Python for Wet Etch Engineers

A blue and yellow snake logo

AI-generated content may be incorrect.

[Installing Python and Packages 4](#_Toc205205307)

[Installing Python 4](#_Toc205205308)

[Installing Visual Studio Code 4](#_Toc205205309)

[Installing Packages 9](#_Toc205205310)

[Common Packages 9](#_Toc205205311)

[datetime – smart handling for date and time 9](#_Toc205205312)

[NumPy – general purpose numerical handling 9](#_Toc205205313)

[Pandas – data analysis 10](#_Toc205205314)

[IPython – robust interactive shell 10](#_Toc205205315)

[email – Sending emails programmatically 10](#_Toc205205316)

[PyUber – SQL Queries 10](#_Toc205205317)

[Coding Basics 11](#_Toc205205318)

[Setting up a Python script 11](#_Toc205205319)

[Objects and Data Types 13](#_Toc205205320)

[Container 15](#_Toc205205321)

[Key Features 15](#_Toc205205322)

[Use For 15](#_Toc205205323)

[Example Code 15](#_Toc205205324)

[Control Flow 17](#_Toc205205325)

[Functions 20](#_Toc205205326)

[Packages and Modules 22](#_Toc205205327)

[Example Code and Summary 24](#_Toc205205328)

[Common Tasks 27](#_Toc205205329)

[Querying SQL 27](#_Toc205205330)

[Sending an Email 27](#_Toc205205331)

[Generating a plot 29](#_Toc205205332)

[Posting a Script to Scripthost 29](#_Toc205205333)

[Integration with JMP 30](#_Toc205205334)

[WEPy Documentation 31](#_Toc205205335)

# Installing Python and Packages

## Installing Python

The bare minimum you need to get started with Python is the programming language itself. To install this, go to <https://www.python.org/downloads/> and select the most recent stable version. This will download an installer. Find it in your downloads folder and run it, following the instructions.

A screenshot of a web page

AI-generated content may be incorrect.

If prompted, make sure that you add Python to your $PATH variable. This variable holds the list of places that Windows will automatically check while executing commands where the file path might be ambiguous. In other words, this will let you access Python anywhere on your computer.

## Installing Visual Studio Code

In principle, you can now write python scripts and run them as you please. In practice, though, there are a number of common tools and packages that make certain tasks and writing code in general much easier. The first one that I recommend is Microsoft’s VS Code. VS Code is a piece of software designed specifically to edit code, which includes syntax highlighting and other convenience features. Most programmers can’t live without it. To install VS Code, go to <https://code.visualstudio.com/download> and install the appropriate version.

A screenshot of a computer

AI-generated content may be incorrect.

A blue logo with a black background

AI-generated content may be incorrect.As before, find the installer in your Downloads folder and run it, following the instructions. Great! To run it, open the start menu and start typing “Visual Studio Code” until it pops up. The logo is pictured on the side. If you plan on coding a lot, it’s probably a good idea to pin VS Code to your taskbar. When you open it, here’s roughly what your screen should look like:

A screenshot of a computer

AI-generated content may be incorrect.

Outlined in green is where the file explorer is. When you start writing Python scripts, this is where your file tree will show up. This works in exactly the same ways as the Windows File Explorer. To open a file or folder to this pane, go to the top left of the window (above the green box in the figure) and select File 🡪 Open File or File🡪 Open Folder. If you want to create a new file, just double click in an empty part of the green pane, and put in the filename, including the extension. In this picture, I have a folder open called Python for Wet Etch. When you select a file, its contents will pop up where the red box is. If you click and drag up from the bottom of the screen, a terminal will show up (outlined in yellow). This is usually called the “command prompt.” Unlike Window’s File Explorer, the command prompt works specifically in one file location. If you look at the text, it will show you what file location you are in. For example, in the image my file path is C:\Users\msuskin\OneDrive - Intel Corporation\Desktop\Python for Wet Etch – that means that I am in the C: drive of my computer, in the Users folder, etc. To change directory, type cd [the directory you want to go to]. This can be done in a few ways:

1. If you want to go to a subfolder of your current folder, you don’t need to type the entire path of that subfolder. You can just type cd [folder name]. If the folder name has spaces, you must put it in quotes. (Like cd “Python for Wet Etch”)
2. Using a single period (.) is a shorthand for your current directory. (For example, the command cd . will not do anything.) That means you can travel to folders relative to yours without specifying the entire file hierarchy.
3. If you want to go up one level, you can just type cd ..

