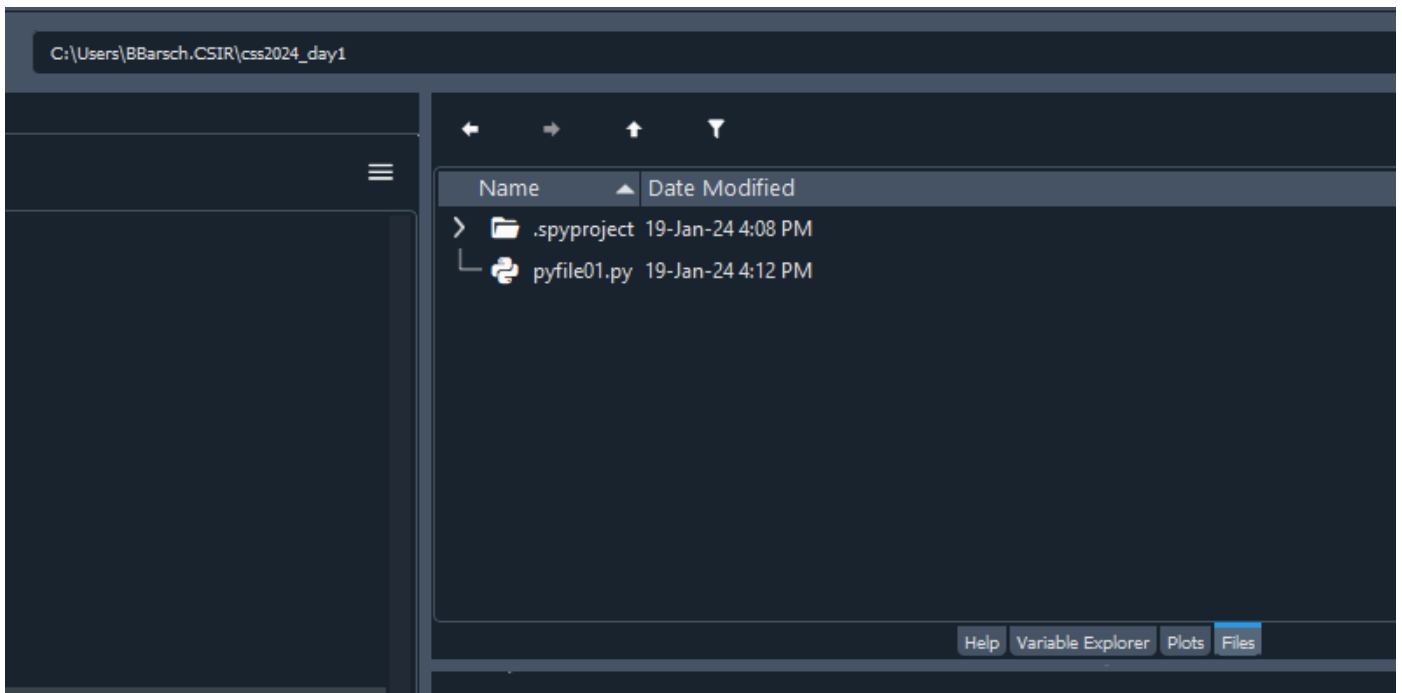
Now in the text editor input `print("My First Python File")`:

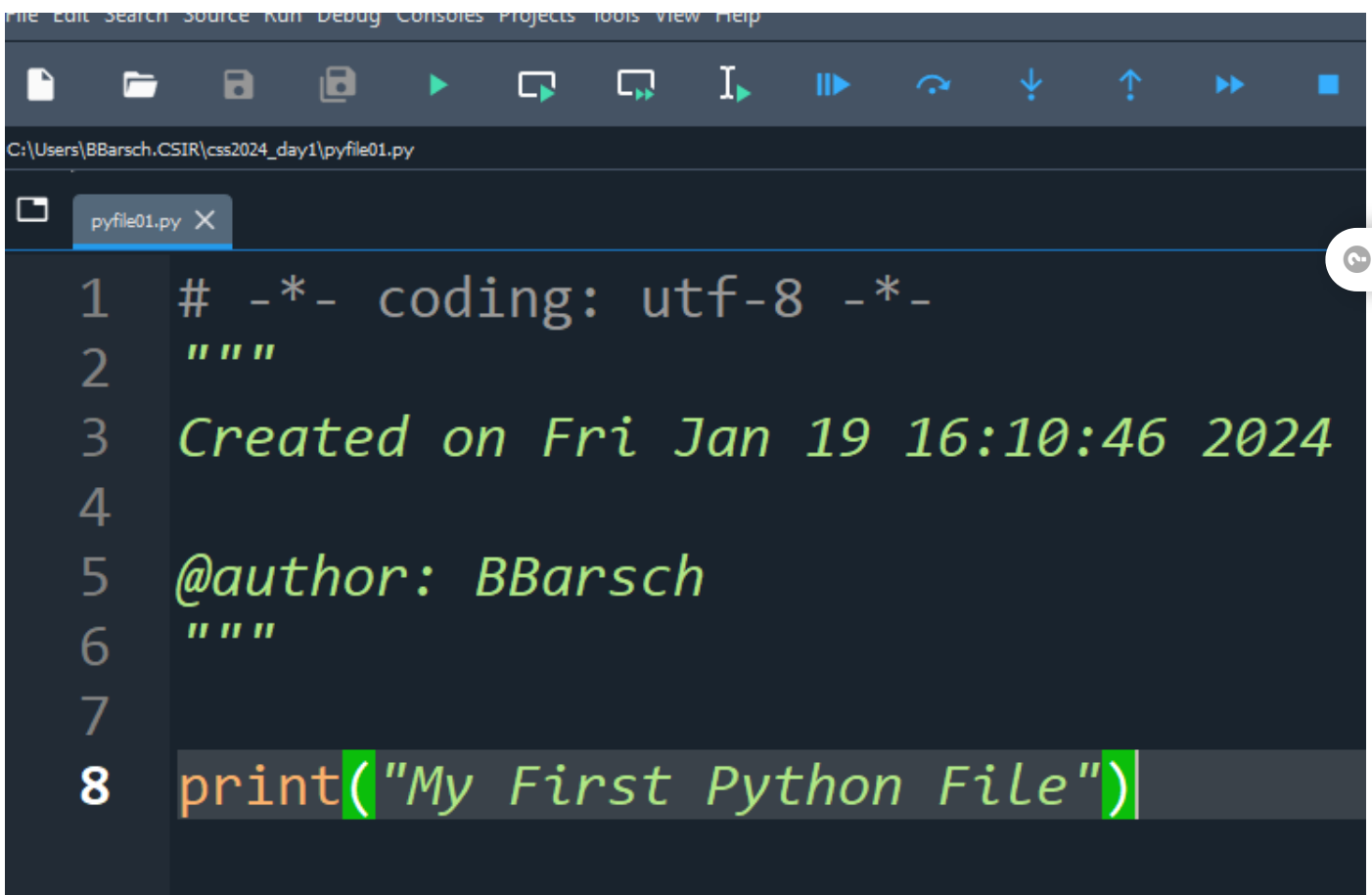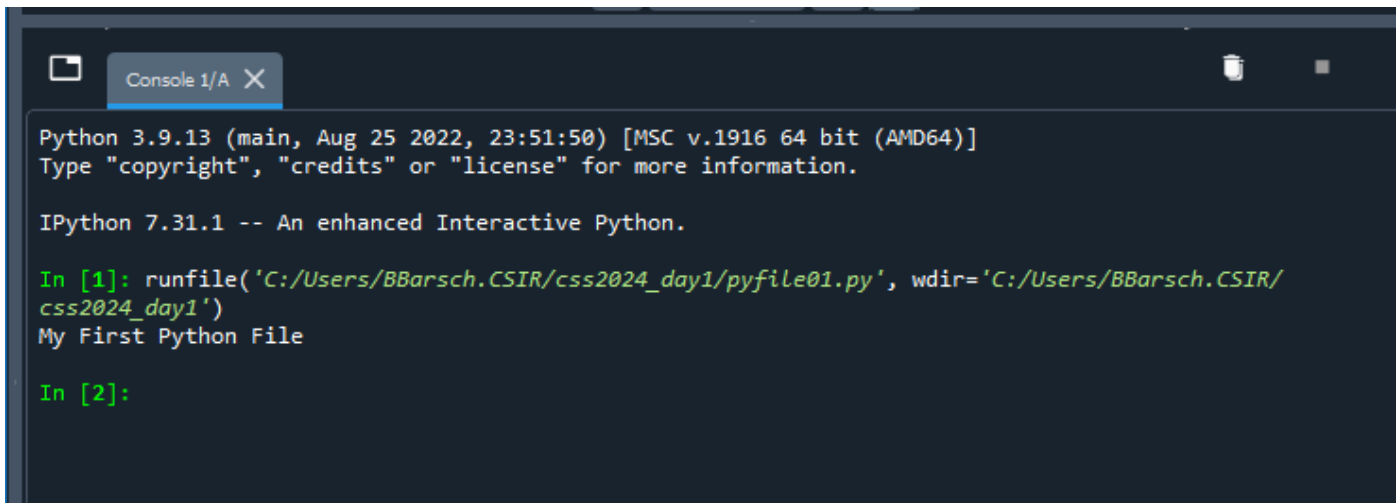Save it "Ctrl+S" and then press the green trianle Run button or press "F5" to run the script. A window may popup suggesting to "Execute in current console", just say Yes.
