Comments

Python is a text based programming language and machines run machine language, therefore before running any of our code it runs through a piece of software that ignores comments and looks at only the python code for the first step in translating to machine language.

There is nothing stopping us from running that same code through a different piece of software that ignores the Python code and looks only at the comments, which opens the door for us to put two entirely different sets of instructions into the same text file, this is the idea behind documentation generators. We put comments in our python code in a certain fashion so that an entire website can be generated documenting the project.

My personal favorite to use is Doxygen --originally for C++ but many more languages including python have been added to its capabilities, however, for this course we will be using a python speffic one: pdoc3

It is high on the recommended list from PyPI so to install pdoc3 use pip in a system terminal (cmd, powershell, bash shell, ...), make sure you install with pdoc3, as the original pdoc behaves differently and is far less powerful, yet both add the instruction "pdoc" to the same system terminal we call pip from. (if you're on a Linux or a Mac make sure you use pip3 here instead --***see bottom of this page for Google Colab or Jupyter Notebook instructions***).

```
pip install pdoc3
```

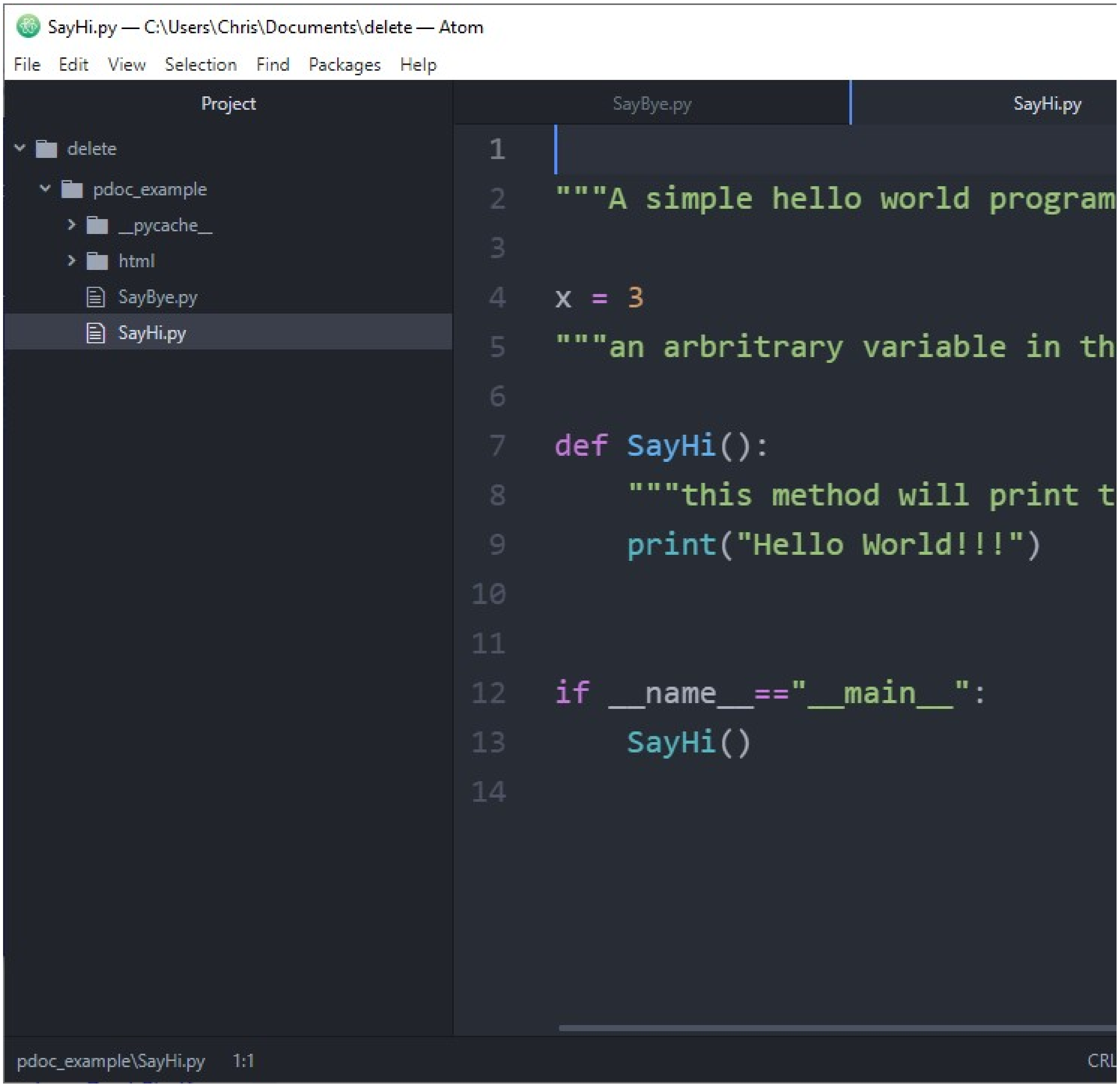
Then use docstrings (the triple quote marks) as comments to document each project, class, or module, then in a system terminal run the instruction...