A screen shot of a computer

AI-generated content may be incorrect.The reason that you are obligated to do any of this, is that to run a Python script you must type python3 [the name of your script.py] into the command prompt. This will only work if your command prompt is set to the right folder. If this proves to be too difficult, VS Code also has a feature that can run code for you. When a Python script is open, you can click on the run icon in the top right of the window. It looks like a play button. You will see that this button essentially just opens up your console and does what is described above.

Python also has what is called a “shell,” which just means a type of program designed to run other programs. To activate the shell, type python into any command prompt. This will cause your prompt to change to >>>. The shell contains many nice features for advanced users, but if you feel lost you can just type help into the Python shell. To quit the Python shell just type quit(). Once you gain some skill with scripting in Python, it would be best to revisit the documentation for the shell so that you can run scripts in more sophisticated ways.

## Installing Packages

A Python package is just a script that somebody else (or many people) have already written to handle a common task or set of tasks. Several packages are already included with the Python installation from python.org (detailed in an earlier section). So many packages exist on the internet that there are actually packages whose sole purpose is installing other packages. One such package is called “PIP,” and it comes pre-installed with Python. To use PIP, go to any command prompt and type python -m pip install [the name of the package you want] --proxy=http://proxy-us.intel.com:912. If you were doing this on your home computer, you would not need the proxy argument.

## Common Packages

Some of the most common Python packages are behemoths in their own right. Depending on your job, you may choose to install some, all, or none of these packages, which provide various functionalities. Some may also already be covered by the WEPy package I’ve written to handle some common wet etch automation tasks. A word of warning: all of the packages listed here are widely used in industry and academia and are generally robust and error-free. They also all have good documentation available online (put “package name docs” into your favorite search engine). As you get into more niche tasks, the packages that you encounter will be more prone to issues and less documented. Remember, all packages are just other user’s code, so be careful with installing them. To see which packages you have installed, type help(‘modules’) into your Python shell. Also, you might want to skip this section and come back once you’ve gotten started programming.

### datetime – smart handling for date and time

<https://docs.python.org/3/library/datetime.html>

The Python datetime library is useful for handling dates and times that may have an inconsistent format. It provides objects and methods for handling dates homogenously. This can be helpful when handling SQLPF output, for example.

### NumPy – general purpose numerical handling

<https://numpy.org/>

NumPy is the gold standard Python package for handling numerical issues and objects. It includes a huge number of sophisticated algorithms ranging from efficient Fourier Transforms to linear algebra routines.

### Pandas – data analysis

<https://pandas.pydata.org/>

The core utility of pandas is the DataFrame object. A DataFrame represents a table of values along with row and column indexes. Surrounding this is an enormous pastiche of different functions for analyzing and manipulating the data.

### IPython – robust interactive shell

<https://ipython.org/>

IPython is different from the other packages in this list in that it doesn’t provide much functionality directly to scripts. Instead, IPython is run from a command prompt in the same way scripts are. IPython provides an extremely well-featured shell that allows you to manipulate Python functions and objects outside of the context of a script.

### email – Sending emails programmatically

<https://docs.python.org/3/library/email.examples.html>

The e-mail package allows you to send e-mails from Python scripts. This functionality is also implemented in the WEPy package.

### PyUber – SQL Queries

The PyUber package allows you to connect to the XEUS database (and others) through Python. To query XEUS, create a connections object using the PyUber.connect function. PyUber.connect takes a string that represents the data source. To call it as a wet etch engineer, you should call it like conn = PyUber.connect(datasource="D1D\_PROD\_XEUS", TimeOutInSeconds = 600). This will store a PyUber.connect object in the variable conn. To access the data as a pandas data frame, use pandas.read\_sql\_query() with a string representing the query and conn as the arguments. I usually generate the query string using SQLPF. See sample code below.

A screen shot of a computer

AI-generated content may be incorrect.

# Coding Basics

Python is what’s called an interpreted programming language. What this means is that every Python script is a set of instructions for a pre-existing program (Python.exe in this case). The instructions are read from top to bottom and executed in that order. The Python.exe program also processes the script you write in a way that makes it more readable for a computer. For example, empty lines will be ignored. You can feel free to add whitespace as you see fit to make code more readable.