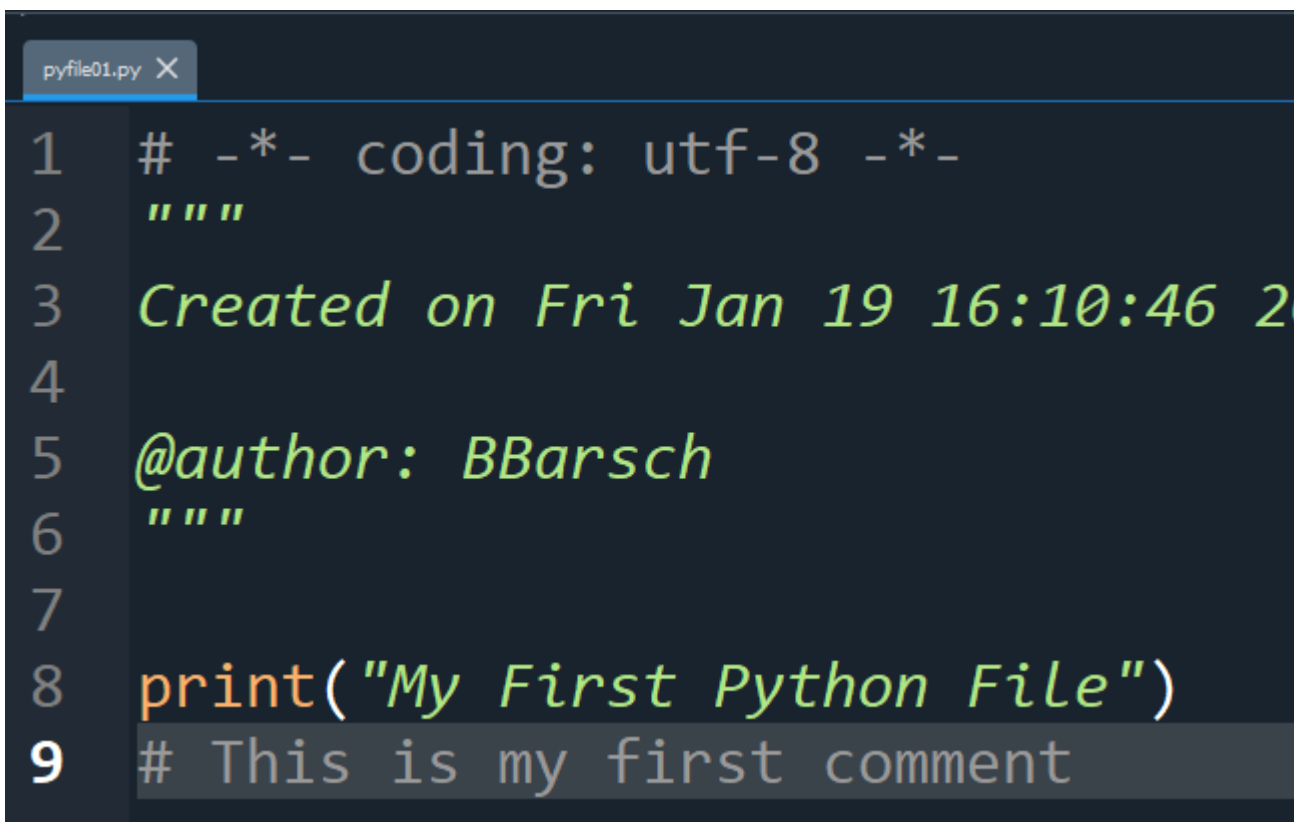
You will see it outputs the text to the console. This is the basic process to code and run the file. Write some text or code in the text editor section, run it, and see the output in the console...thats it.