for a Single script to create a Single web page:

```
pdoc --html -o filepath/output-folder-name filepath/scriptname.py
```

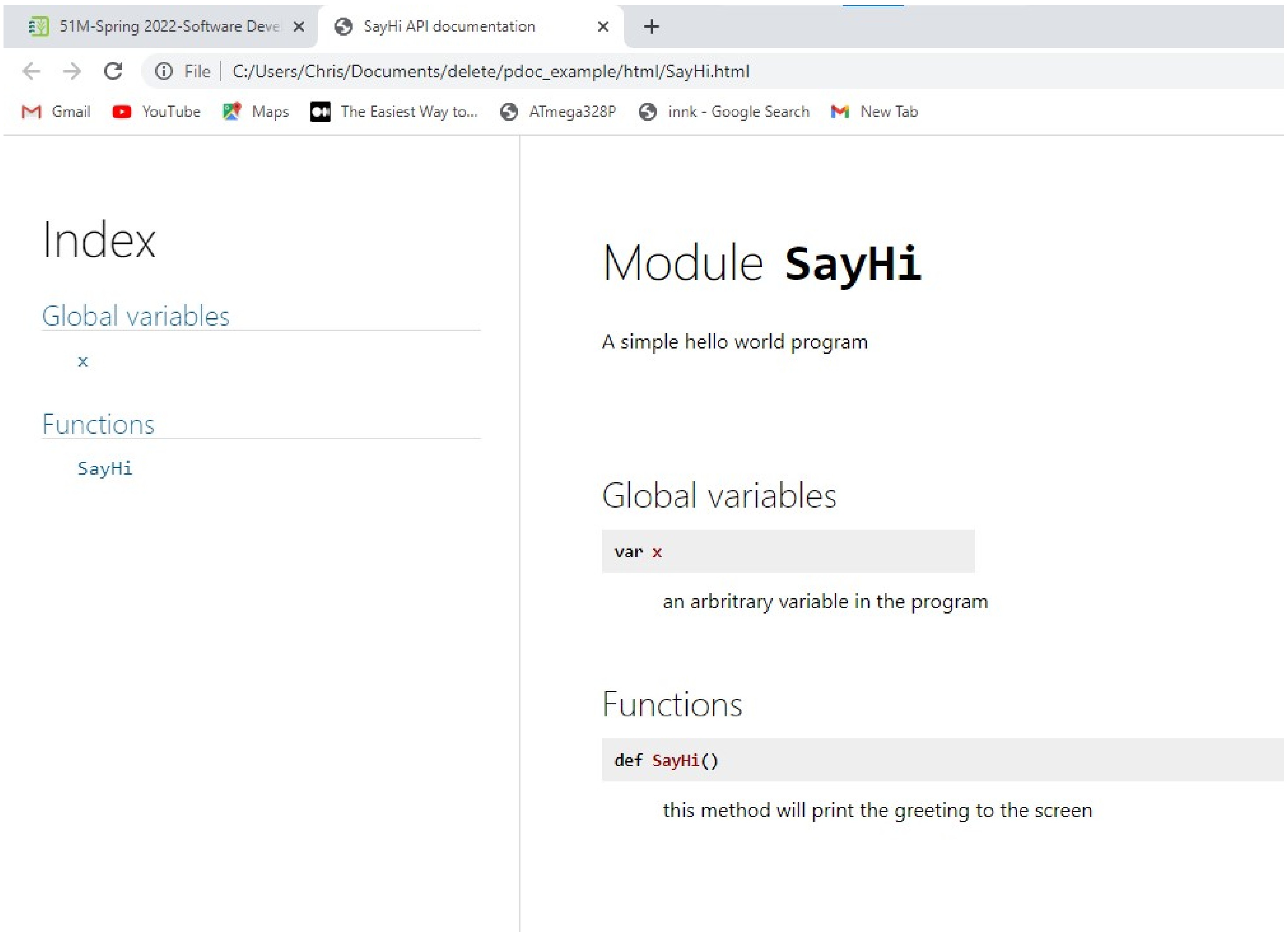
** the -o parameter is where to put the output, f you navigate to the directory you want to put it in (such as in the project folder itself) that can be omitted.