A screen shot of a computer

AI-generated content may be incorrect.Another thing ignored by Python.exe is a comment. Adding a hashtag (#) symbol before a line will make the interpreter ignore it. This is true even if there is other code on the line before it. You can also add triple quotes around text to make it ignored for multiple lines.

As you can see in the image, if you are using VS Code comments will automatically be colored (green for single line, brown for multiline). Why would you want to do this? Even if you know at the moment what your code is supposed to do, you might forget by the time you get back from lunch, let alone after not looking at it for a month. This is especially true if your code is particularly long or sophisticated. Adding comments always pays dividends when you’re trying to understand your own code, to say nothing of other people’s code.

You can also “comment out” code while testing or debugging a script to examine how the behavior changes without that segment of code. This is fairly common practice, and if you look at scripts written by other people you can sometimes see commented out code that didn’t make the final cut.

In fact, a novice programmer will find that many things in Python are “unnecessary.” There is nothing stopping you from skirting all of these extra features, but in time you will come to think of them as luxuries, not hindrances. The experienced coder knows that an error you understand is always better than an output that confuses you.

## Setting up a Python script

There are only two types of file on your computer: text and binary. The file extensions you encounter daily (.txt, .py, .xlsx, etc.) are just decoration that suggest to the computer and the users how the file should be handled. As a result, any text file can be fed to the Python interpreter and VS Code is mostly just a fancy text editor. To write your first Python script, just open VS Code and double click on the file pane. Type in a name that ends in A screenshot of a computer

AI-generated content may be incorrect..py (in the picture I’ve done test.py).

A typical thing for a programmer to do when they begin programming in a new language is to output “Hello world” to the command prompt. This is important because the command prompt (also known as the console or terminal) is where you’ll want to put all of the intermediate output. By intermediate output, I mean any output related to debugging, making sure that the program is proceeding as intended, and maybe even the overall results of your calculations. For example, if you write code to send the results of an SQL query in an email, you may want to write out to the console after the SQL query to confirm that you successfully pulled the data. While testing the program, you may choose to also print out the query output to confirm that it looks how it should.

Enough stalling: to write out anything to the console, use the print()function. The print function takes whatever you pass to it, tries to find a text-based representation, and writes it to the terminal. So, to output “Hello world,” write print(“Hello world”), save your script (Ctrl + S on windows), and then run it, either by typing python3 your\_file.py or by clicking the play button in the top right of the VS Code window. Congratulations on your first Python script! Your screen should like something like the picture below.

A screenshot of a computer

AI-generated content may be incorrect.

On Windows, you can type clear or cls to clear the console.

## Objects and Data Types

A screen shot of a computer

AI-generated content may be incorrect.You are probably aware that computers store data as binary. Another way to say this is that all data looks the same, and the thing it represents is only decided by interpretation. For this reason, data is split up into objects, which have types. The intuition behind this is obvious: we should have a way to distinguish between text and integers, for example. Also, the type of an object specifies how it should behave. For example, would you expect 5 + “apple” to be a meaningful piece of code? Even though every object has a type, the Python interpreter does not force you to specify in advance what the type is and tries to infer it from context. But the type is always there. Here is some Python shell output that shows what the type of “Hello world” is from the program above. Text is represented by objects of class str, which is short for string. If you are not certain of what the type of an object is, you can find out by calling the type() function on it, as in the picture. Other types include int (integer), float (floating point number, which represents a decimal number), char (character), list, tuple, dict (dictionary), set, bool (Boolean, either True or False), and NoneType. This list is not exhaustive of the default Python types, but you can also define your own types, and packages frequently will do so.

Out of all these types, int, float, char, and bool have something in common: they are not container types. A container type means a type that holds references to other objects. For example, a list, as the name suggests, is an ordered sequence of other objects. In fact, under the hood the str type is just a list of char objects!

A screenshot of a computer program

AI-generated content may be incorrect.The way that you assign objects in Python is simple. Just write object = what you want to assign. The interpreter will automatically guess (“interpret”) what you want the object to be typed as. For example, the code x = 5 will create an integer variable named “x” that has value 5. This works in all the ways you would expect with respect to usual functions. However, a subtle point here is that the interpreter is guessing based on what you assign. Case in point: the assignment x = 5.0 is different than the code above, since 5.0 will be represented as a float by the interpreter. This is also true for x = “5”.