A good practice with coding is to add comments to your file so you know what have done. To create a comment you type the # hash symbol before anything, then you can add any text after that. Python will interpret that as a comment and not execute it. Input the following under your print() function in the file:

```
# This is my first comment
```


Run the file, do you see any change?

For all the next few lessons, try out the examples by putting it into your python file and running it.

Python Browser Option

Link:

https://trinket.io/embed/python3

Console:
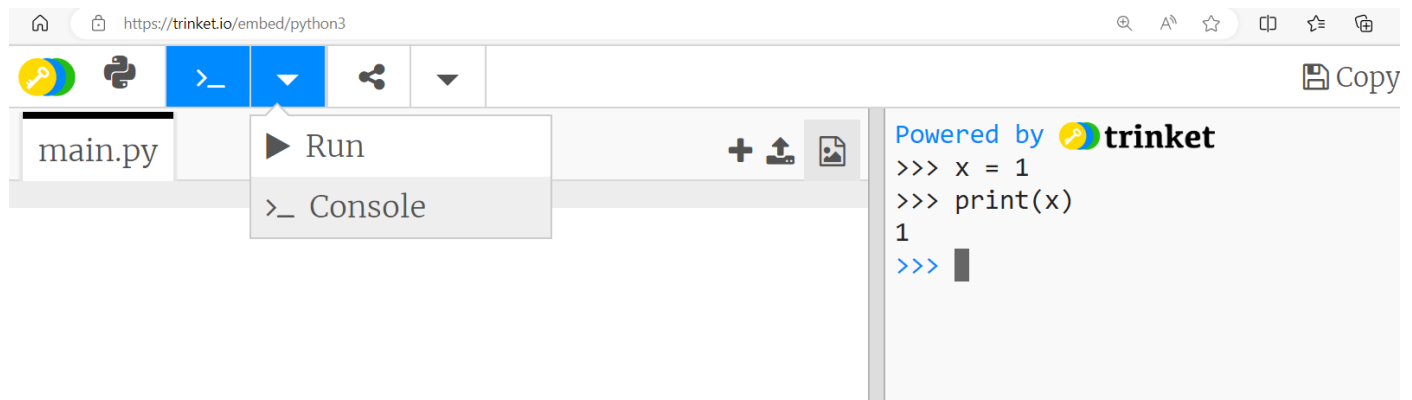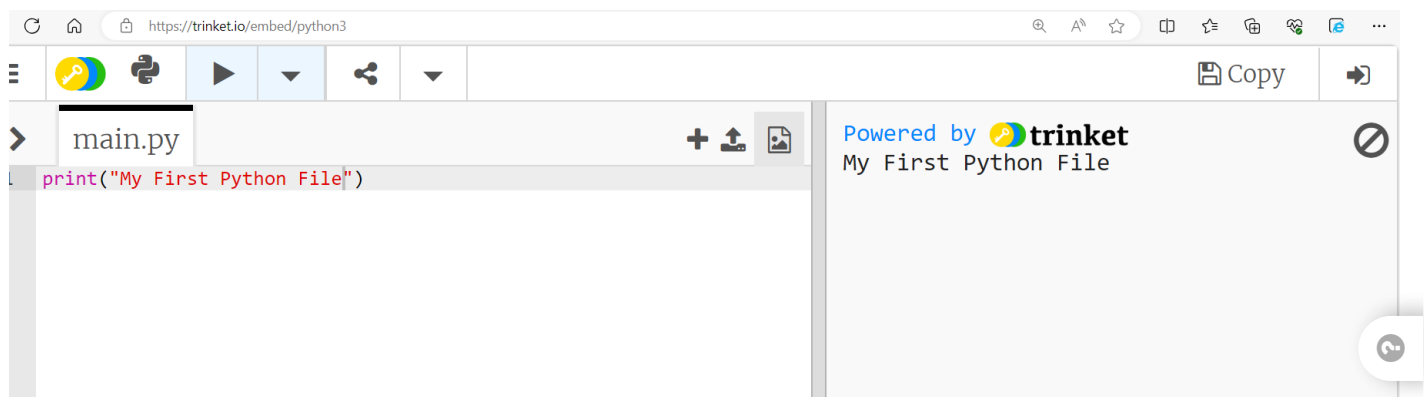
Script:

2. Variables

"Why was the data scientist sad?... Too many variables and not enough observations?"

Variable Naming

It is recommended that when naming variables you follow the following rules:

- A variable can have a short name like x or y or more descriptive ones like age, my*age, country*list.
- A variable must start with a letter or underscore
- A variable cannot start with a number
- A variable can only contain alpha-numeric characters and underscores
- A variable name is case-sensitive: age, Age, and AGE are all different variable names

Have a look at the examples below:

```
# Valid Names
x = 50
age = 50
_age = 50
```

```
my_age = 50
mYagE = 50
my_age_2 = 50
```

```
# Invalid names
2age = 50
my age = 50
my-age = 50
```

Try them out in the console and see what happens...

Variable Types

In the previous we lesson we did a few examples with variables. We learnt about two types of variables which is an integer and a string.

*And integer (or int) refers to whole numbers that are either positive or negative. For example 5, 2021, or -256. *

A string refers to any text. That can be a single character, word, or groups of words.

There is one more type that we will use often which is a float or a variable number that has decimals.

The float or floating point number is any decimal number like 0.5, -34.56, and 3.1451245.

3. Reading In Files With Pandas

Create a new file called `pandas01.py`.

Download "data_01.zip" folder found in "Week 1" section.

Unzip it. You will find the a file called "country_data.csv", put this file in your project folder in Spyder.

Now input the following code in the script and run it:

```
import pandas

file = pandas.read_csv("country_data.csv")

print(file)
```

You should see the following output in the console:

```
    Age Gender       Country
0    39      M  South Africa
```

```
1      25        M         Botswana
2      29        F    South Africa
3      46        M    South Africa
4      22        F            Kenya
5      35        F       Mozambique
6      22        F          Lesotho
7      49        M            Kenya
8      30        M            Kenya
9      40        F            Egypt
10     30        M            Sudan
```