Here's an example, this script...


SayHi.py — C:\Users\Chris\Documents\delete — Atom
File Edit View Selection Find Packages Help
Project
delete
pdoc_example
__pycache__
html
SayBye.py
SayHi.py
SayBye.py
SayHi.py
pdoc_example\SayHi.py 1:1
CRL

```

"""A simple hello world program

x = 3
"""an arbitrary variable in th

def SayHi():
    """this method will print t
    print("Hello World!!!")


if __name__=="__main__":
    SayHi()

```

generated this webpage...



Index

Global variables

x

Functions

SayHi

Module **SayHi**

A simple hello world program

Global variables

```
var x
```

an arbitrary variable in the program

Functions

```
def SayHi()
```

this method will print the greeting to the screen

Or if you organize your scripts into a single project directory and want a full website:

```
pdoc --html -o filepath/output-folde-name filepath/project-folder-name
```

** ***It is easiest if you navigate into the directory with the project, then just use the dot operator to say "document stuff in this directory" with the command:***

```
pdoc --html .
```

Here's a zip file example (https://ivylearn.ivytech.edu/courses/1120010/files/96722114?wrap=1) (https://ivylearn.ivytech.edu/courses/1120010/files/96722114/download?download_frd=1) using all the files in the project shown in the atom text editor screenshot above to generate an entire website with an index page.

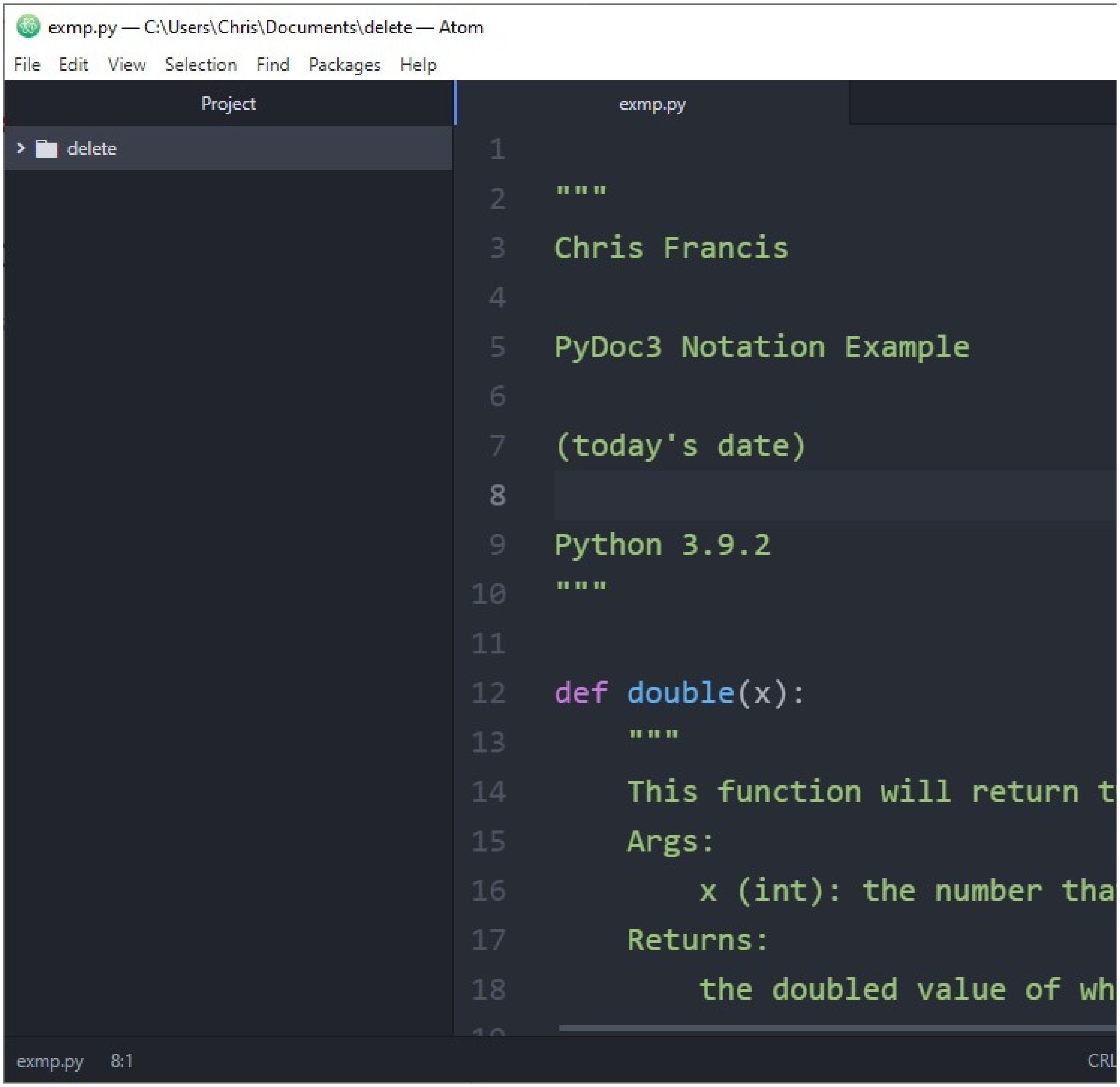
Please do this to document your code this semester, **you do not have to submit the website but feel free to** (if you're coding in notebooks special instructions are at the bottom of this page), **but you do have to include these sort of comments** include a docstring header with your name, the assignment, the date, and which version of the Python Language you used to test/debug the program on so that I can grade it on the correct version of Python. Here's an example, note the extra space is needed for linebreaks in summary section:

```
"""
Chris Francis

PyDoc3 Notation Example

(today's date)

Python 3.9.2
"""
```