A screen shot of a computer

AI-generated content may be incorrect.

A black background with white and pink text

AI-generated content may be incorrect.A black screen with white text

AI-generated content may be incorrect.Unfortunately, there is no way to specify types ahead of time (that is native to Python). The only way to guard against type errors like this is vigilance and care with what you feed your variables. You can also convert variables of a given type to another type, if this behavior is defined. It might not be in certain cases, though, or it might behave unexpectedly even if it is. For example, converting a float to an int will automatically round it to the nearest integer. Even though this is a reasonable thing to do, it might mess up calculations if you aren’t expecting it. (In this example 5.5 is converted to just 5) This process of converting one type to another is called “casting.”

The container objects from the above list are list, tuple, dict, and set. They each have different properties that make them suited for different purposes. In each case, the way that you access the object held by a container set is using an index. What the index is depends on the container object. No matter what, the syntax looks like container[index]. Here is a summary of these four container objects and how they work:

|  |  |  |  |
| --- | --- | --- | --- |
| Container | Key Features | Use For | Example Code |
| List | Has a specific order.  Can be changed after assignment (mutable). Indexed by numbers. | Fairly general purpose. Use any time order is important. |  |
| Tuple | Cannot be changed after assignment (immutable). More memory efficient. Indexed by numbers. | Use when you need to reference static data frequently. |  |
| dict | Can be changed after assignment. Indexed by objects. The first object in a pair is called a key and the second is a value. The value can be any object, but this is not true for the key. | Use for when you need to set up a correspondence between pairs of objects. |  |
| Set | Cannot contain duplicate items. Does not have an order. | Sets have efficient operations. Use when order does not matter. You can use this object to get unique values from a list. |  |

If you aren’t too sure about sets and tuples, just stick to lists and dicts. These are the two most useful types. Something very important to consider is that Python objects with a numerical index have that index start at 0, not 1. As you can see from the example code, the container[index] could mean multiple different things based on the context. In short, calling it as container[index]will return the object contained at that index. This can be used either to get the object itself, or as an assignment. If an object already exists at that index, it will be overwritten or modified, depending on the object and the function called. Otherwise, the object and index will be created in the container object (assuming the container is mutable).

Lastly, unlike non-container objects, since container objects can be empty but still exist, they can be assigned ahead of time. For example, to create a list called x with no elements, just write x = list(). Then x will behave as lists usually do. Like non container objects, you can cast containers. For example, one method to get the unique elements of a list is to cast it as a set:

A black screen with white text

AI-generated content may be incorrect.

Even though this has obvious utility, it still bears mentioning that casting when you don’t mean to can mess up your script in unpredictable and sometimes hard-to-spot ways.

A black screen with white text

AI-generated content may be incorrect. The other way to use this index notation is by doing something called “slicing.” Slicing a number-indexed container object will return a subset of the same container object using the conditions you provided. For example, you can get the second and third element of a list called x with the expression x[1:3]. This is because the second index is not included in the slice, and the first index is 0. Another subtle point is that even if your index returns only a single element, it will still be a list, just one containing only one element. Negative numbers count backwards from the end. (x[-1] will give you the last element of x)

Types also can have functions that are specific to them. They are called type functions or class methods and are called by the special syntax object.function(). For example, calling x.keys() on a dict x will return a list of the first values for each pair in x. More on functions in subsequent sections.

To help digest the material so far, I suggest the following exercises:

1. Open up a Python shell and find out what 5 times 10 is.
2. Make a list that is the numbers 1 through 10 in order. (Like x = [1, 2, … 10]). Then see what indexes correspond to what numbers.
3. Do this one in a script. If you have pets or kids, make a dictionary where their names are matched up with their ages. (Like x = {‘Spot’ : 5, ‘Timmy’ : 25}) Make the script print Timmy’s age to the terminal. Then change it to output Spot’s age.

## Control Flow

The main things that computers do better than people are that they are very fast with arithmetic, and they don’t mind doing boring things a truly large number of times. In that spirit, there are some functions that fall under the category of control flow that specify how the Python script should do a particular action or set of actions multiple times. This is usually referred to as a loop.

