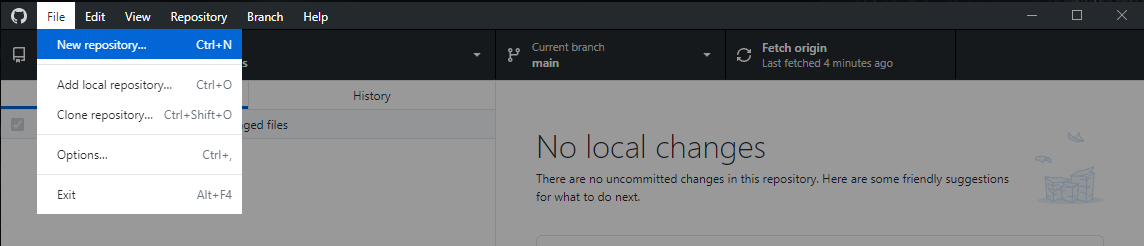
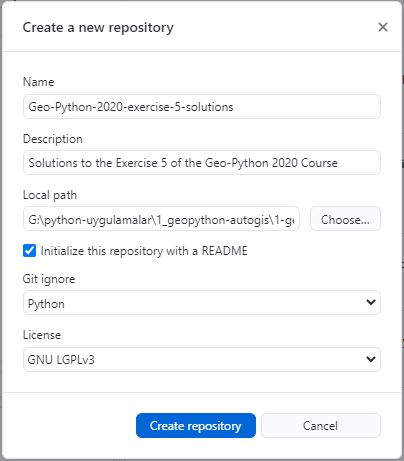
**How to do?**

Dr. Hayati TAŞTAN

September 3, 2021, Ankara

**Download GitHub Desktop for Windows (GitHubDesktopSetup-x64.exe ) from** [**https://desktop.github.com/**](https://desktop.github.com/) **into G:\python-uygulamalar\0\_GitHub\_Desktop\) and install.**

**Create a local repository within GitHub desktop (File / New repository):**

**New repository:**

**Name:** Geo-Python-2020-exercise-5-solutions

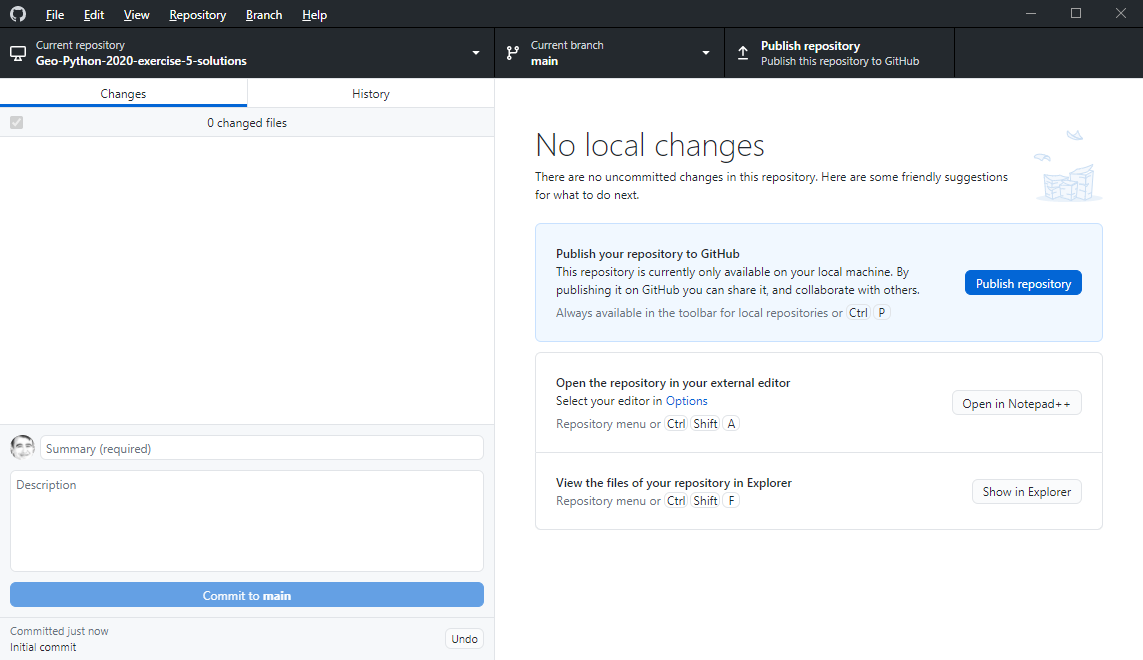
**Description:** Solutions to the Exercise 5 of the Geo-Python 2020 Course.

**Local path:** G:\python-uygulamalar\1\_geopython-autogis\1-geo-python-2020\ornekler

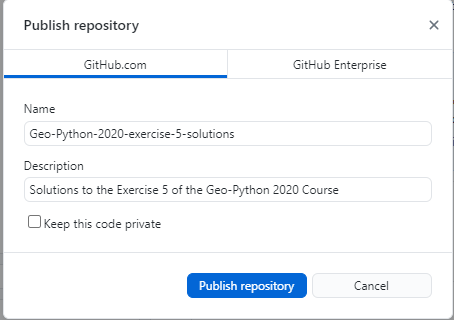
Initialize the reposiyory with a README

**Git ignore:** Python

**License:** GNU LGPLv3

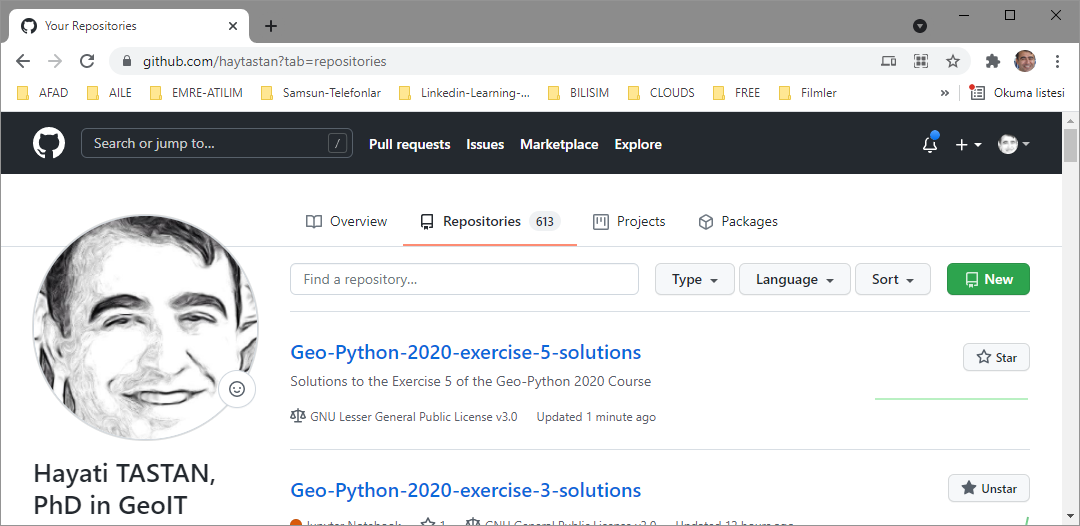
**Create repository:**

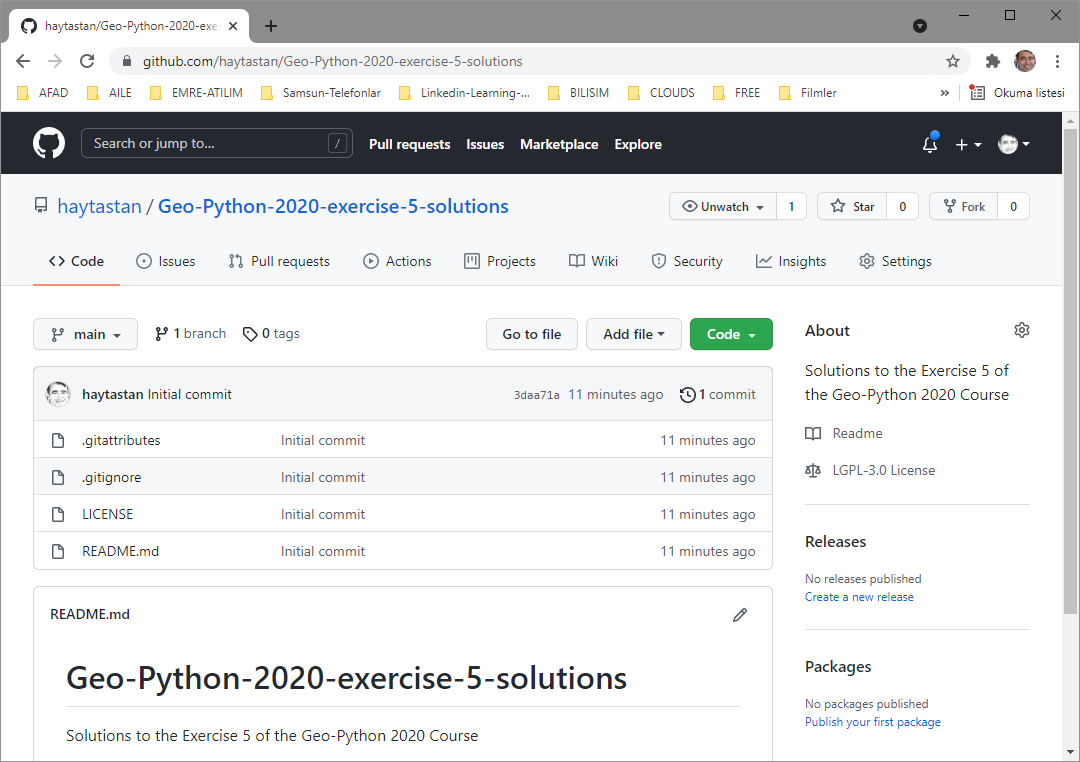
**Publish reposiyory:**

****

**Uncheck Keep this code private**

**Publish repository**

**Sign in to** [**https://github.com/haytastan**](https://github.com/haytastan)

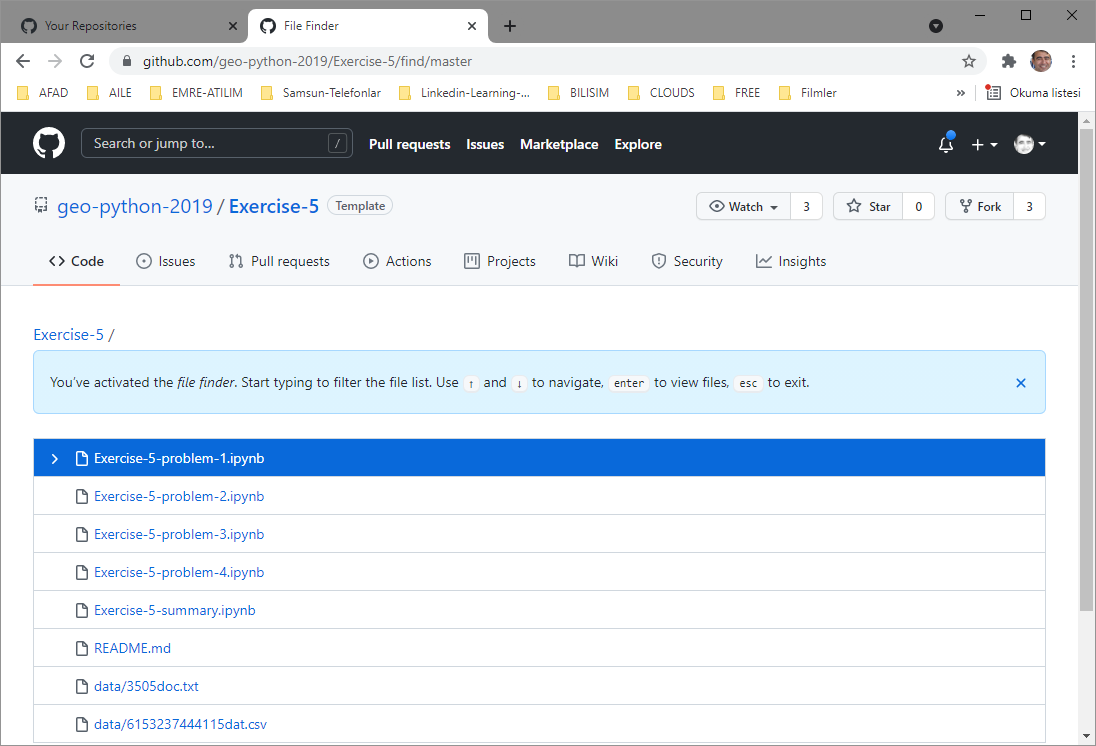
 **Enter the repo** Geo-Python-2020-exercise-5-solutions:

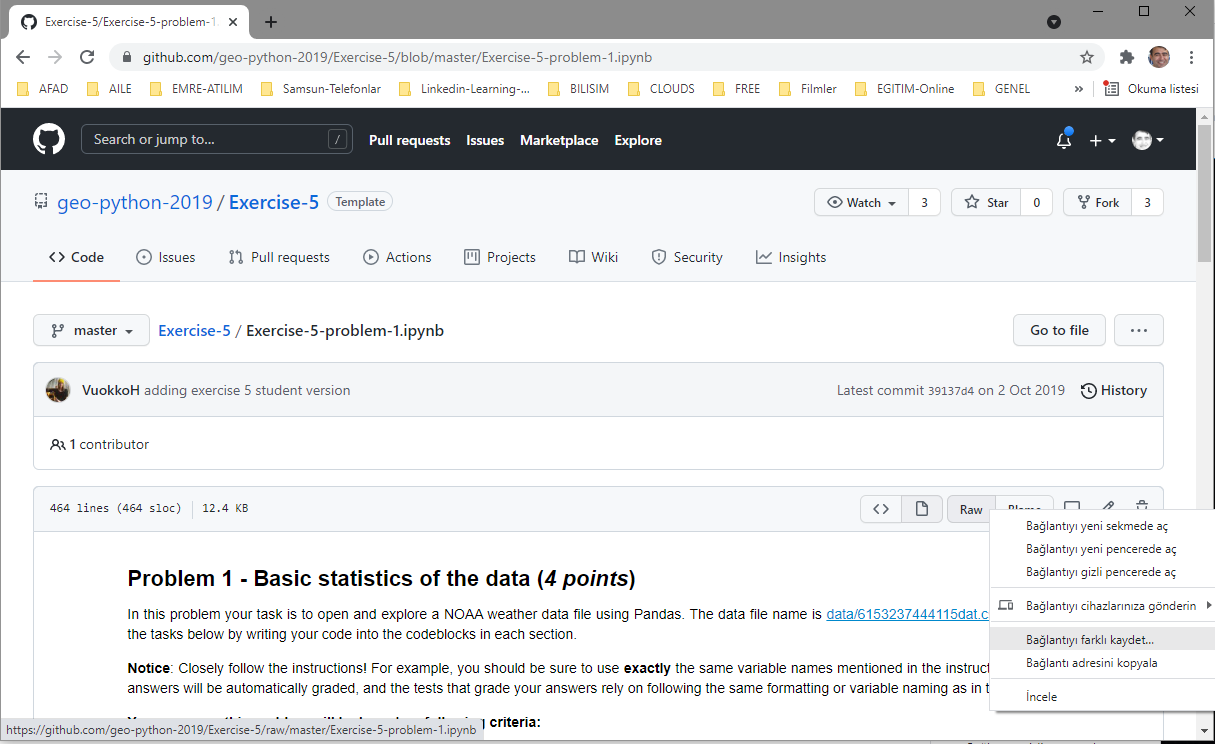
**Download jupyter projects (\*.ipynb) and README.md files from:**

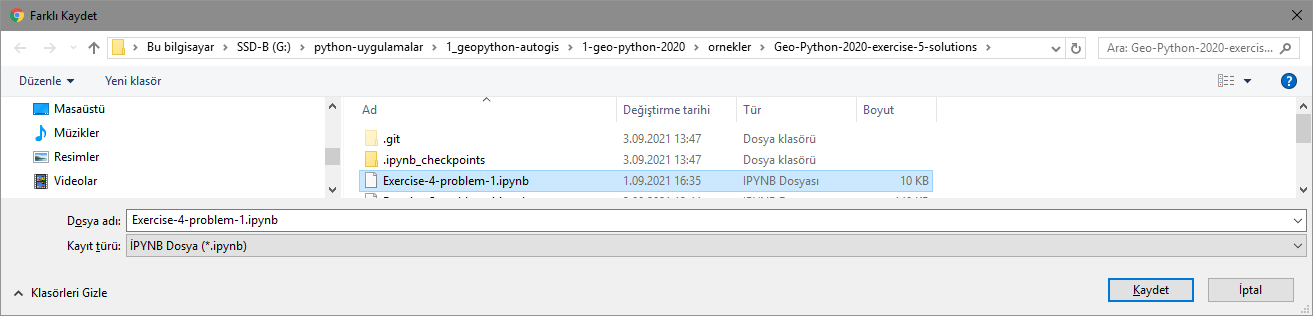
<https://github.com/geo-python-2019/Exercise-5>

**into the local repo:**

G:\python-uygulamalar\1\_geopython-autogis\1-geo-python-2020\ornekler

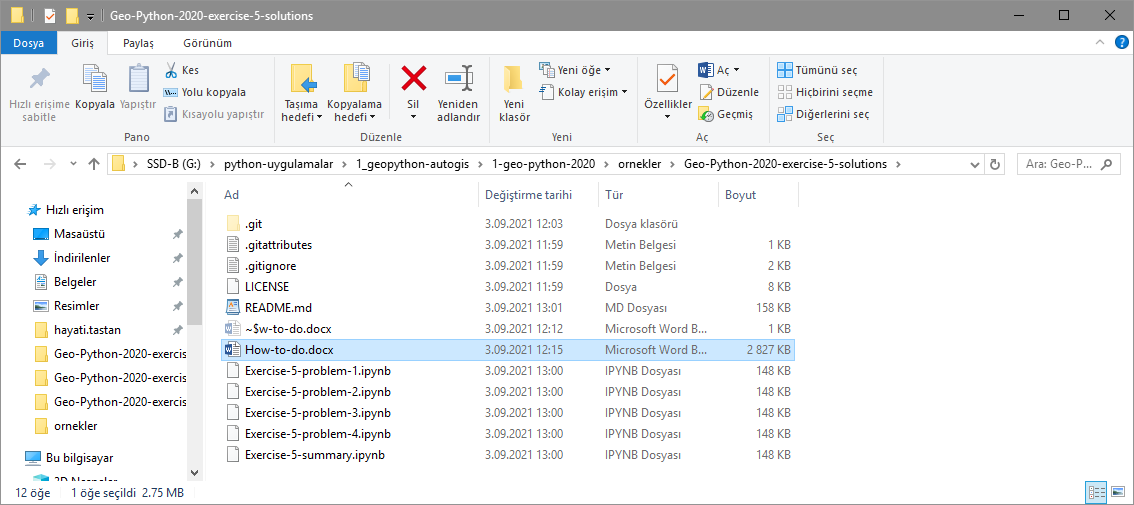
**Go to file:**

**Select the file to download, double click, goto to the ikon raw, mouse right click and save as:**

**Save as with the same name (i.e: Exercise-4-problem-1.ipynb)**

**Goto** <https://github.com/geo-python-2019/Exercise-5/find/master>

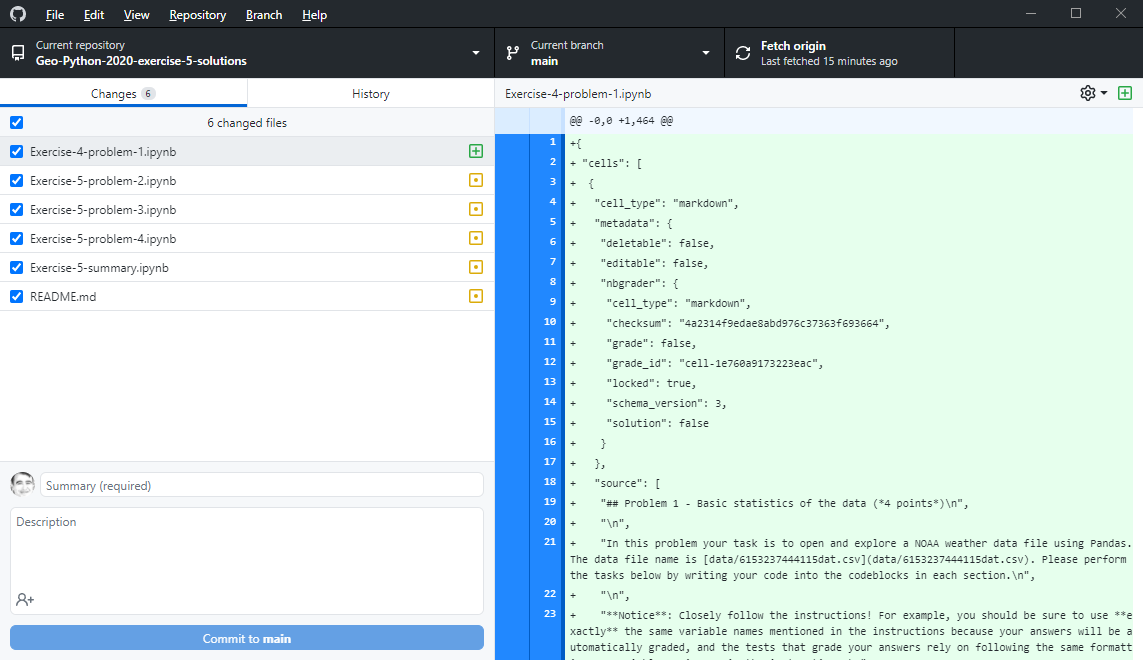
and repeat this copy procedure for all **\*.ipnyb** and **README.md** files.

Files downloaded are:

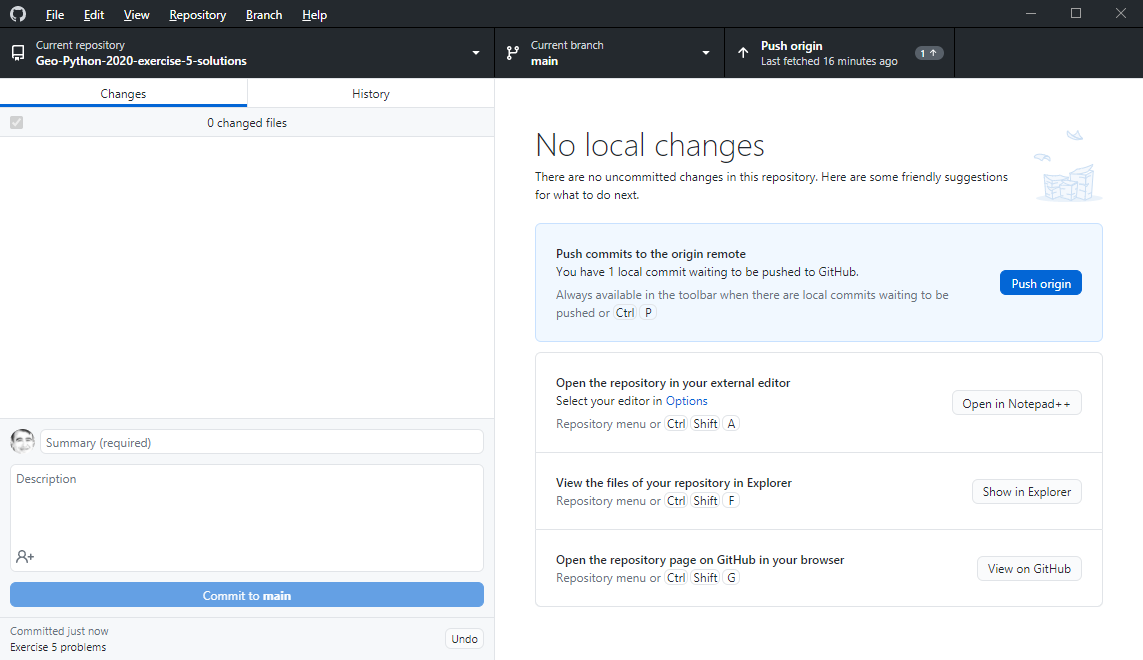
**Goto GitHub Desktop (which is already running):**

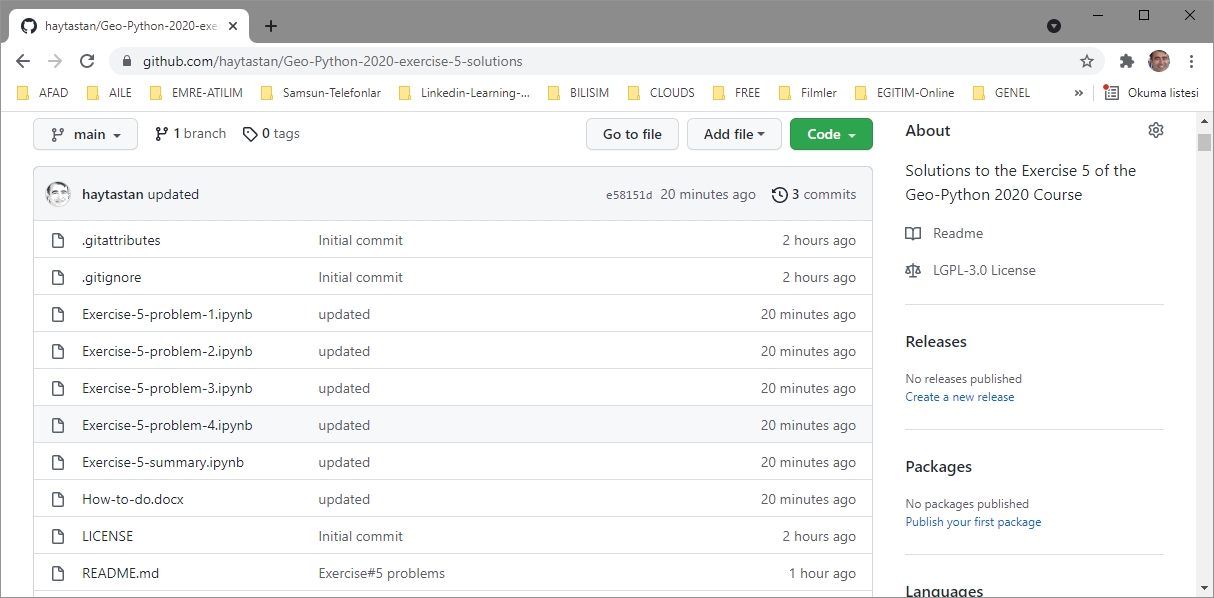
**Write something into Summary and Description boxes as below:**

**Summary:** Exercise 5 problems

**Description:** Downloaded from <https://github.com/geo-python-2019/Exercise-5/find/master>

Commit to main:

**Push origin:**

**Enter the remote repo** <https://github.com/haytastan/Geo-Python-2020-exercise-5-solutions> (if entered already, refresh the browser):

**Let’s start to solve the problems:**

**Run (OSGeo4W Shell if you installed QGIS 3.201) or Run cmd (if you didn’t install QGIS but installed Python 3.9.7 seperately) or Run the shortcut to the OSGeo4W Shell (**i.e. for my case python-qgis-cmd-shotcut which points to python-qgis-cmd.bat (i.e to C:\programs\qgis320\bin\python-qgis-cmd.bat).