Also document any functions/methods and classes in the following format (this is the industry standard known as "Google Notation" you'll be able to use in another document generator later in your career, I will also accept "Numpy Notation" in this class, see this page for examples look for methods named "google" and "numpy" here and expand the source code at the bottom of that function: **https://pdoc3.github.io/pdoc/doc/pdoc/test/example_pkg/#gsc.tab=0 (https://pdoc3.github.io/pdoc/doc/pdoc/test/example_pkg/#gsc.tab=0)**).

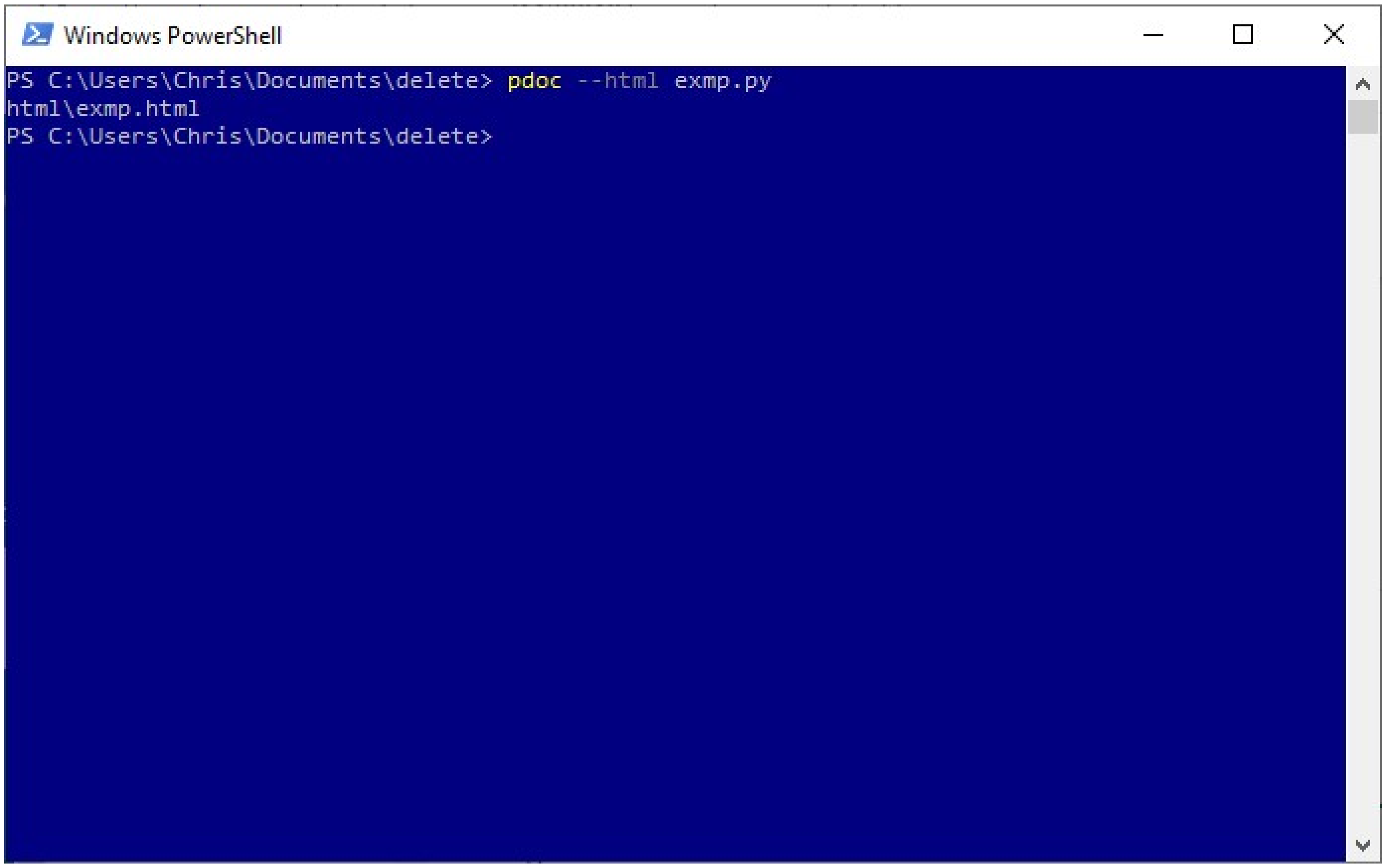
```
def double(x):
    """
    This function will return twice as much as the value passed into it
    Args:
        x (int): the number that needs to be doubled
    Returns:
        the doubled value of what was passed in
    """
    return 2 * x
```

Here's that example above in pictures:



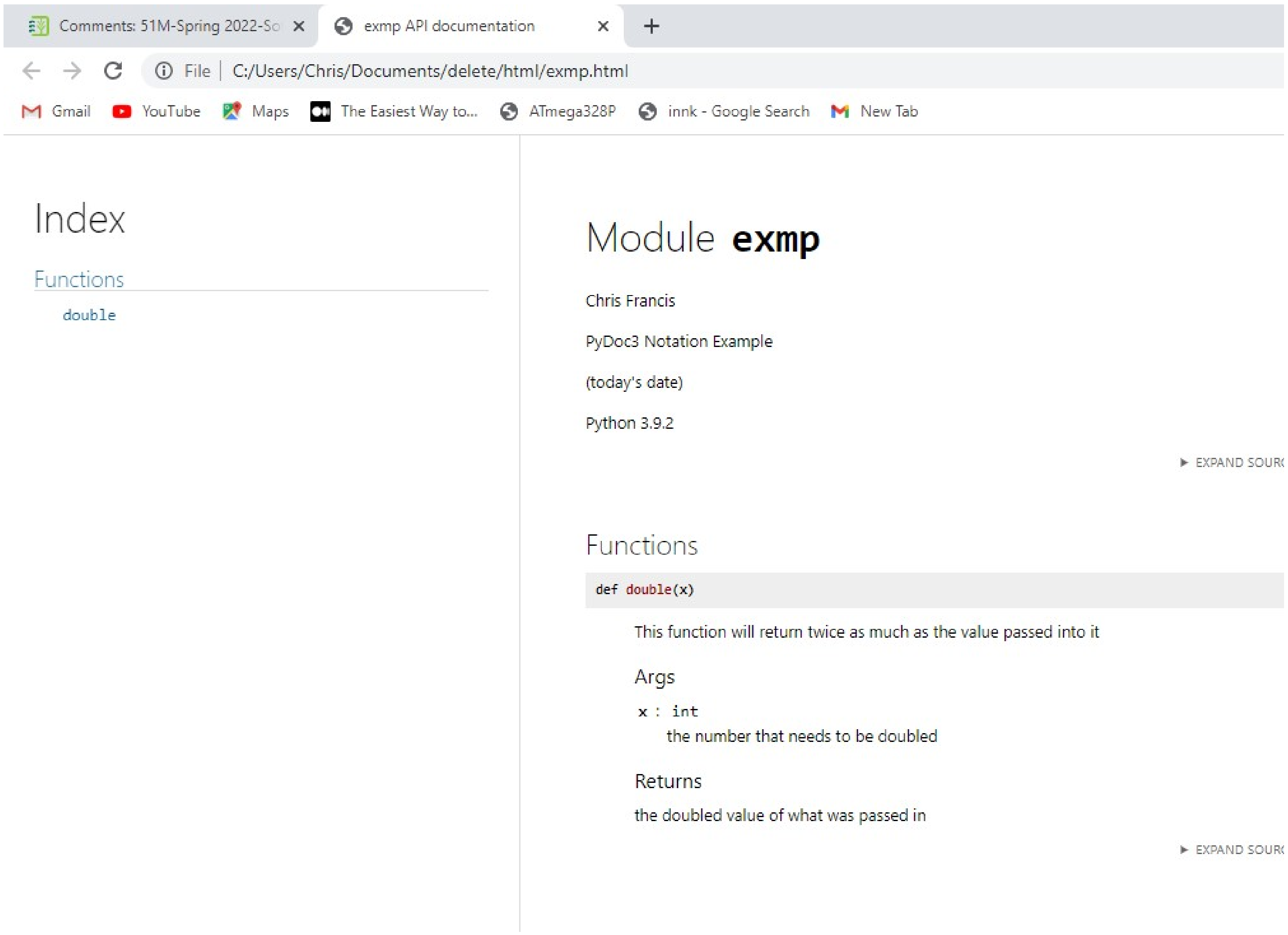
exmp.py — C:\Users\Chris\Documents\delete — Atom

```

"""
Chris Francis

PyDoc3 Notation Example

(today's date)

Python 3.9.2
"""

def double(x):
    """
    This function will return t
    Args:
        x (int): the number tha
    Returns:
        the doubled value of wh
```



Windows PowerShell

```
PS C:\Users\Chris\Documents\delete> pdoc --html exmp.py
html\exmp.html
PS C:\Users\Chris\Documents\delete>
```



Index

Functions

double

Module **exmp**

Chris Francis

PyDoc3 Notation Example

(today's date)

Python 3.9.2

▸ EXPAND SOUR

Functions

```
def double(x)
```

This function will return twice as much as the value passed into it

Args

`x : int`
the number that needs to be doubled

Returns

the doubled value of what was passed in

▸ EXPAND SOUR

Google Colab Instructions:

Here's what I have found works in Google Colab for this (Jupyter Notebooks online will follow same steps but hosted locally will have direct access to downloads folder so use that instead of dot in step 3), note there's a cell above the screenshot with.

```
pip install pdoc3
```