There are two types of loops, and they are fundamental to programming everywhere. The first is a for loop, and it specifies an action to be performed for each element of an iterable object. An iterable object is essentially just a container object where the notion of a “next” element is defined. The most common example is a list or tuple. The for loop is best used when the number of iterations is fixed, but the action changes with each one. The syntax goes like this:



Here is an example of a for loop:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

In this example, at each step in the loop, the next element in apples is assigned to the variable apple. Then, the print() function is called on the variable apple. The power of the for loop is that the same code can be reused many times to different effects each time, depending on the iterable.

A screen shot of a computer

AI-generated content may be incorrect. The other kind of loop is the while loop. This loop runs while a bool variable evaluates as True. If you need a loop running while a condition is False, then just add not in front of the expression. The condition is checked before starting each loop. This loop is used when the loop should be run an indefinite number of times. Be careful with for loops, since if the expression doesn’t ever evaluate to false, then the loop will run forever. A common pattern with while loops is to have a counter, and to increment it when a specific thing happens, and to set the break condition of the loop as a certain counter value. An example of this:

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.A screen shot of a computer code

AI-generated content may be incorrect.Also under this category are functions that cause an action to be performed only if a condition is met. Specifically, the if function executes a block of code only if the bool condition it has as an argument is True. This can also be paired with an else statement, which will execute its code only if the if expression doesn’t happen. Therefore, the two code snippets on the right are equivalent.

A computer screen with white and blue text

AI-generated content may be incorrect.You can also chain multiple statements together using the if/elif/else construct. “elif” is short for “else if.”

Believe it or not, these three things are the building blocks that essentially make up all of software. Every program that exists can be rephrased in terms of just these constructs.

Here are some exercises to make sure you have the basics down:

1. Write a script that searches a string for a certain character and then prints its position in the string. For example, when you run your script with “ventricle” and “n” it should print out 3.
2. If you used a for loop for the previous exercise, use a while loop instead. If you used a while loop, use a for loop. These two constructs are usually interchangeable.

## Functions

A black background with white text

AI-generated content may be incorrect. Functions are the programmer’s lifeblood. Broadly speaking, a function is a block of code that takes zero or more arguments and then does something. It can also optionally return an object at the end. Functions in python are defined in the following way:

Every single function is defined like this; even functions you might not expect. For example, here is part of the definition for the add function.

A screen shot of a computer program

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.This is the function that is called when you write an expression like 1 + 2 or ‘a’ + ‘b’. Here is another example:

A screen shot of a computer

AI-generated content may be incorrect.This example is another common pattern; defining a function that returns a bool value, which is then used as the condition for an if/else block. Notice that there is no need to store the output in a variable. The python interpreter first stores the function isAnimal(). Then, in line 11 the if statement invokes isAnimal. isAnimal returns a bool value (i.e., True since my\_pet == “caterpillar” evaluates as True). Within the function, the return statement immediately ends function execution.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A function need not have any return value. This is where the special NoneType comes in. Have a look at the code on the left; the first statement stores the return value of the print function called with the argument ‘Hello!’ in the variable x. The print function, however, does not have any return value. Its only purpose is writing to the console. (something that a function does outside of its own scope in addition to what it returns is sometimes referred to as a “side effect.” There are entire languages designed with the express purpose of having no side effects. Thankfully, Python isn’t one of them.) Therefore, x simply holds a NoneType object. The whole point of this object is that it is not compatible with any other objects under any operations. Imagine, for a minute, if print returned 0 on a successful print to console. If you made a mistake and wrote x = print(‘Hello!’) when you meant to write x = 5, then you wouldn’t get an error. You would just be stuck trying to figure out why your answer was 5 off. This is the utility of the NoneType. You can also explicitly specify an object of NoneType like x = None.

Here are some exercises that reinforce the ideas of this section:

1. Functions can take any type of object as an argument. Write a function that applies a different function to every element of a list. When you’re done, you can make sure yours works right by comparing to the map() function.
2. It may surprise you to learn that functions can include calls to themselves in their definitions. This is called “recursion.” Using this fact, write a function called Fibonacci(n) that returns the nth Fibonacci number. (Fibonacci numbers are defined recursively. The first two are both 1. Then, every Fibonacci number is the sum of the previous two. So, the third number is 1 + 1 = 2, and the fourth is 1 + 2 = 3, and the fifth is 3 + 2 = 5, and so on.)
3. Write a function that calculates the area of a square or a rectangle.