The content of python-qgis-cmd.bat is as below:

@**echo** off

**call** "**%~dp0\o4w\_env.bat"**

@**echo** off

**path** **%OSGEO4W\_ROOT%**\apps\qgis\bin;**%PATH%**

**path** **%PATH%**;C:\programs\nodejs

**path** **%PATH%**;C:\programs\Git\cmd

**set** QGIS\_PREFIX\_PATH**=%OSGEO4W\_ROOT:\=/%**/apps/qgis

**set** GDAL\_FILENAME\_IS\_UTF8**=**YES

rem Set VSI cache to be used as buffer, see #6448

**set** VSI\_CACHE**=**TRUE

**set** VSI\_CACHE\_SIZE**=**1000000

**set** QT\_PLUGIN\_PATH**=%OSGEO4W\_ROOT%**\apps\qgis\qtplugins;**%OSGEO4W\_ROOT%**\apps\qt5\plugins

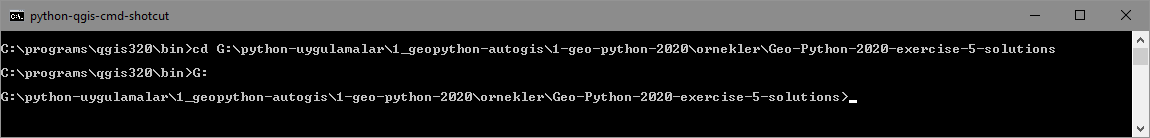
**set** PYTHONPATH**=%OSGEO4W\_ROOT%**\apps\qgis\python;**%PYTHONPATH%**

rem "%PYTHONHOME%\python" %\*

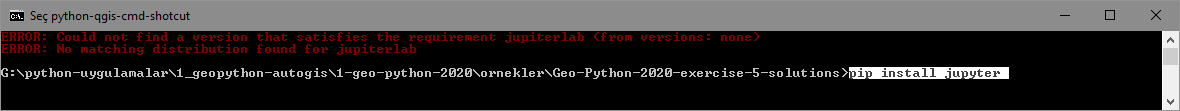
cmd

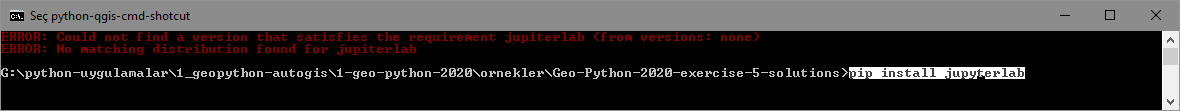
**Change directory to the local repo:**

cd G:\python-uygulamalar\1\_geopython-autogis\1-geo-python-2020\ornekler\Geo-Python-2020-exercise-5-solutions

G:

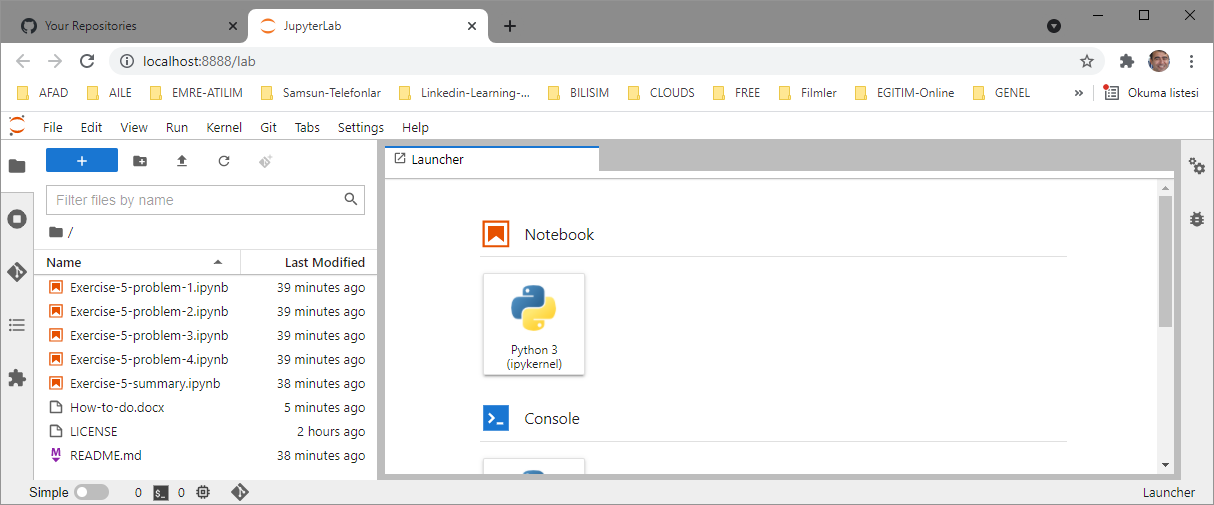
**Install Jupyter Lab (if not installed):**

pip install jupiter

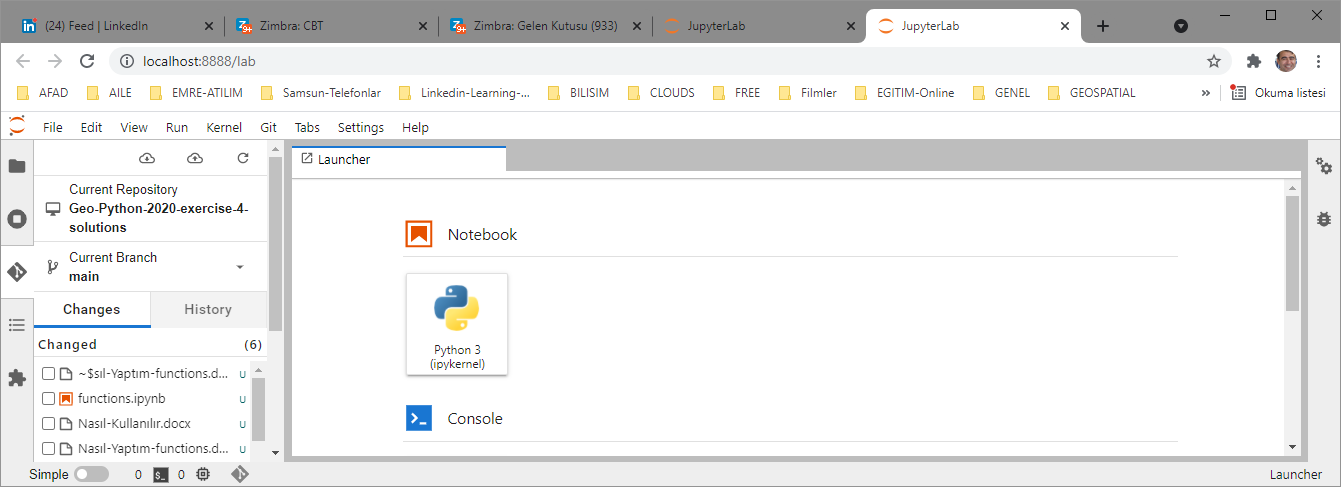
pip install jupyterlab

**Run Jupyter Lab:**

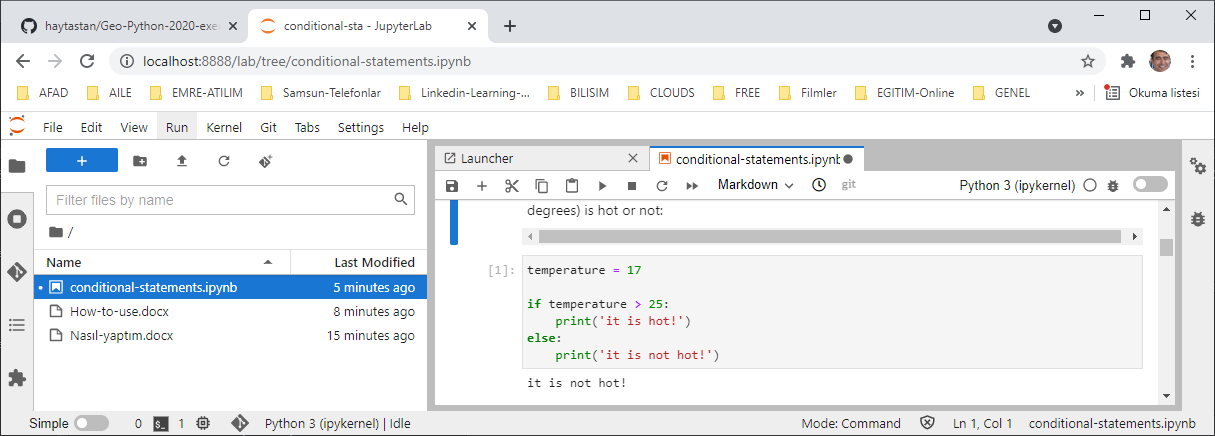
jupyter lab

**A tab for Jupyter lab is created at the browser:**

**Double click one of the Jupiter projects (e.g. Exercise-5-problem-1.ipnyb) in the local repo:**

Jupyter lab

**functions.ipynb** isimli jupyter projesine çift tıklayalım:



Yukarıdaki sayfadaki kutular üzerine gelip, SHIFT+ENTER basarak kutu içindeki python komutlarını çalıştır (veya üstteki > düğmesi ile çalıştır)

**Geo-Python-2020-exercise-3- solutions** isiminde açılacak github reposuna konmak üzere

**Örnek uygulama dosyaları :**

Yukarıdaki kodları **jupyter lab** ortamında çalıştıralım:

QGIS Shell (veya G:\python-uygulamalar\python-qgis-cmd-kısayol çalıştır)

Cd G:\python-uygulamalar\1\_geopython-autogis\1-geo-python-2020\ornekler\Geo-Python-2020-exercise-4-solutions

G:

Jupyter lab

Soldaki klasör ikonunu seçip, yerel repoda **functions.py** isimli bir dosya oluşturalım ve içine functions.ipynb içindeki kutulardan kapyalarak) aşağıdaki kodu yazalım:

def hello(name, age):

return 'Hello, my name is ' + name + '. I am ' + str(age) + ' years old.'

output = hello(name='Hayati TAŞTAN', age=35)

print(output)

def celsius\_to\_fahr(temp):

return 9/5 \* temp + 32

def kelvins\_to\_celsius(temp\_kelvins):

return temp\_kelvins - 273.15

freezing\_point = celsius\_to\_fahr(0)

print('The freezing point of water in Fahrenheit is:', freezing\_point)

print('The boiling point of water in Fahrenheit is:', celsius\_to\_fahr(100))

absolute\_zero = kelvins\_to\_celsius(temp\_kelvins=0)

print('Absolute zero in Celsius is:', absolute\_zero)

def kelvins\_to\_fahr(temp\_kelvins):