then the four step process shown in image below

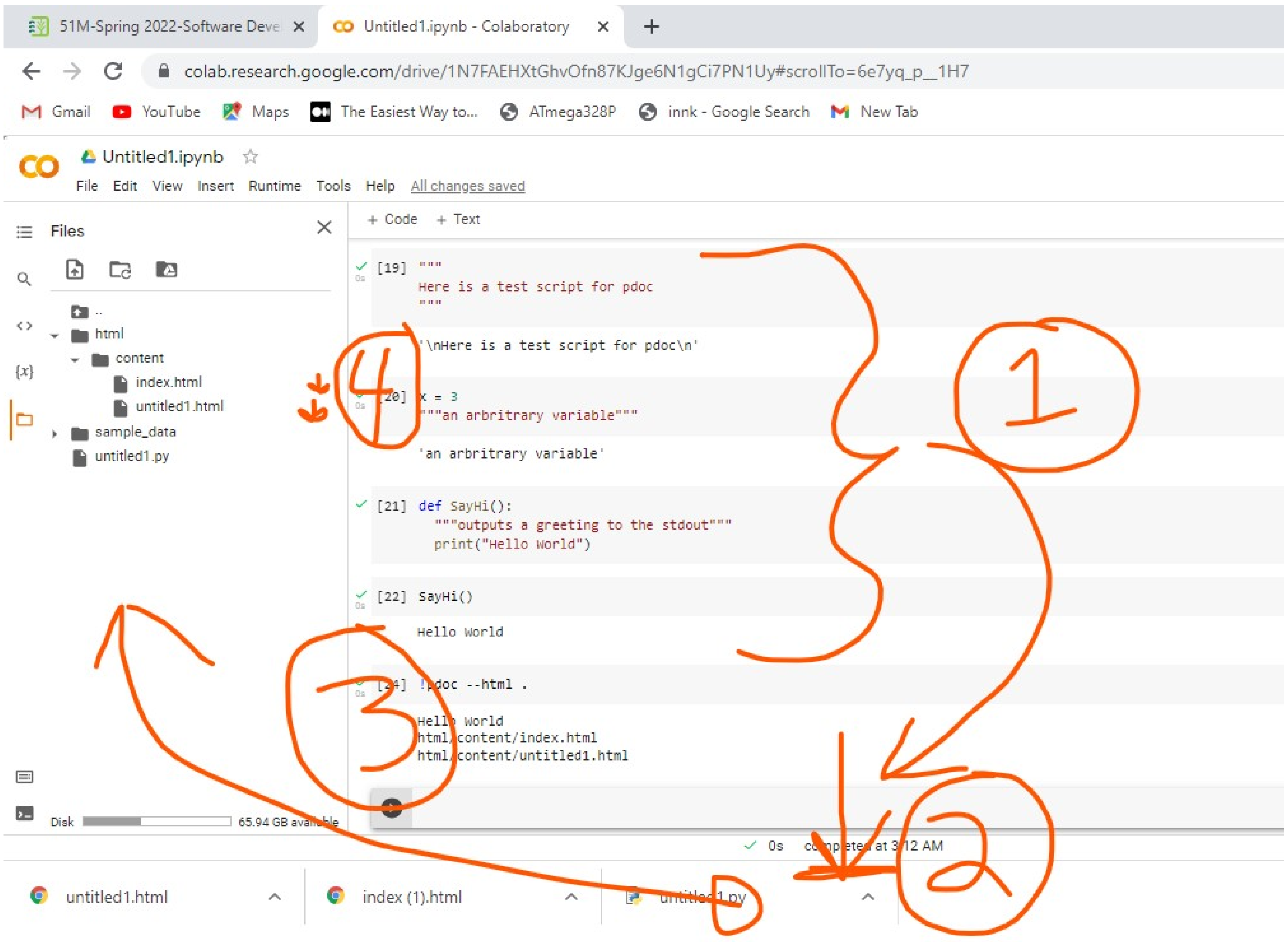
1. when program is finished download as .py file

2. immediately reupload that file to the session storage (file folder icon on left navigation, close and reopen to refresh view)

3. run pdoc with the ! symbol so notebook knows it's a system command not python code, and dot operator for location