## Packages and Modules

Packages provide functionality outside the basics described previously in this section. To include the functions and objects from a package in your script, you must use the import statement. It goes like this: import package. Just include this text at the top of your script, and you can then invoke the functions and objects like package.function() or package.object(). If you only need one thing from a package then this syntax can get quite annoying. Therefore, you can also write this: from package import something. Then you can just call something() as though you had defined it yourself. If you want multiple things, write from package import thing1, thing2. If you want all the things from a package, write from package import \*. If the package is large, then this is not recommended, since importing a function with the same name as one you’ve defined or one that is provided by Python can cause issues and unexpected behavior.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

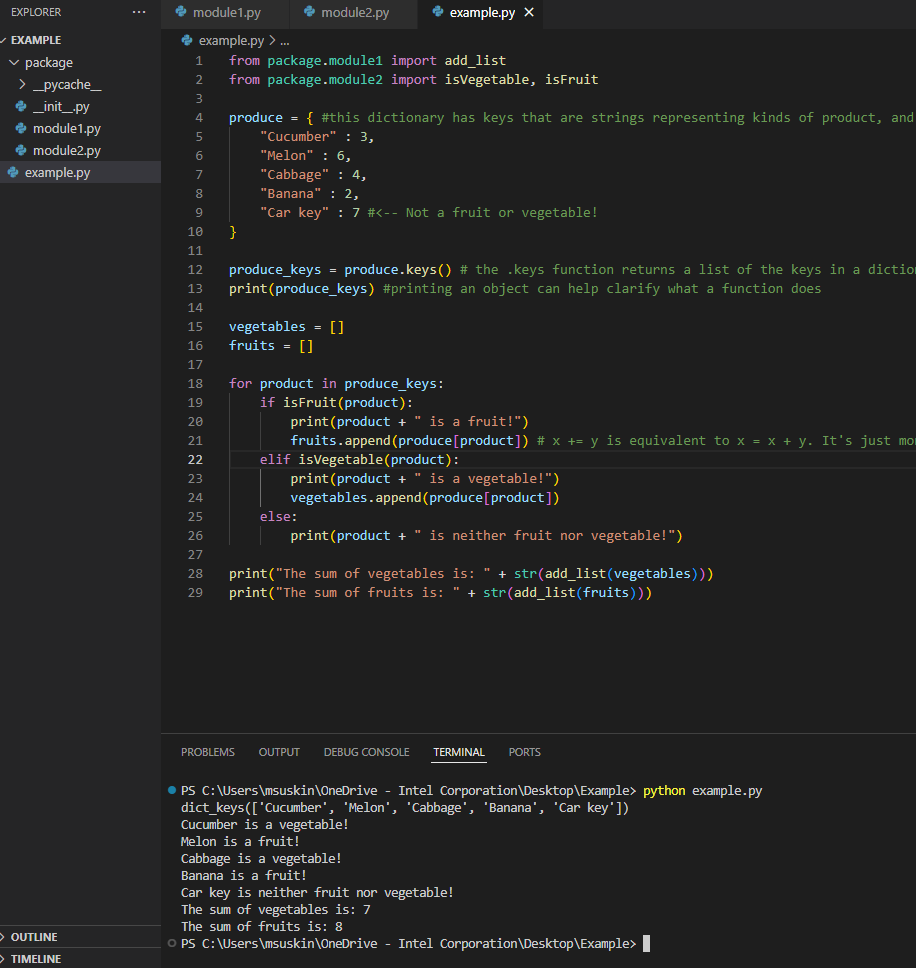
AI-generated content may be incorrect.You can also import functions from your own scripts. If you’ve defined f(x) in function.py, you can include it in script in the same directory with the statement from function import f. function.py would be called a module. A package is simply a collection of modules in the same folder. To make a folder into a package, make a script and called \_\_init\_\_.py, and put it in the folder. This script doesn’t need to have anything in it; it just needs to exist.

Now your folder is a package. Call on functions and objects from your package using:

* import package
* import package.module
* from package import module
* from package.module import object.