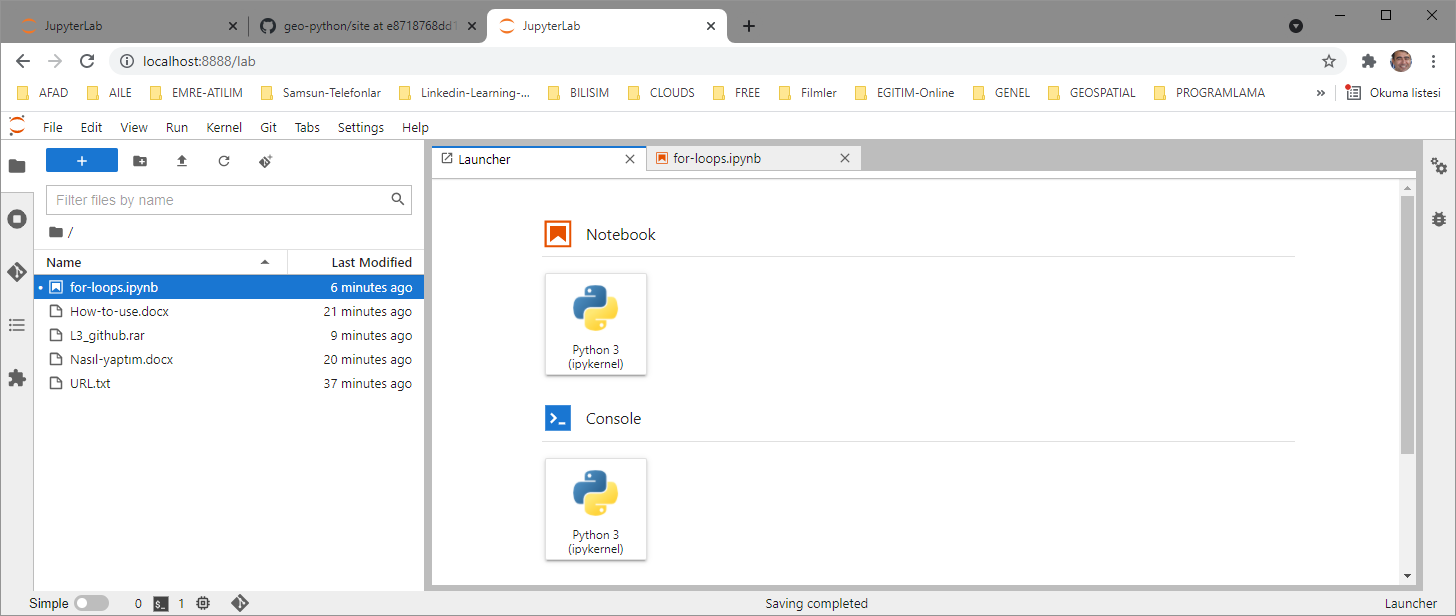
temp\_celsius = kelvins\_to\_celsius(temp\_kelvins)

temp\_fahr = celsius\_to\_fahr(temp\_celsius)

return temp\_fahr

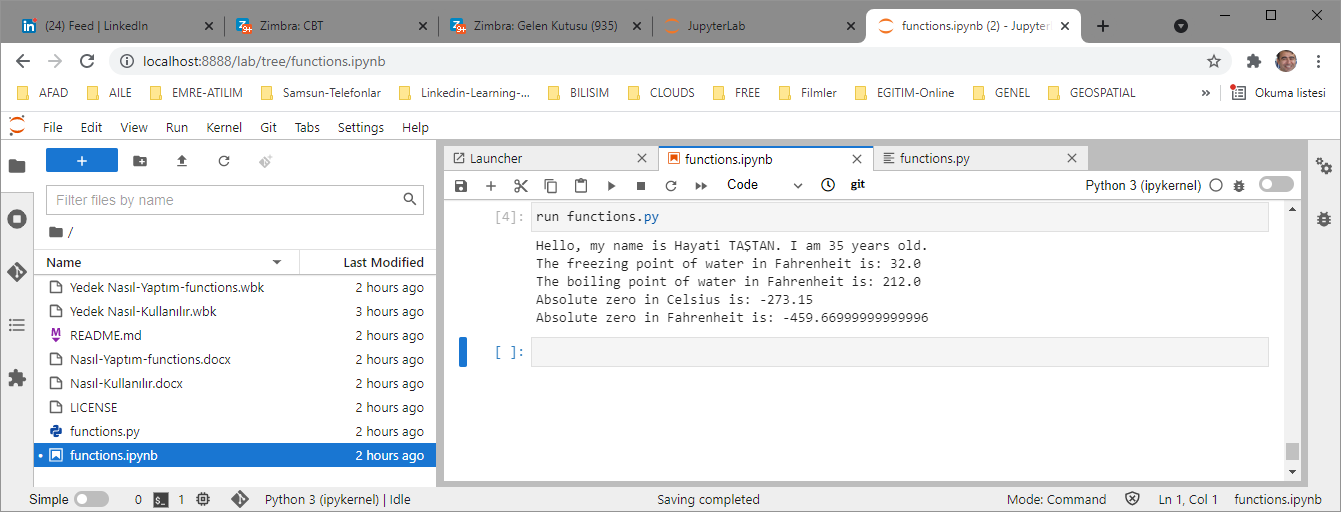
absolute\_zero\_fahr = kelvins\_to\_fahr(temp\_kelvins=0)

print('Absolute zero in Fahrenheit is:', absolute\_zero\_fahr)

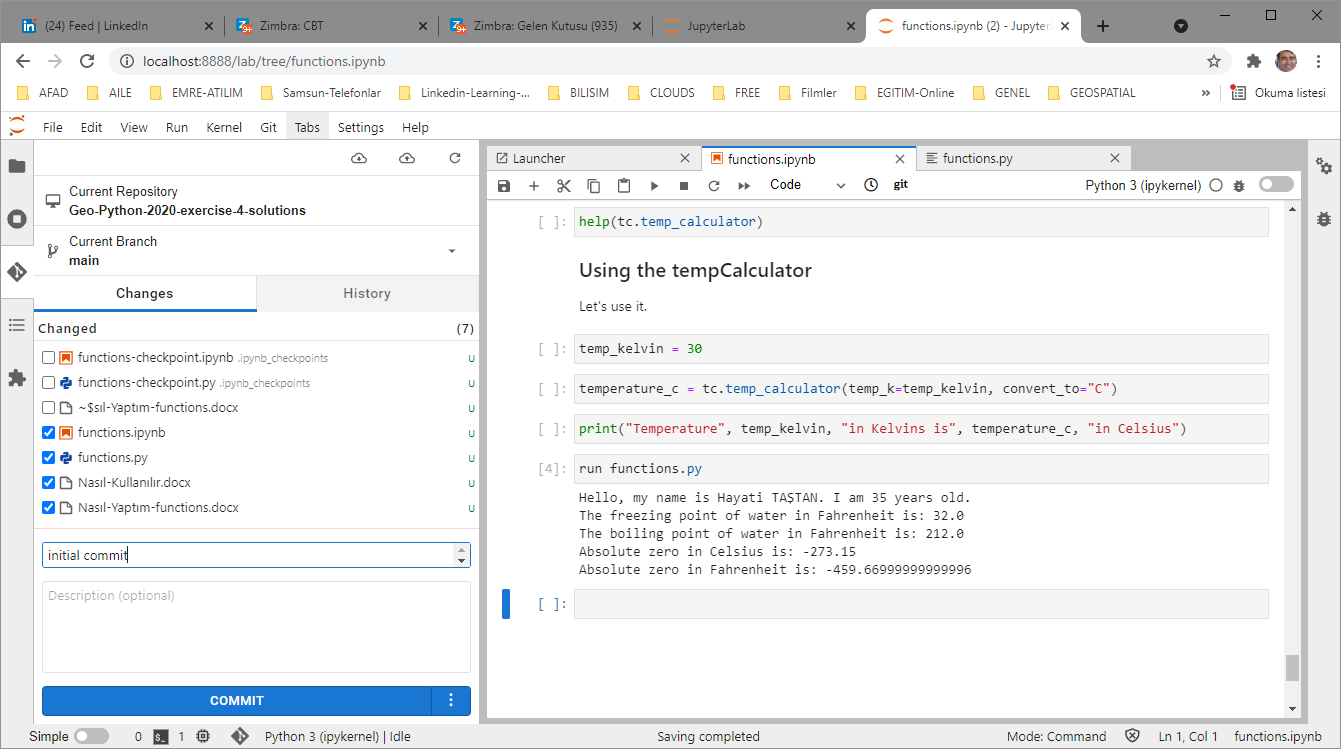


**functions.ipynb** proje dosyasına çift tıklatılır, üstteki + ikonuna tıklanarak boş bit komut kutusu açılır ve açılan kutu içine aşağıdaki komut yazılır:

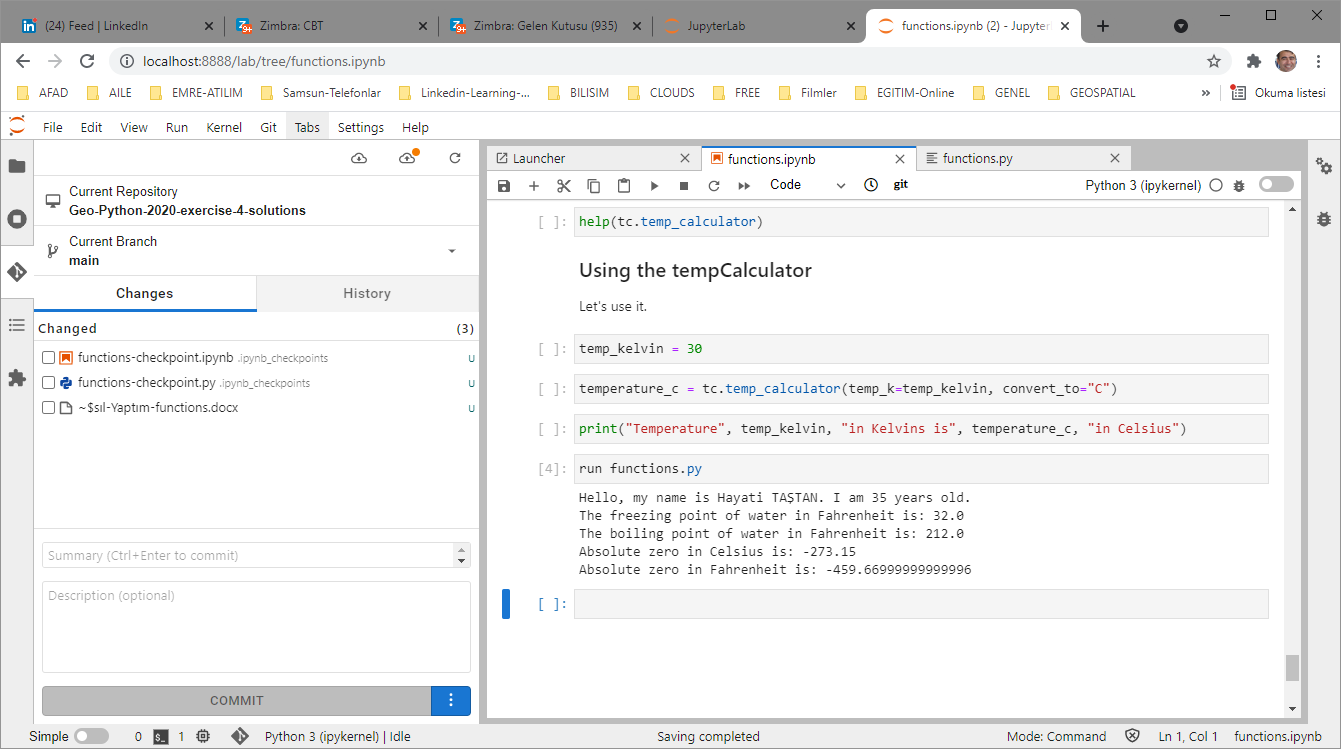
**run functions.py**

ve imleç kutu içinde iken **SHIFT+ENTER** tuşlarına basılara komut çalıştırılır:

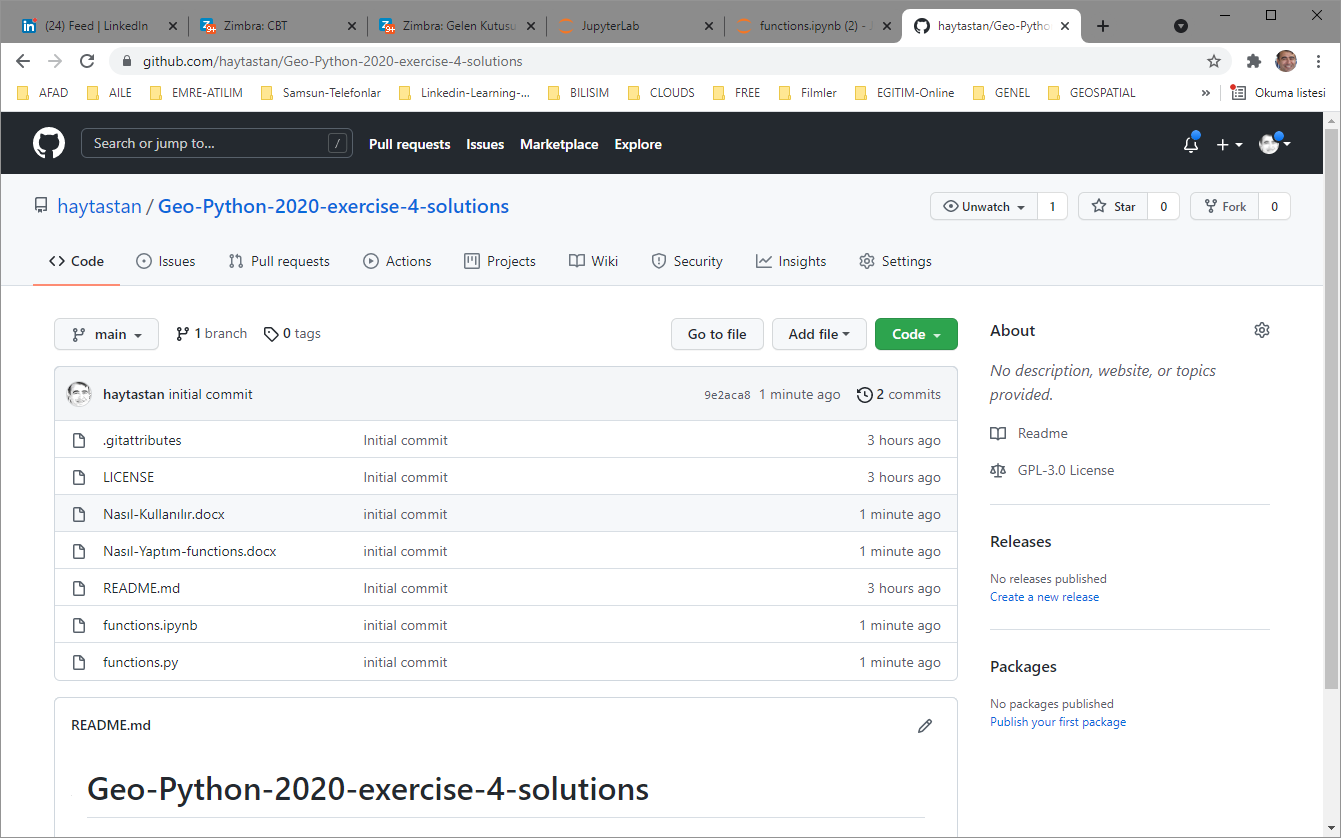
**functions.py dosyasını github’a aktarma:**

Üstteki menüde Git/Simple staging seçeneği seçili olsun. Sol panodaki **Git** ikonuna tıklanıp**, functions.py** seçilir.