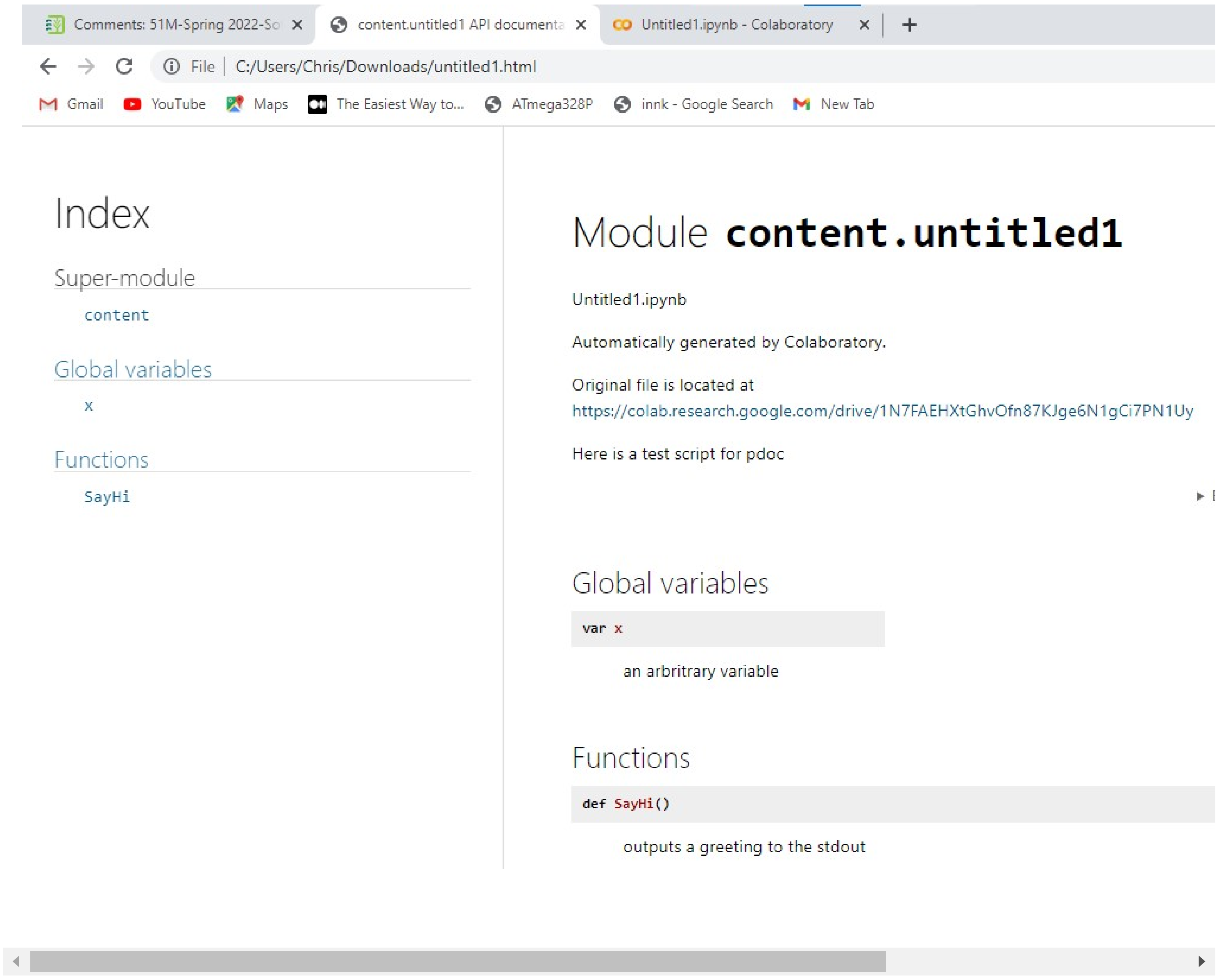
```
!pdoc --html .
```

4. download the html files before ending the session (google will clear session storage)



```
[19] """
     Here is a test script for pdoc
     """

'\nHere is a test script for pdoc\n'

[20] x = 3
     """an arbritrary variable"""

'an arbritrary variable'

[21] def SayHi():
       """outputs a greeting to the stdout"""
       print("Hello World")

[22] SayHi()

Hello World

[24] !pdoc --html .

Hello World
html/content/index.html
html/content/untitled1.html
```

51M-Spring 2022-Software Deve
Untitled1.ipynb - Colaboratory
colab.research.google.com/drive/1N7FAEHXtGhvOfn87KJge6N1gCi7PN1Uy#scrollTo=6e7yq_p__1H7
Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
html
content
index.html
untitled1.html
sample_data
untitled1.py
Disk 65.94 GB available
completed at 3:12 AM
untitled1.html
index (1).html
untitled1.py
1
2
3
4


Comments: 51M-Spring 2022-So
content.untitled1 API documenta
Untitled1.ipynb - Colaboratory
File
C:/Users/Chris/Downloads/untitled1.html
Gmail
YouTube
Maps
The Easiest Way to...
ATmega328P
innk - Google Search
New Tab

Index

Super-module

`content`

Global variables

`x`

Functions

`SayHi`

Module **`content.untitled1`**

Untitled1.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1N7FAEHXtGhvOfn87KJge6N1gCi7PN1Uy

Here is a test script for pdoc

Global variables

```
var x
```

an arbritrary variable

Functions

```
def SayHi()
```

outputs a greeting to the stdout