## Example Code and Summary

Here is a recap of the Python basics. A Python script is a list of instructions for the Python compiler. A Python script works by storing a variety of different objects in memory and taking actions on them. In practice, a script usually has some kind of input, and almost always has some kind of output. Instead of specifying long or repetitive instructions by hand, loops are used to specify what to do at each step using logic.

Here is an example of a script that uses all the concepts that have been outlined, with clarifying comments.

A screenshot of a computer program

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.

Explaining, this program is split over three scripts: module1.py, module2.py, and example.py. module1 and module2 are both part of the package package. (See file structure in the example.py image.) module1 defines the add\_list() function, which returns the sum of a list of numbers. module2 defines the isFruit and isVegetable, which return bool indicating whether a string is a fruit or vegetable.

The main script, example.py, has a dictionary which gives a number for several different items which are each fruit, vegetable, or neither. You might expect this sort of setup at a grocery store, for example. In a real script, this data would probably be sourced from some other program or database.

* The keys class method is used to retrieve the keys of the dict produce, so that the for loop can iterate over the list that produce.keys() returns.
* Two lists are declared, fruits and vegetables.
* For each element in the list produce\_keys, the imported functions are used to decide whether each element is a fruit, vegetable or neither.
* After the loop is complete there are two print statements. For strings, the + function concatenates. So ‘hello’ + ‘guys’ 🡪 ‘helloguys’. The add list function is called to sum up each of the lists of fruit and vegetable numbers. Then the output is printed. Since + is not defined between str and int, the output of add\_list must be cast to a str first.

Naturally, this is not the most efficient way to accomplish this task, but it serves as an example of many concepts discussed.

Here are some exercises that reinforce the material discussed so far:

1. Container objects can contain other container objects. The syntax for indexing them is as you would expect: container[index1][index2]. Suppose you are given a container object that is nested in this way. Write a function that can detect the total number of objects contained by the highest container in the hierarchy. For example, your function should return 3 when called on the list l = [[], [[]]] because the list contains one list that is empty and one list that contains another list.
2. Can you write a function that takes a function as its input and returns a dict that says what the return type is for a given input type for that function? For example, your function should return {int : int, float : float, etc. } when it gets the exponent function as input.

# Common Tasks

## Querying SQL

Querying the XEUS database and others can be done with the PyUber package. The way to do this directly through the PyUber package is described in the common packages section. However, you can also use the wrapper function provided by the WEPy (Wet Etch Python) package. First, set up your query in SQLPF. Then, when you are ready, press the button that looks like a hand writing on a notepad.

A screenshot of a computer

AI-generated content may be incorrect.

Copy the text between where it says “Begin SQL” and “End SQL.” Your copied text should have the format “SELECT… FROM… WHERE… ORDER BY…”. Simply paste this into your Python script and store it in a variable for ease of use. Here’s what your script might look like:

A screenshot of a computer

AI-generated content may be incorrect.

## Sending an Email

Sending an email can be done in one of two ways: directly using the Python email library, or by using the WEPy package. The WEPy package should be sufficient for most job functions, but power users may feel the need to deal with the email package directly for niche cases. Anyone interested in using the email package directly is referred to the documentation: <https://docs.python.org/3/library/email.html#module-email>.

To use the WEPy email functionality, you must instantiate a WEPy report object. The report object is instantiated with a dict, where the keys (the first object in a pair) are names used to reference data, and the values (the second object in a pair) is the data itself. The WEPy package is set up so that all of the functions that deal with retrieving data can be fed directly to the report object without modification. Without going into excessive detail, there are five core functionalities to the report object: add\_table, add\_plot, add\_text, add\_image, and email\_report. The first four append the described object to the email, based on the data provided in the dict during instantiation. email\_report sends the email via the Intel email server. These methods are especially helpful when paired with an SQL query. See example code below.

A screenshot of a computer program

AI-generated content may be incorrect.

And the email output that it generates:

A white background with black text

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Not the prettiest, but it sure is easy to code up quickly. If you do prefer to stylize your HTML elements a bit more substantially, the add\_table() class method accepts an argument called styler\_func, which is a function that acts on a pandas styler object. See <https://pandas.pydata.org/docs/reference/api/pandas.io.formats.style.Styler.html>.

## Generating a plot

Generating a plot from a pandas dataframe can be done using the plot() method from pandas. Be aware that this method depends on having the matplotlib package installed. See <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html>.