Üstteki **summary kutusu içine** initial commit yazılır. Ve en alttaki **COMMIT** düğmesine tıklanarak değişiklikler, dosya push için hazır hale getirilir:

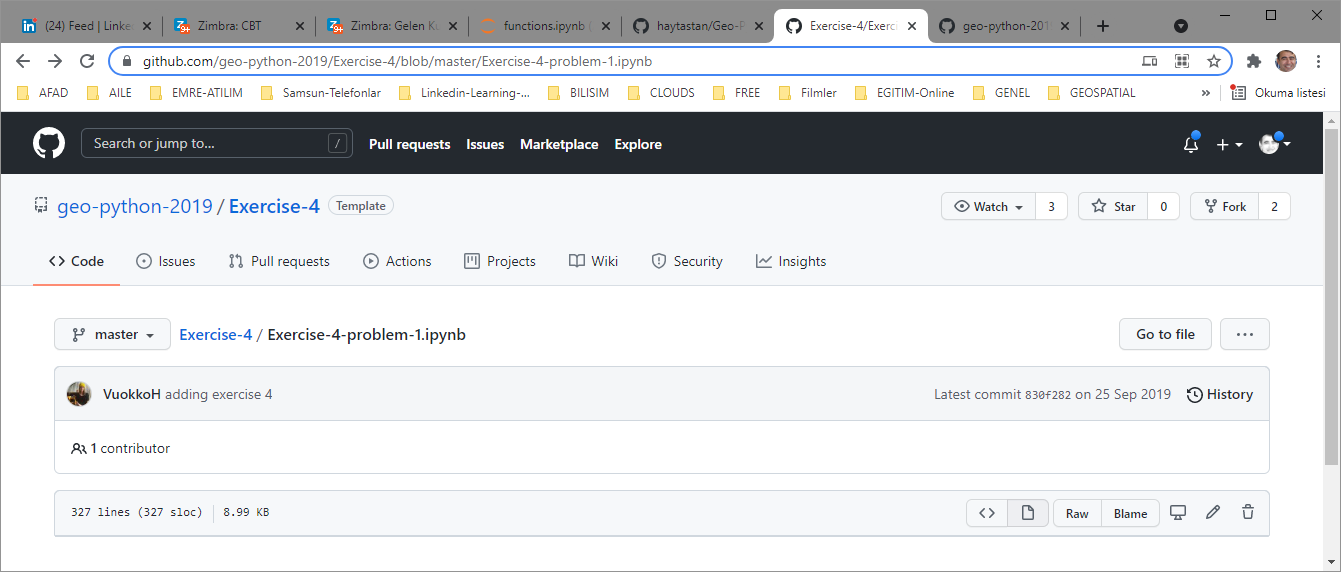


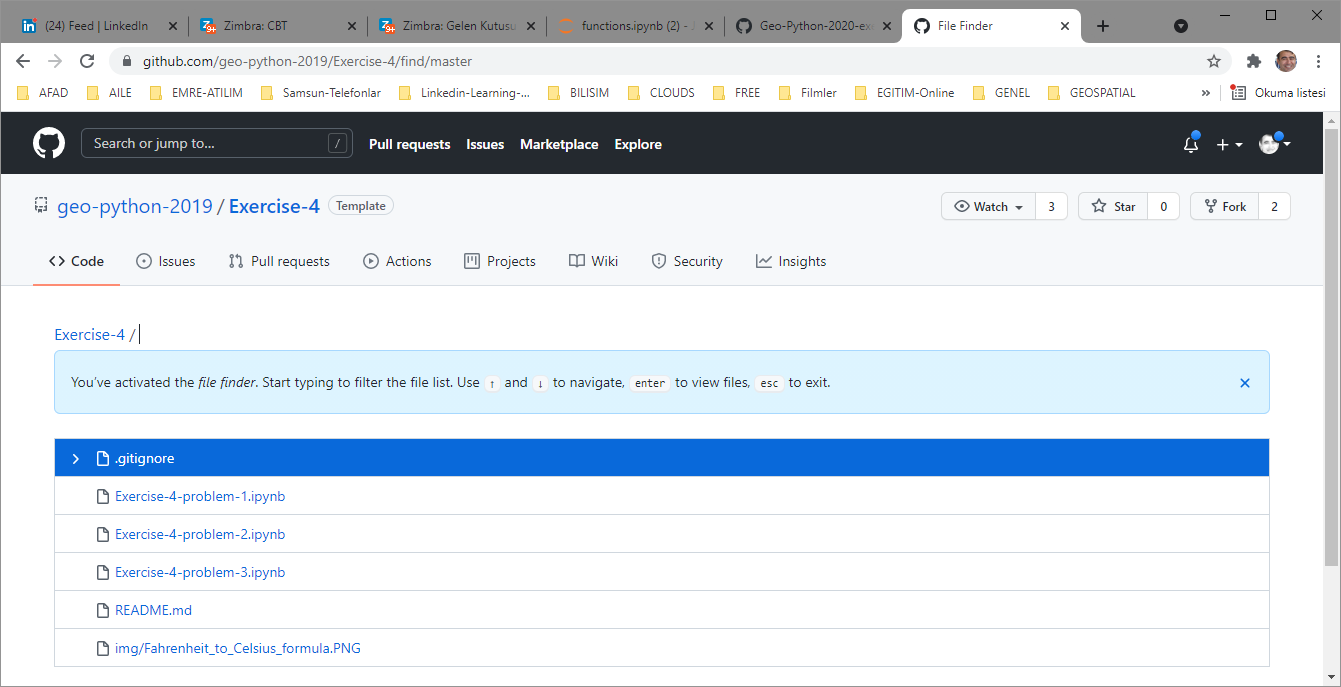
Üstteki **Git/Push to remote** menüsü ile **stage area**’daki değişiklikler uzaktaki **github** reposuna gönderilir.

Github reposunda, **functions.py** görülür:

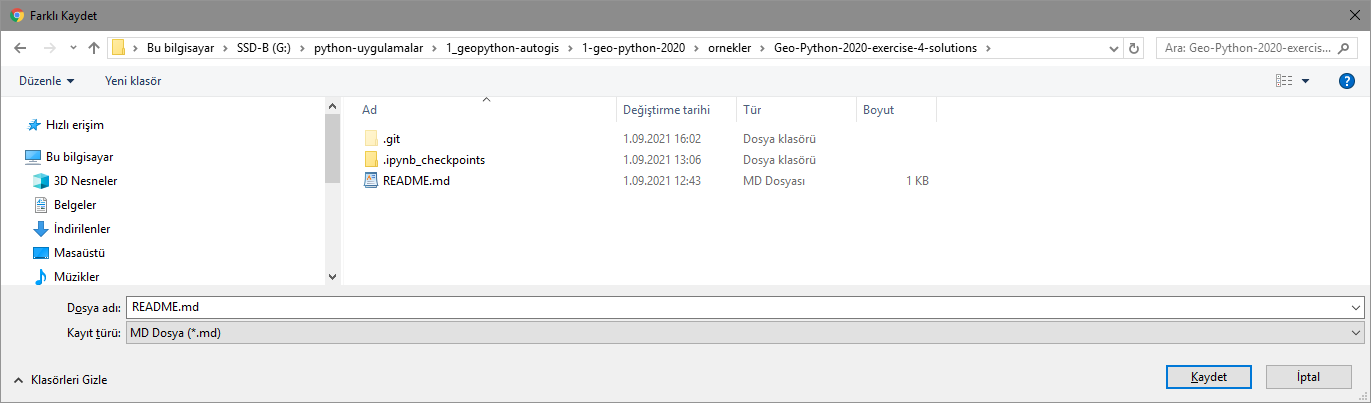
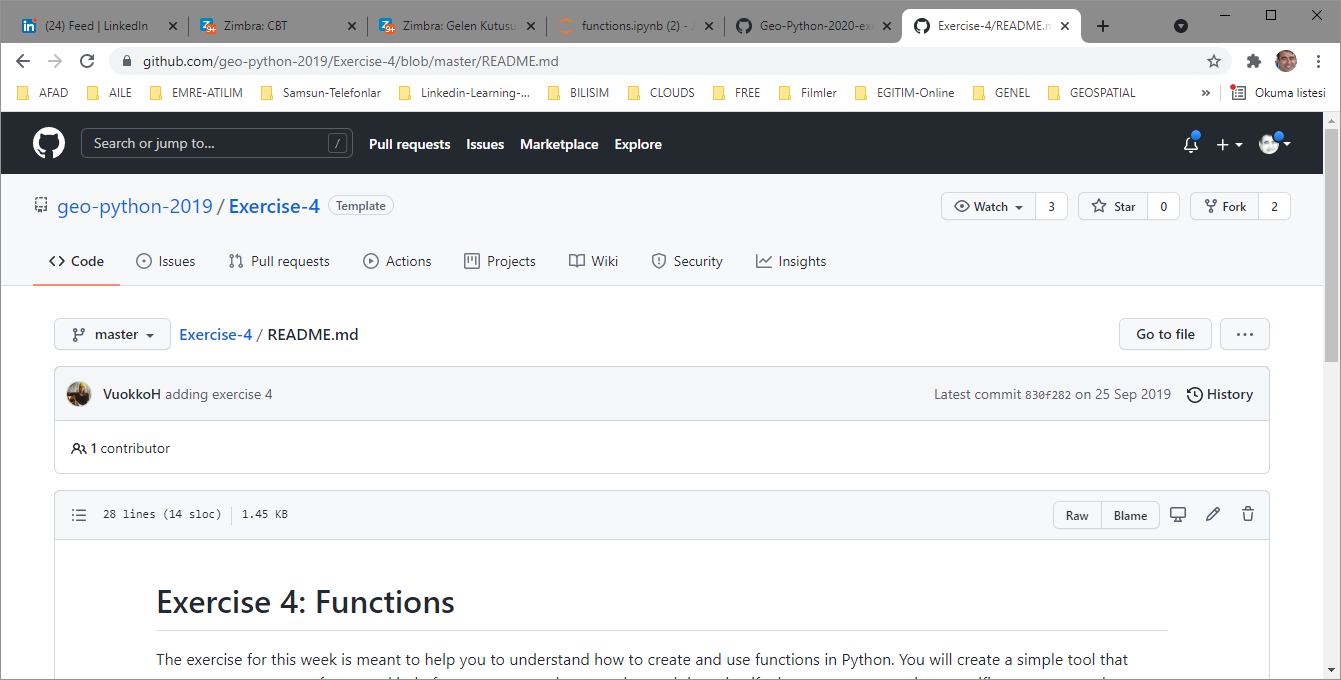
**Kurs adresindeki (**<https://github.com/geo-python-2019/Exercise-4> **) README.md dosyasını kişisel github reposu (**[**https://github.com/haytastan/Geo-Python-2020-exercise-4-solutions**](https://github.com/haytastan/Geo-Python-2020-exercise-4-solutions)**) içine aktarma:**

[**https://github.com/geo-python-2019/Exercise-4/blob/master/Exercise-4-problem-1.ipynb**](https://github.com/geo-python-2019/Exercise-4/blob/master/Exercise-4-problem-1.ipynb)adresindeni üstteki “Goto file” seçin:

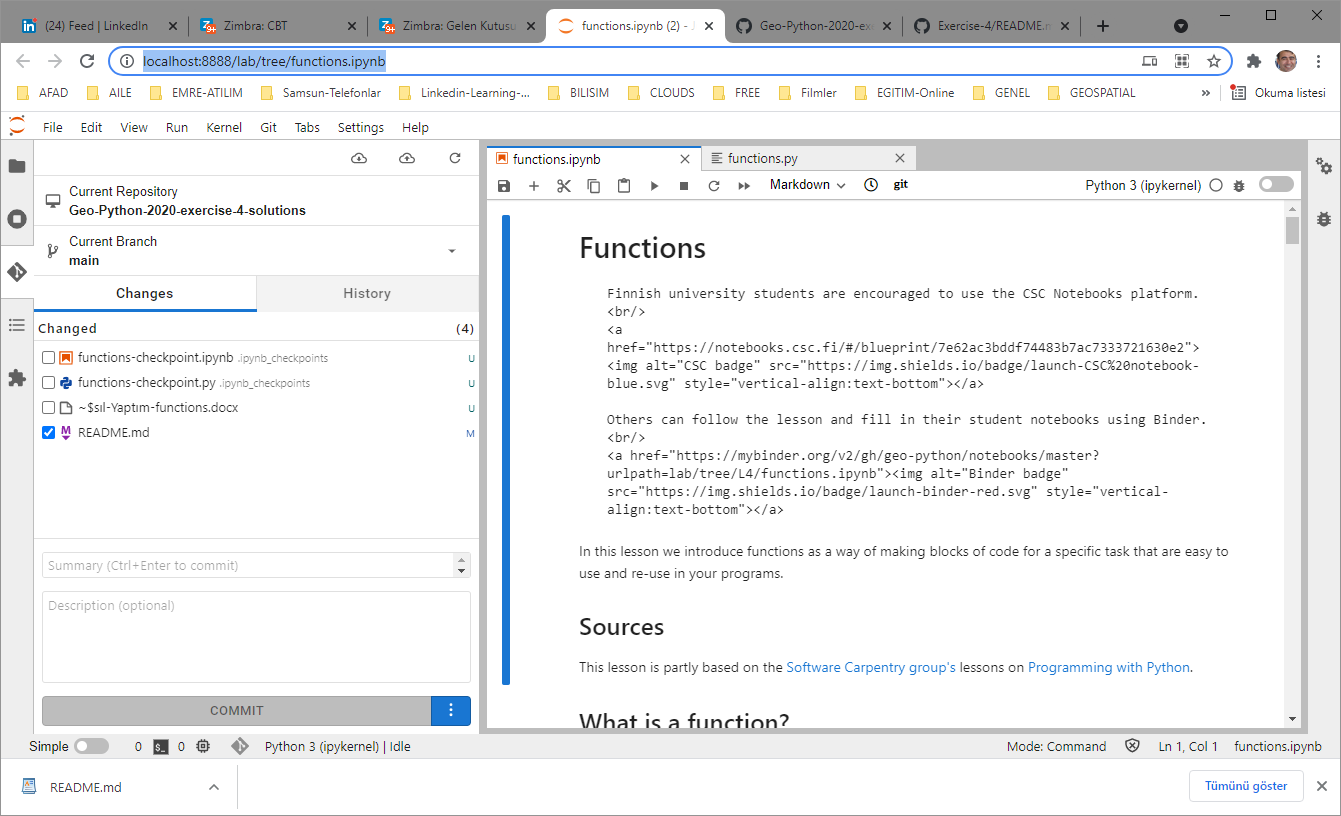


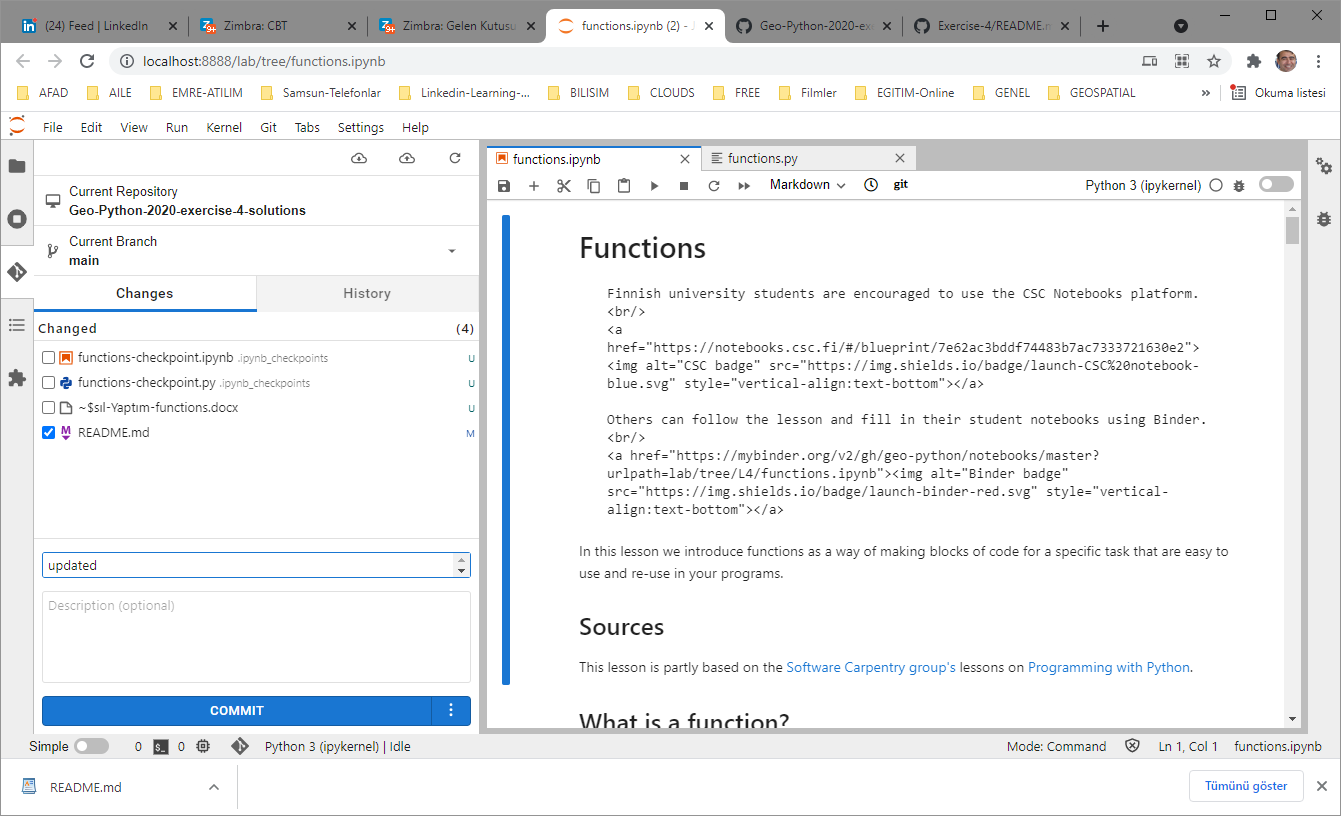
**Goto file:**

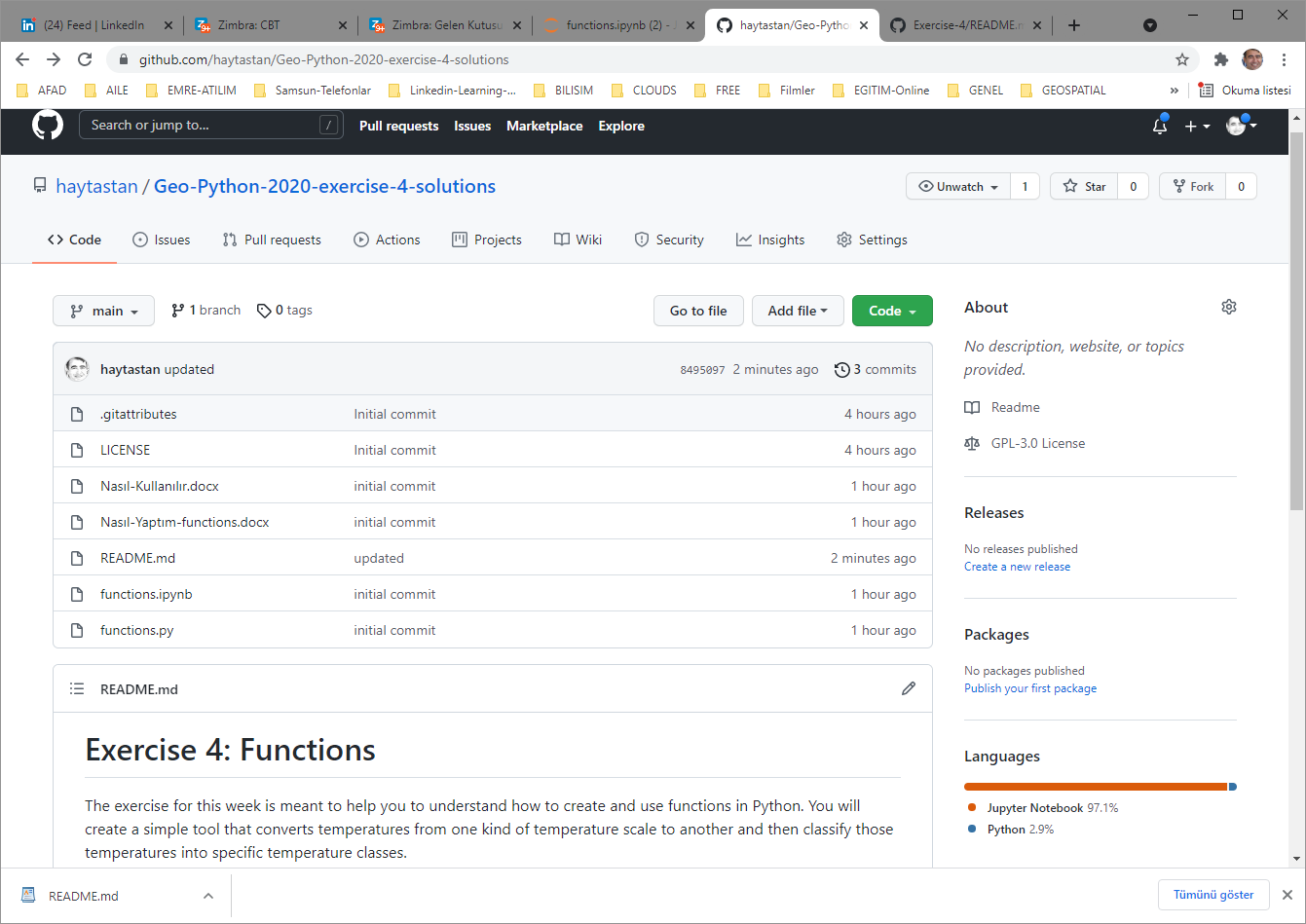
**README.md dosyasına tıklayın:**

“**Raw**” ikonuna üzerine gelip mouse sağ tuşu basıp, “**Farklı kaydet**” komutu ile **README.md** dosyasını yerel repo (**G:\python-uygulamalar\1\_geopython-autogis\1-geo-python-2020\ornekler\Geo-Python-2020-exercise-4-solutions**) içine inidirin:

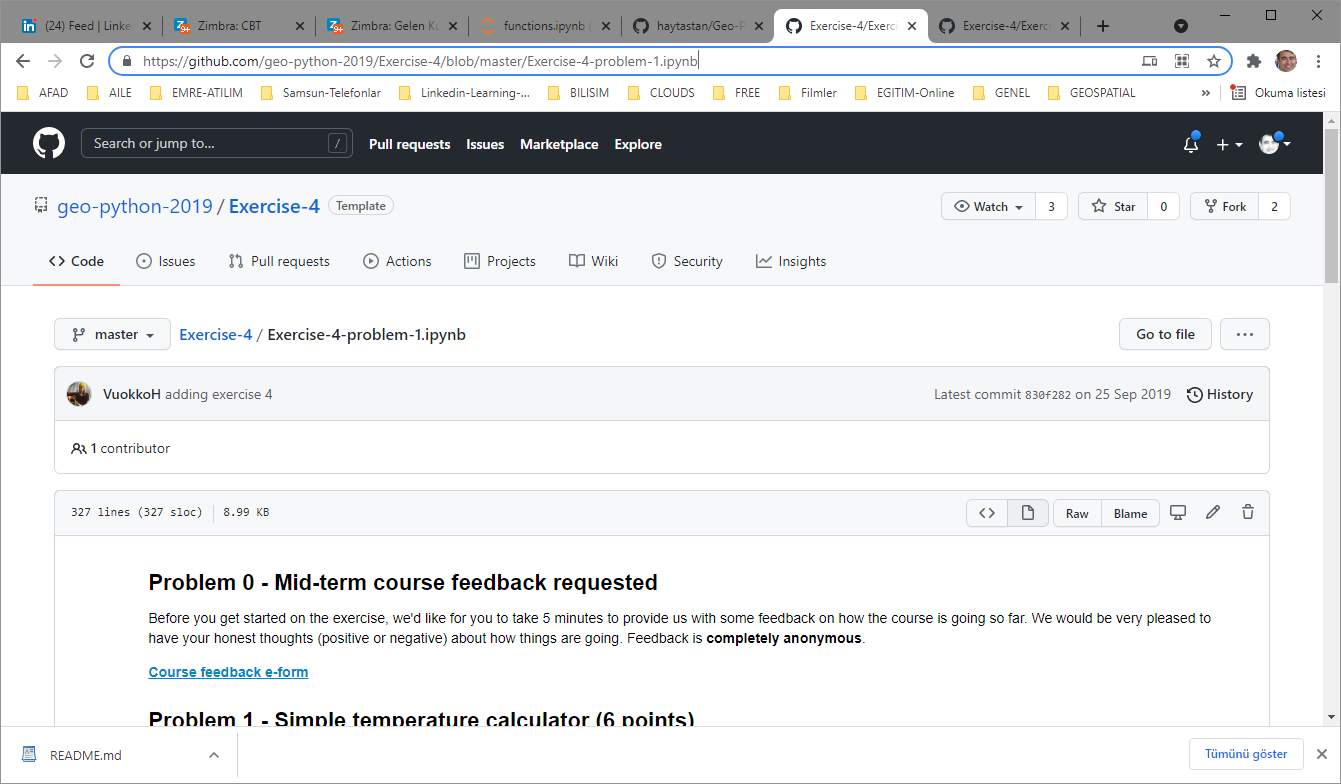
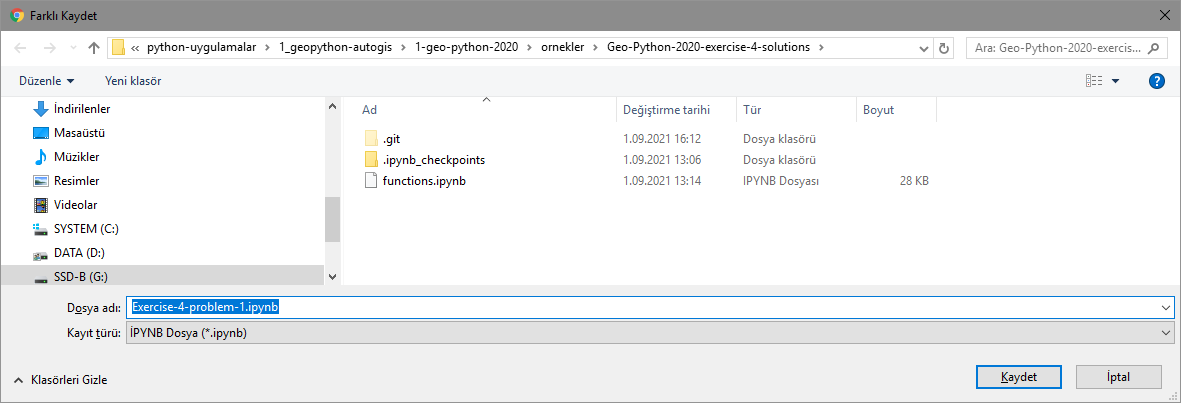
Kaydet:

<http://localhost:8888/lab/tree/functions.ipynb> adresindeki proje dosyasına tıkladığınızda README.md dosyası yerel repo içinde görünecektir:

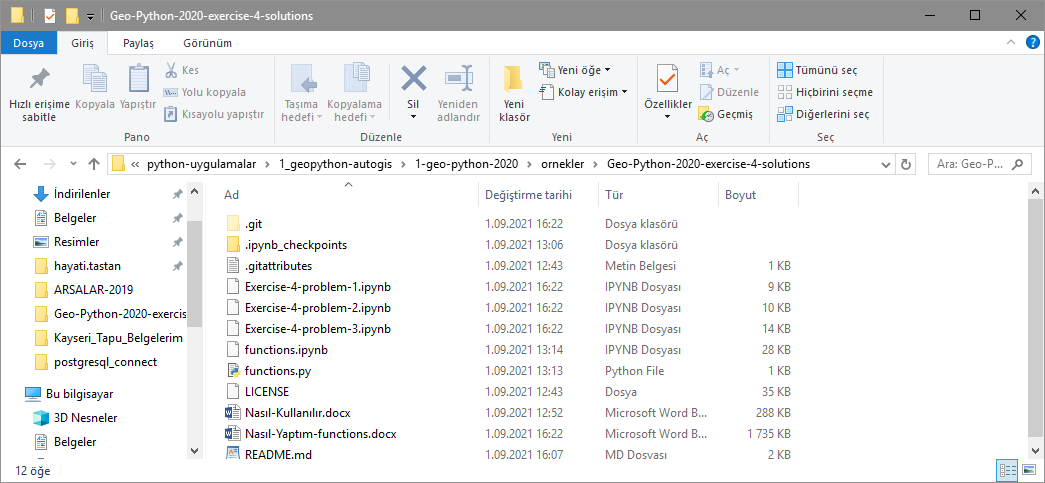
Dikkat edilirse **README.md** dosyası Git/Changes panosu içinde mavi işaretlidir (Yani değişikliğe uyramıştır). Soldaki panoda ilk kutu içine “updated” yazıp aşağıdaki **COMMIT** düğmesine tıklayın:

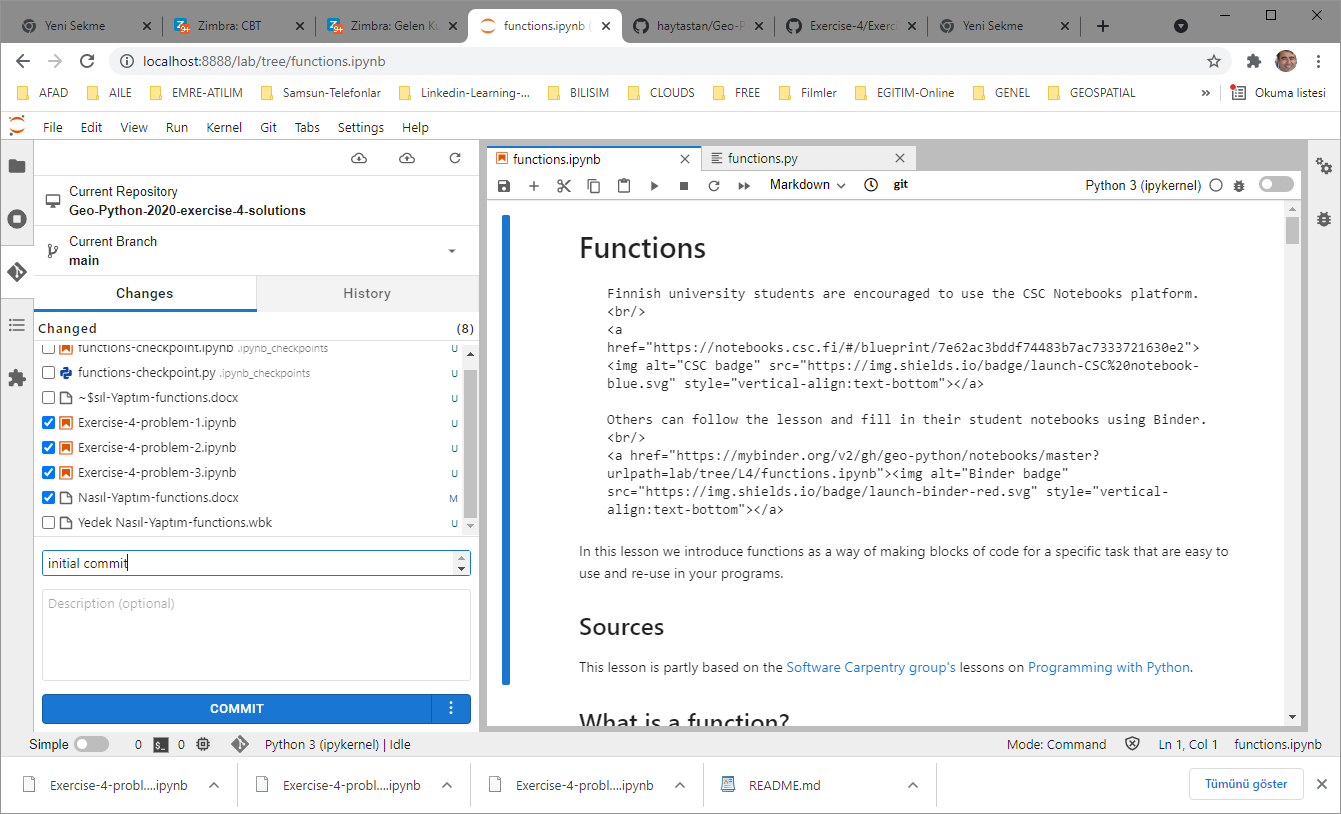
Şimdi en üstteki Git/Push to remote menüsü tıklanırsa, yereldeki REDAME.md dosyasının uzaktaki (GitHub) içindeki repo içine gönderildiği görülür:

**Kurs adresindeki (**<https://github.com/geo-python-2019/Exercise-4> **) Jupyter proje dosyalarının (**Exercise-4-problem-1.ipynb, Exercise-4-problem-2.ipynb, Exercise-4-problem-3.ipynb**), kişisel github reposu (**[**https://github.com/haytastan/Geo-Python-2020-exercise-4-solutions**](https://github.com/haytastan/Geo-Python-2020-exercise-4-solutions)**) içine indirme:**

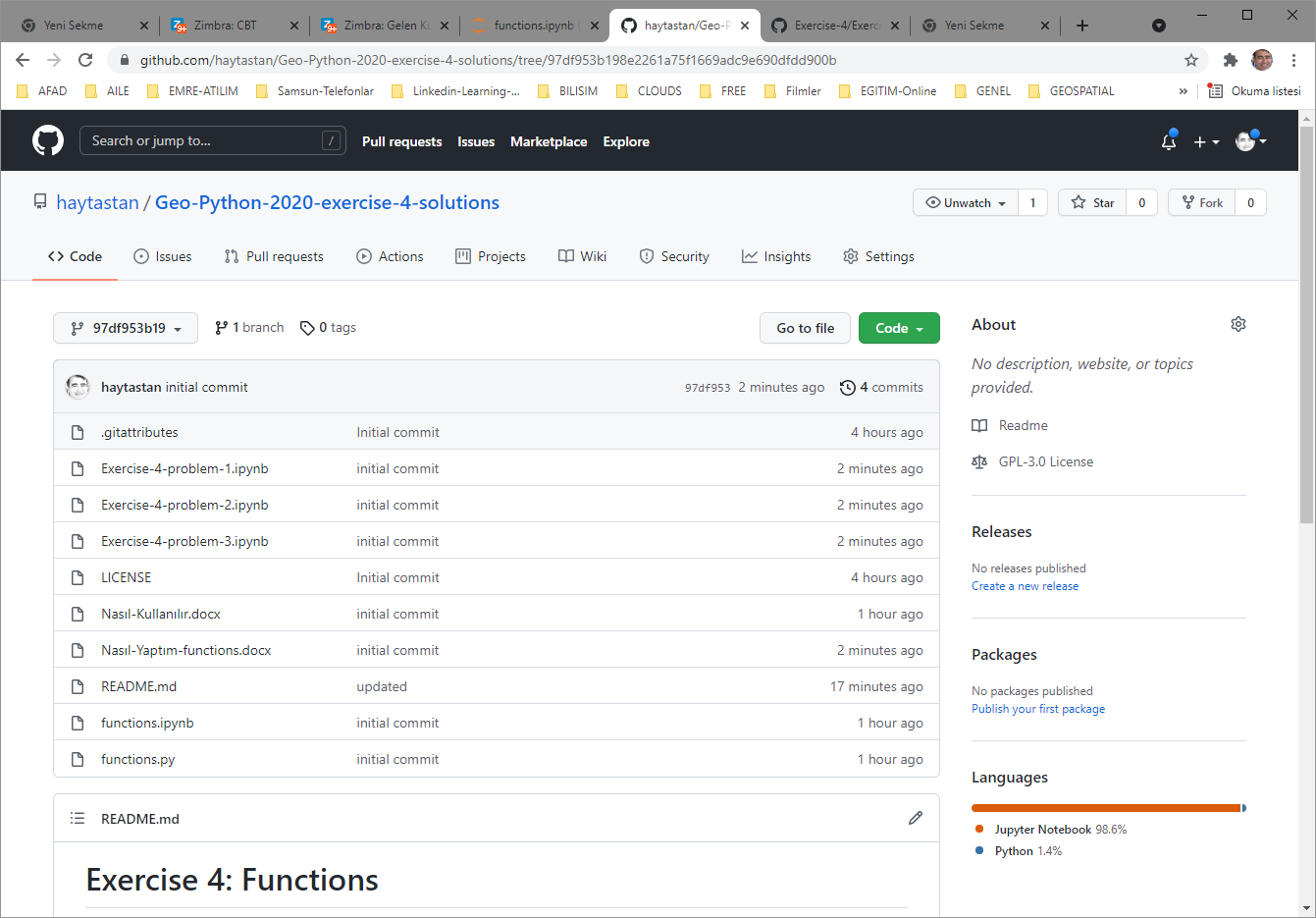
1. <https://github.com/geo-python-2019/Exercise-4/blob/master/README.md> dosyası içindeki linklerden [Problem 1 - Simple temperature calculator (6 points)](https://github.com/geo-python-2019/Exercise-4/blob/master/Exercise-4-problem-1.ipynb) linkine tıklanarak Exercise-4-problem-1.ipynb proje dosyası açılır:
2. **Raw** üzerine gelinip , **sağ tuş + Farklı kaydet** ile dosya yerel repo içine indirilir:

Yukarıdaki 1 ve 2 adımları uygulanarak diğer iki proje dosyası da yerel reopya indirilir:



**Jupyter Lab** ortamında, **Changes** panosu altındaki **proje dosyaları seçilir**, **Summary** kutusu içine initial commit yazılır ve **COMMIT** düğmesine tıklanarak, değişiklikler, yerel repodaki **stages ortamına** aktarılır:

Yereldeki değişiklikleri onaylanmış dosyalar uzak repoya (github) gönderilir (Git/Push to Remote):