And if you click on your "variable explorer" above the console you should see a variable called "file":
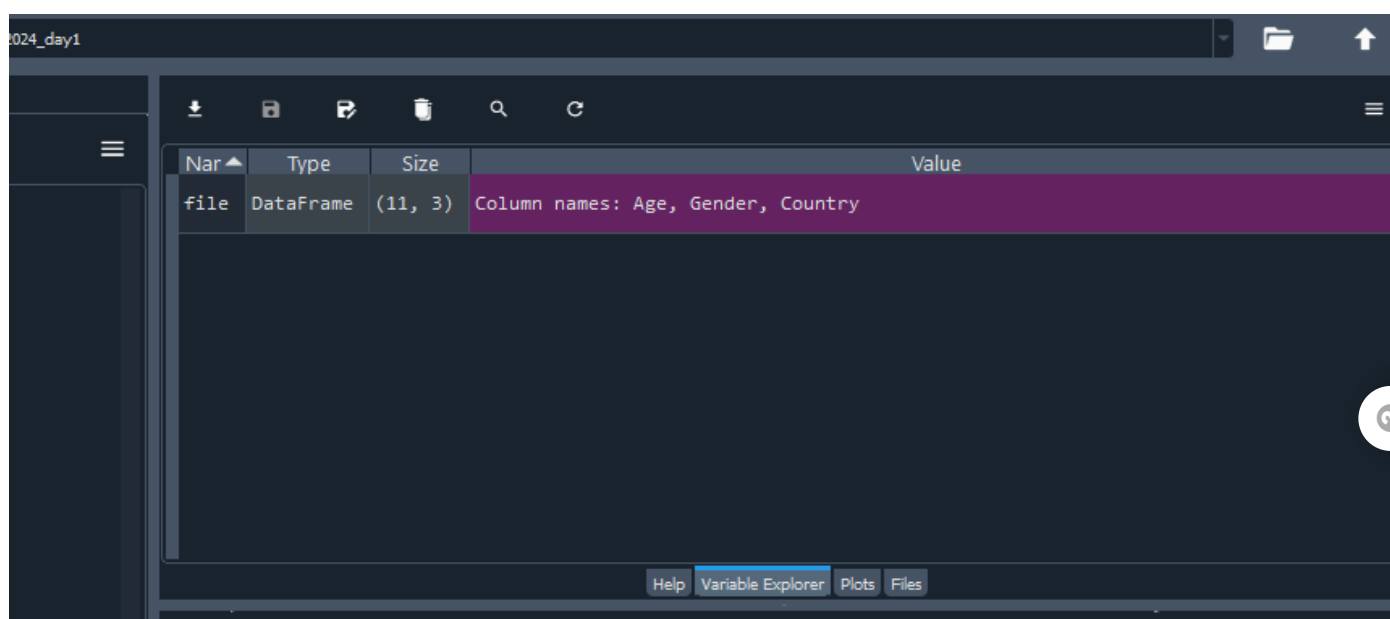
Double click on "file" in the "variable explorer" and you should see your data in a spreadsheet format:

And that is it, with 3 lines of code you read in a csv file in Python that you normally would open up in Excel.

Now let us explain the code:

`import pandas` : This line imports the pandas library.

* Pandas is a powerful library in Python used for data manipulation and analysis. *

`file = pandas.read_csv("country_data.csv")` :

This line reads a CSV file from a specified URL and assigns the resulting data to the variable named 'file'.

The CSV file contains country data on a website, you can also open files locally on your computer.

Also You can read in a variety of different types of files as shown on the Pandas official website: https://pandas.pydata.org/docs/reference/io.html

`print(file)` : This line prints the content of the 'file' variable to the console. It will display the data from the CSV file in a structured format, as pandas typically represents data in tabular form.

Now let us add two useful lines of code.

The first is:

```
print(file.info())
```

Console Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Age      11 non-null     int64
 1   Gender   11 non-null     object
 2   Country  11 non-null     object
dtypes: int64(1), object(2)
memory usage: 392.0+ bytes
```

This is a really useful feature of Pandas as it summarizes for you table. It tells you the amount of columns, their names, Non-Null Count, and the Dtype or data type. *The last one is an important one to consider as many times when you are reading in data, the numbers or dates can be read in as text or strings. Which prevents you from doing an approporiate filtering, sorting or numerical analysis*. In this case it was was read in correctly:

- int64: integer or whole numbers
- object: text or strings

The second useful featue of pandas is the following:

```
print(file.describe())
```

Console Output:

```
              Age
count   11.000000
mean    33.363636
std      9.233339
min     22.000000
25%     27.000000
50%     30.000000
75%     39.500000
max     49.000000
```

The .describe() method is used to generate descriptive statistics of a DataFrame or Series. It provides a summary of the central tendency, dispersion, and shape of the distribution of a dataset. It is primarily designed for summarizing numerical data, and it provides statistics that are meaningful for quantitative variables.

Practice!

Now use pandas and read in these files like we did above and run the .info() and .describe() methods as well.

You can find the files in "Week 1" section in the "data_01.zip" folder:

- iris.csv
- diab_data.csv
- housing_data.csv
- insurance_data.txt

Do you find anything odd or wrong with them?

Upload some of your own data files to the `#chat` *channel in slack, we can analyze them too!*

Python Browser Option
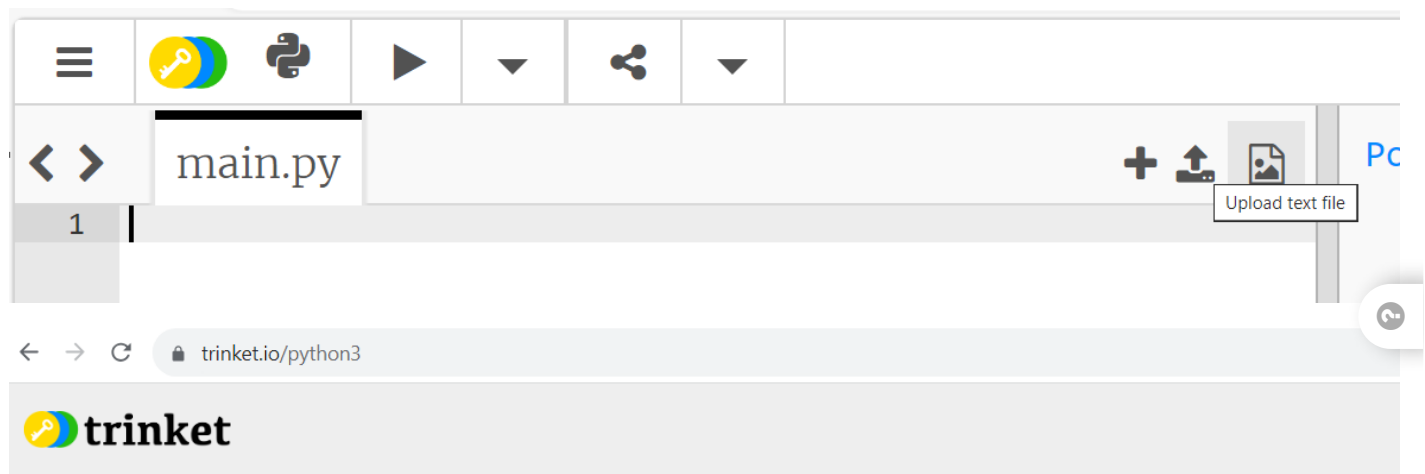
Upload a file:
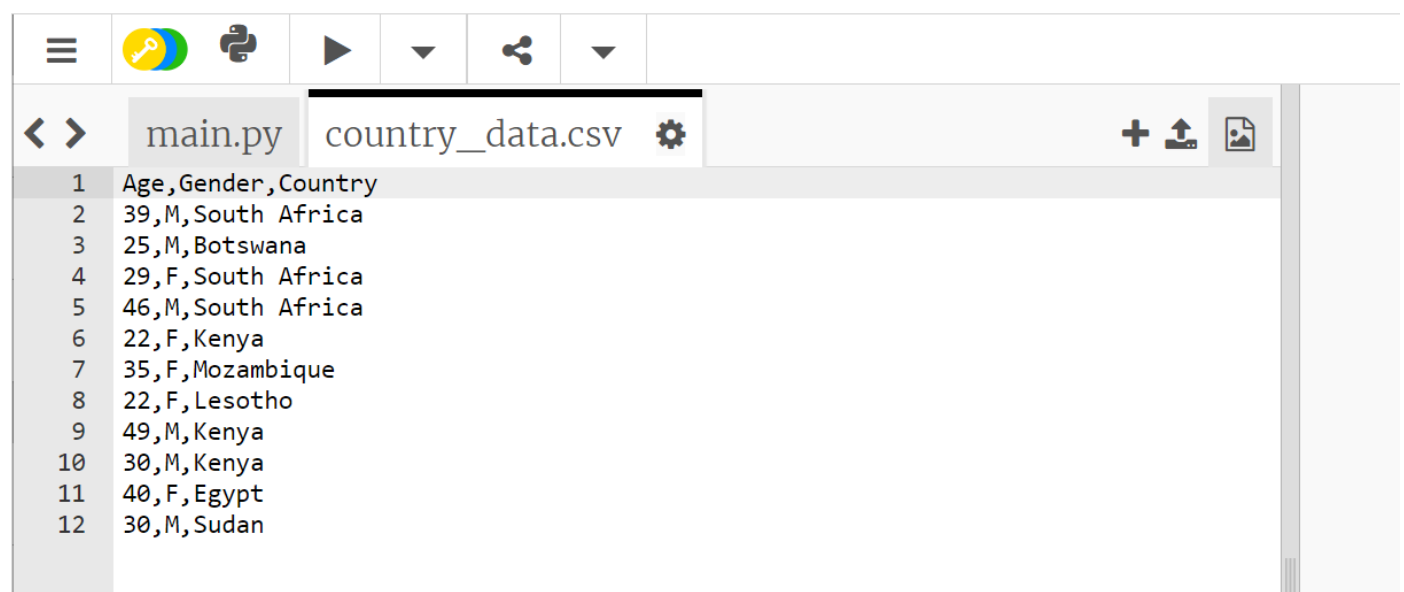
Put the code in "main.py" and run it:

```
import pandas

file = pandas.read_csv("country_data.csv")

print(file)
```

Console Output:

```
     Age Gender        Country
0    39      M   South Africa
1    25      M       Botswana
2    29      F   South Africa
3    46      M   South Africa
4    22      F          Kenya
5    35      F     Mozambique
6    22      F        Lesotho
7    49      M          Kenya
8    30      M          Kenya
9    40      F          Egypt
10   30      M          Sudan
```

4. Storing Data in Python

We will look at a few ways, namely lists, dictionaries and most importantly data frames which is associated with Pandas.

Now let us look at an excel dataset:

	A	B	C	D
1	Age	Gender	Country	
2	39	M	South Africa	
3	25	M	Botswana	
4	29	F	South Africa	
5	46	M	South Africa	
6	22	F	Kenya	
7	35	F	Mozambique	
8	22	F	Lesotho	
9	49	M	Kenya	
10	30	M	Kenya	
11	40	F	Egypt	
12	30	M	Sudan	
13				

The first thing we wanted to do was get some statistics on the age column, like the minimum, maximum, average, and standard deviation metrics.

Now there are two ways of doing this. The better approach is just to import this excel file into Python and then find the metrics. The second way and longer way is to input the data ourselves. We will be doing the latter in order to understand the Python principles of variables and data types.

Just like in excel where we are storing numbers in cells we can also store numbers in Python as variables. So for example we have at cell B2 is equal to 30, cell B3 is equal to 40, and so on. Run the code below:

```
# Storing Data
B1 = 30
B2 = 40
B3 = 30
B4 = 49

print(B1)
print(B2)
```

Try storing the rest of the variables that we got from the excel dataset in column B and print them out too.

Now to store each value in a separate variable is a tedious process especially if we want to get some statistics from it. So let us rather use a variable that can store multiple values. There are many of these in Python, for now we are going to use one called a list. And a list is defined by square [] brackets with the values inside the square brackets separated by commas. We can also give a list a unique name. Let us call it age to match the excel column B dataset and do it as follows:

```python
# Using Lists
age = [30,40,30,49,22,35,22,46,29,25,39]
print(age)

# Lists indexes start at 0 which has the value of 30
print(age[0])
print(age[1])
print(age[10])

# This will give an error as there is no value at index 11
print(age[11])
```

This looks a lot better. We can store all those values of age in a list data type called age:

```python
age = [30,40,30,49,22,35,22,46,29,25,39]
```

Take note that to access any of the values all we need to do is use age[index]. Where index refers to the location in the list datatype. Note, the first value is stored at index 0 not 1. The second value is stored at index 1 and so on. The last value is stored at index 10:

age =	30	40	30	49	22	35	...
index	0	1	2	3	4	5	...

If you try to access the index at 11 you will get the error list index out of range. See the excel data set again and look at column A called index to see how the values are stored.

Now we use some built in Python functions to get the same statistics we got in excel as shown below. Run the code below:

```python
# Basic Stats
age = [30,40,30,49,22,35,22,46,29,25,39]

print(min(age))
print(max(age))
print(len(age))
print(sum(age))
```

```
average = sum(age)/len(age)
print(average)
```

So the cool thing is we can use the built in functions min, max, len, and sum to get the minimum and maximum number, the length of the list, and the sum of all the numbers. Thereafter, we can easily get the average. We will look at more of a manual approach later instead of using these functions. However, you will need to learn a little more do that. We will look at getting standard deviation a little later as well.

Let us now record all the information in other columns in lists too. So in the excel data set we have column C that holds Gender values for M (male) and F (female). Like before we can store the variables individually based on the cells. Run the code below:

```
# Storing Text
C2 = M
C3 = M
C4 = F
```

But wait! Its giving us an error name 'M' is not defined Why is that? Any letters, words, group of words fall under the category of a string data type. Therefore we need to add " to the beginning and end. Run the code below:

```
C2 = "M"
C3 = "M"
C4 = "F"
print(C2)
print(C3)
print(C4)
```

Again this is a tedious task so let us create a list for gender:

```
# gender list
gender = ["M","M","F","M","F","F","F","M","M","F","M"]
```

Run the code below:

```
gender = ["M","M","F","M","F","F","F","M","M","F","M"]
print(gender[0])
print(gender[1])
print(gender[2])
print(gender[-1])
```

This is the same as before. We can access individual values starting with index = 0 for the first value. A nice feature of Python is that you can access the last value in the list by setting the index

= -1.

Lastly, we can do the same for the country column and just make a list. Note, country values are strings as they are words or groups of words:

```
# country list
country = ["South Africa","Botswana","South Africa","South Af
rica","Kenya","Mozambique","Lesotho","Kenya","Kenya","Egyp
t","Sudan"]
```

Run the code below:

```
country = ["South Africa","Botswana","South Africa","South Af
rica","Kenya","Mozambique","Lesotho","Kenya","Kenya","Egyp
t","Sudan"]

print(country)
print(country[0])
print(country[5])
```

"Coding like poetry should be short and concise." — Santosh Kalwar

Lists

Lists have many features. We will be covering some of them in this lesson.

A list can also contain all kinds of data types, run the following code:

```
# Data Storage With Lists
my_list = [42, -2021, 6.283,"tau", "node"]
print(my_list)
```

We can do many things with lists:

- print all the items in the list using [:]
- we can add items to the end of the list using append()
- add items at specific positions using insert()
- remove an item from the list using the remove() function
- check how many variables are in the list using the len()
- print a range of values using [start:end] – just like in excel!
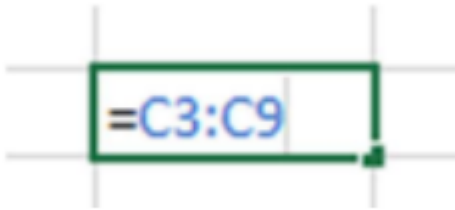
Let us see some examples of this. Run the code below:

```python
# Data Storage With Lists
my_list = [42, -2021, 6.283,"tau", "node"]
print(my_list)
print(my_list[:])

my_list.append("pi")
print(my_list)

my_list.insert(1,"pi2")
print(my_list)

my_list.remove("pi")
my_list.remove("pi2")
my_list.remove("tau")
print(my_list)
print(len(my_list))

# View a certain range of items:
print(my_list[0:3]
```

When we are getting only a certain range of values from a list we call this list slicing. We can do a lot of cool things with lists like sorting, and joining. Please note, like with any tool, this data type has its limitations. You will learn over time to use the best data type for the right task.

Dictionaries

A dictionary in Python is a collection of key-value pairs, where each key is unique. Dictionaries are also known as associative arrays or hash maps. They are similar to lists, but instead of using integer indices to access elements, you use keys. It is an unordered data type that is structured using keys and values and is defined with curly brackets {}. Where keys are like the column names and values are the lists of data.

The syntax for creating a dictionary is as follows:

```python
d = {'key1': 'value1', 'key2': 'value2'}
```

Here's an example of how you can use a dictionary to store information about a person:

```
person = {'name': 'John Doe', 'age': 30, 'address': '123 Main
St.'}
print(person['name']) # 'John Doe'
print(person.get('age')) # 30
person['phone'] = '555-555-5555'
```

In this example, the dictionary "person" contains three key-value pairs: name, age, address, and phone. You can access the values using the keys, add a new key-value pair and remove an existing key-value pair.

Dictionaries are commonly used in Python to store and organize data, and they can be very useful when working with large amounts of data, because you can quickly look up values by key without having to iterate through the entire data set.

Data Frame

When dealing with tabular data or datasets, the Pandas library provides a powerful and versatile data structure called a DataFrame. A DataFrame is a two-dimensional, labeled data structure with columns that can be of different types, similar to a spreadsheet or SQL table.

Let's convert the previous lists into a Pandas DataFrame to demonstrate its advantages over lists and dictionaries:

```
import pandas as pd

# Creating a DataFrame
data = {
    'age': [30, 40, 30, 49, 22, 35, 22, 46, 29, 25, 39],
    'gender': ["M", "M", "F", "M", "F", "F", "F", "M", "M",
"F", "M"],
    'country': ["South Africa", "Botswana", "South Africa",
"South Africa", "Kenya", "Mozambique", "Lesotho", "Kenya", "K
enya", "Egypt", "Sudan"]
}

#df = data frame
df = pd.DataFrame(data)

# Displaying the DataFrame
print(df)
```

Note: `import pandas as pd` and `import pandas` essentialy do the same thing. The `pd` is just a shorthand or shortcut way of using the `pandas`.

This single DataFrame now represents the entire dataset with columns for age, gender, and country. Let's explore some advantages of using Pandas DataFrame over lists and dictionaries:

1. Conciseness:

Storing and accessing data in a DataFrame is concise and easy to read.

2. Column-Based Access:

Accessing columns in a DataFrame is straightforward and intuitive.

```
# Accessing specific columns
print(df['age'])
print(df['gender'])
```

3. Descriptive Statistics:

Pandas provides built-in functions for basic statistics on columns.

```
# Basic statistics
print(df['age'].min())
print(df['age'].max())
print(df['age'].mean())
```

4. Filtering and Slicing:

Selecting and filtering data based on conditions is simplified.

```
# Filtering data
print(df[df['age'] > 30])

# Slicing rows and columns
print(df[1:4])  # Select rows 1 to 3 and all columns
```

5. Flexibility:

DataFrame allows easy addition or removal of columns.

```
# Adding a new column
df['new_column'] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
print(df)

# Removing a column
```

```
df.drop(columns=['new_column'], inplace=True)
print(df)
```

Well done if you have gotten this far. Remember to try all these examples yourself.

> *Start thinking of how you can use your own data in Python*