## Posting a Script to Scripthost

Unfortunately, the only way that I know of to post a Python script to Scripthost is kind of convoluted. First, make sure that your script is in a folder that is accessible to Scripthost, like MAO data or similar. If it has any dependencies other than standard packages (or common packages such as pandas, Numpy, etc.) they should also be in the same folder. Then, open a new SQLPF query. Go to Add View or Utility 🡪 Utilities 🡪 Python. Double-click the Python utility and specify the file path to your script. Now you can post to Scripthost as you would for any other query.

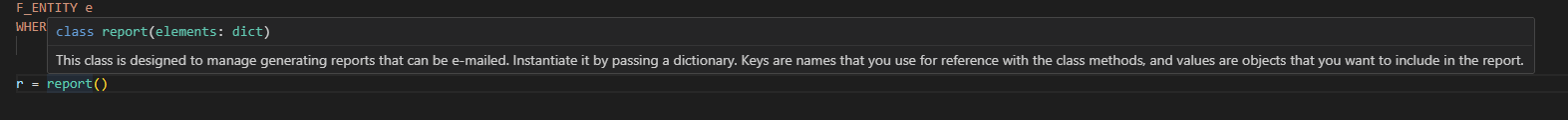
## Integration with JMP

Since JMP comes with a set of Python integration features, no attempt has been made to add any to WEPy. See <https://www.jmp.com/support/help/en/18.1/index.shtml#page/jmp/python.shtml>.

# WEPy Documentation

1. File\_methods.py
   1. get\_csv(args)
   2. get\_excel(args)
2. SQL\_methods.py
   1. get\_SQL(args)
3. report.py
   1. report(self, elements)
      1. elements: dict where keys are nicknames (strings) used to reference the values. Values can be any object.
   2. report.add\_table(self, tables, cols, join\_type, join\_on, styler\_func)
   3. report.add\_plot(self, tables, x\_axis, y\_axis, cols, join\_type, join\_on, \*\*kwargs)
   4. report.add\_text(self, header, body, \*\*kwargs)
   5. report.add\_image(self, image)
   6. report.email\_report(self, subject, sender, recipients)
4. tooling.py
   1. Attributes:
      1. tool.name: a string representing the name of the tool, such as ‘DRT452’ or ‘AUR408\_POLYMIX’
      2. tool.isTool: a Boolean representing whether or not the tool object has a physical counterpart. This is a safety feature to prevent trying to retrieve information from XEUS about tools that don’t exist.
      3. tool.children: a list of tool objects that are subtools of the tool.
   2. tool(self, name, isTool)
      1. Instantiate a tool object, which will recursively detect all child entities from XEUS. Therefore, you can do either an actual tool name or just a pattern (‘AUR’ will detect all AUR tools)
   3. str(tool)
      1. casting a tool object to a string (or doing it implicitly by calling print(tool object) will recursively walk over the tool hierarchy and return a human-readable string.
   4. get\_state(self)
      1. return the UTP state of a tool object. This involves a call to XEUS.
   5. isCu(self), isPb(self), isAu(self)
      1. returns True if the tool is a copper tool (or lead or gold).

Note: having self as an argument means it is a class method, and you call the function like object.class\_method(arguments).

When working in VS Code, hover over a function to see an explanation for it.

All the get\_\* methods work the same. args can be a string or some combination of container types holding strings. In the case of get\_csv and get\_excel, they should be the file path indicating where the file is held. For get\_SQL, they should be the query strings. For these functions, you can pass a string, multiple strings, a list of strings, or a dict. The functions roughly preserve the structure you give them, except with the strings replaced with dataframe objects. For example, if you pass a list of strings to these functions, they will return a list of dataframes.

The report module provides a class and class methods that allow you to easily generate and send email reports. To do this, first instantiate a report object using the report(dict) method. Then use various functions to add whichever parts of the report you would like (images, text, plots, etc.) Last, call the email\_report() class method to send the report as an HTML email.

The tooling module provides a way to programmatically represent a real tool or toolset without instantiating the model manually yourself. Although it is convenient, please be aware that this module is quite expensive since it makes recursive calls to XEUS. The overhead is least when you reference the same data many times. This is also not much of any issue if you plan on posting your script to scripthost.