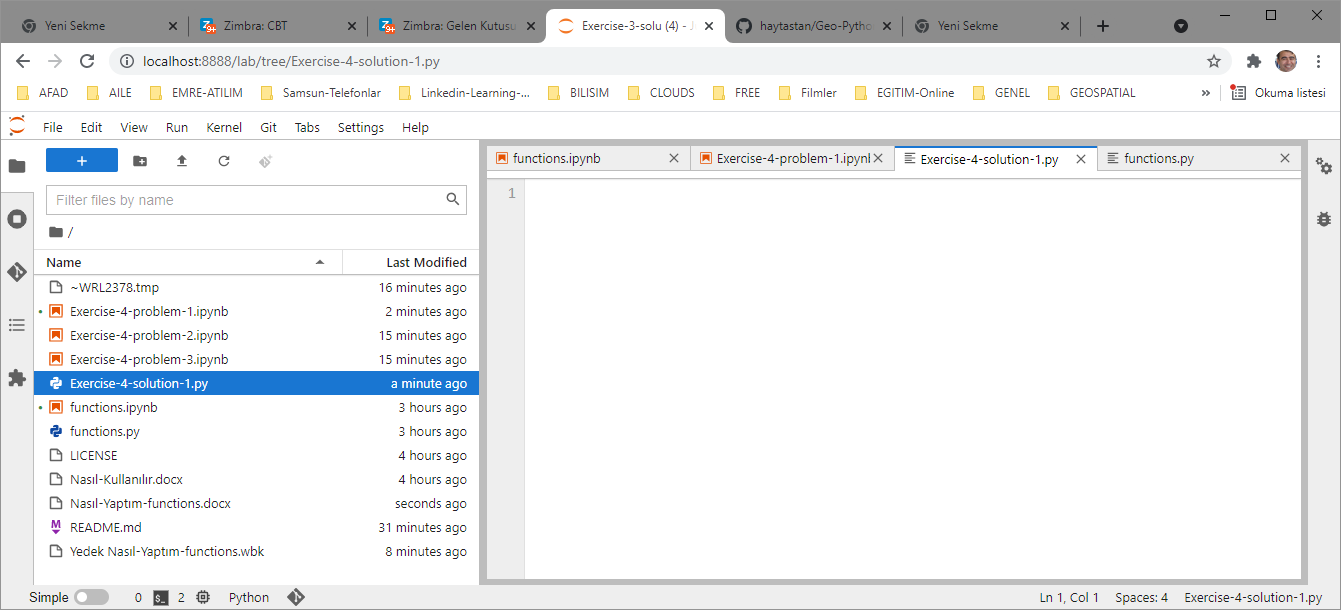
**Exercise Çözümleri:**

QGIS Shell ortamında:

Cd G:\python-uygulamalar\1\_geopython-autogis\1-geo-python-2020\ornekler\Geo-Python-2020-exercise-4-solutions

G:

Jupyter Lab

 Yerel **Jupyter Lab** ortamında, sol panodaki klasör ikonuna tıkla ve **Exercise-4-solution-1.py** isimli bir dosya oluştur:

**Exercise-4-solution-1.py** dosyası içine aşağıdaki çözüm satırlarını yaz:

# Dr. Hayati TAŞTAN

# 30.08.2021

# xxxxxxxxx

# =================

#Introducing the for loop

european\_cities **=** **[**'Amsterdam'**,** 'Brussels'**,** 'Lisbon'**,** 'Reykjavik'**]**

**for** city **in** european\_cities**:**

**print(**city**)**

**print(**'--------------------'**)**

#for loops and the range() function

**for** i **in** range**(2,9,3):**

**print(**i**)**

**print(**'--------------------'**)**

# Looping over the length of lists using index values

numbers **=** **[5,** **6,** **7]**

**for** i **in** range**(**len**(**numbers**)):**

**print(**'Value of i:'**,** i**)**

**print(**'Value of numbers[i] before addition:'**,** numbers**[**i**])**

numbers**[**i**]** **=** numbers**[**i**]** **+** i

**print(**'Value of numbers[i] after addition:'**,** numbers**[**i**])**

**print(**''**)**

**print(**numbers**)**

**print(**'--------------------'**)**

cities **=** **[**'Helsinki'**,** 'Stockholm'**,** 'Oslo'**,** 'Reykjavik'**,** 'Copenhagen'**]**

countries **=** **[**'Finland'**,** 'Sweden'**,** 'Norway'**,** 'Iceland'**,** 'Denmark'**]**

**for** i **in** range**(**len**(**cities**)):**

**print(**cities**[**i**],** 'is the capital of'**,** countries**[**i**])**

**print(**'--------------------'**)**

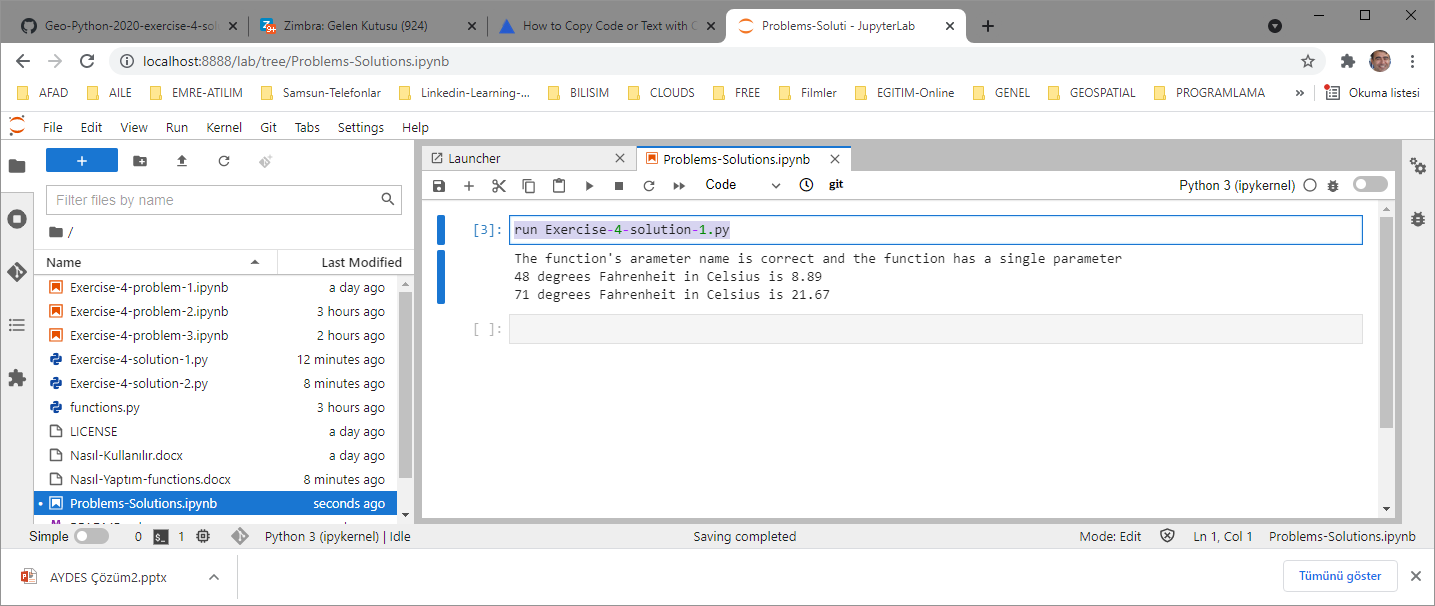
odd\_numbers **=** **[1,** **3,** **5]**

even\_numbers **=** **[2,** **4,** **6]**

**for** i **in** range**(**len**(**odd\_numbers**)):**

**print(**odd\_numbers**[**i**]** **+** even\_numbers**[**i**])**

**problems-solutions.ipnyb** dosya oluştur (Python lunch ile), varsa dosya içini temizle (hepsini seç ve makas ile sil), boş bir kutu içine yukarıdaki dosya içeriğini yaz ve SHIFT+ENTER; veya kutu içine sadece run Exercise-4-solution-1.py yaz:



**Exercise-4-solution-2.py** dosyası içine aşağıdaki çözüm satırlarını yaz:

# Hayati TAŞTAN

# 02.09.2021

# ===================

# SUBJECT: Temperature classifier

# PROBLEM: https://github.com/haytastan/Geo-Python-2020-exercise-4-solutions/blob/main/Exercise-4-problem-2.ipynb

# SOLUTION:

# P.A.: pip install nose.tools

**from** nose**.**tools **import** ok\_**,** assert\_equal

**import** inspect

**def** temp\_classifier**(**temp\_celsius**):**

**if** **(**temp\_celsius **<** **-2):**

category **=** **0**

**elif** **(**temp\_celsius **<=** **-2** **or** temp\_celsius **<** **2):**

category **=** **1**

**elif** **(**temp\_celsius **<=** **2** **or** temp\_celsius **<** **15):**

category **=** **2**

**else:**

category **=** **3**

**return** **(**category**)**

# Check that the function has a single parameter and the parameter name is correct

params **=** list**(**inspect**.**signature**(**temp\_classifier**).**parameters**.**keys**())**

**if** ok\_**(**params**[0]==**'temp\_celsius'**)** **and** ok\_**(**len**(**params**)==1):**

**print(**'The function\'s parameter name is incorrect and the function has not a single parameter'**)**

**else:**

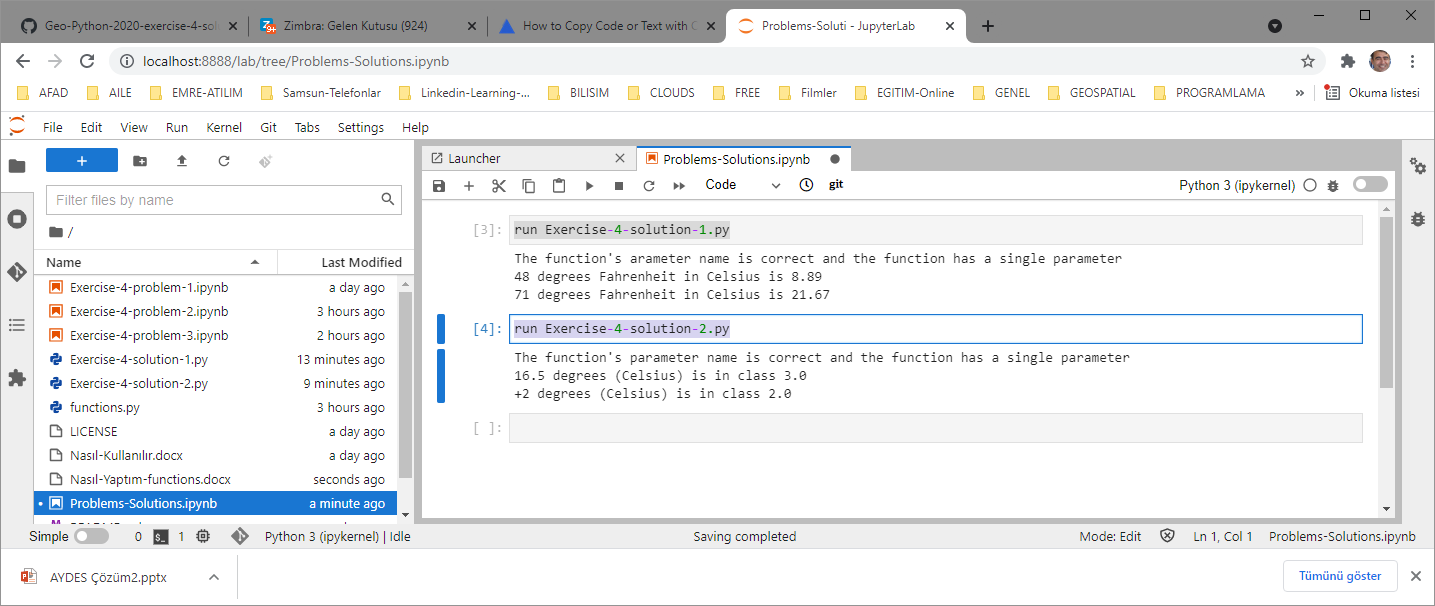
**print(**'The function\'s parameter name is correct and the function has a single parameter'**)**

# 1. What is the class value for 16.5 degrees (Celsius)?

**print(**'16.5 degrees (Celsius) is in class {0:.1f}'**.**format**(**temp\_classifier**(16.5)))**

# 2. What is the class value for +2 degrees (Celsius)?

**print(**'+2 degrees (Celsius) is in class {0:.1f}'**.**format**(**temp\_classifier**(2)))**

**problems-solutions.ipnyb** boş bir kutu içine yukarıdaki dosya içeriğini yaz ve SHIFT+ENTER; veya kutu içine sadece run Exercise-4-solution-2.py yaz:

**Exercise-4-solution-3.py** dosyası içine aşağıdaki çözüm satırlarını yaz:

# Hayati TAŞTAN

# 02.09.2021

# ===================

# SUBJECT: Applying the classifier

# PROBLEM: https://github.com/haytastan/Geo-Python-2020-exercise-4-solutions/blob/main/Exercise-4-problem-3.ipynb

# SOLUTION:

# P.A.: pip install nose.tools

**from** nose**.**tools **import** ok\_**,** assert\_equal

**import** inspect

**from** temp\_functions **import** fahr\_to\_celsius**,** temp\_classifier

temp\_classes **=** **[]**

# List of half-hourly temperature values (in degrees Fahrenheit) for one week

temp\_data **=** **[19,** **21,** **21,** **21,** **23,** **23,** **23,** **21,** **19,** **21,** **19,** **21,** **23,** **27,** **27,** **28,** **30,** **30,** **32,** **32,** **32,** **32,**

**34,** **34,** **34,** **36,** **36,** **36,** **36,** **36,** **36,** **34,** **34,** **34,** **34,** **34,** **34,** **32,** **30,** **30,** **30,** **28,** **28,** **27,**

**27,** **27,** **23,** **23,** **21,** **21,** **21,** **19,** **19,** **19,** **18,** **18,** **21,** **27,** **28,** **30,** **32,** **34,** **36,** **37,** **37,** **37,**

**39,** **39,** **39,** **39,** **39,** **39,** **41,** **41,** **41,** **41,** **41,** **39,** **39,** **37,** **37,** **36,** **36,** **34,** **34,** **32,** **30,** **30,**

**28,** **27,** **27,** **25,** **23,** **23,** **21,** **21,** **19,** **19,** **19,** **18,** **18,** **18,** **21,** **25,** **27,** **28,** **34,** **34,** **41,** **37,**

**37,** **39,** **39,** **39,** **39,** **41,** **41,** **39,** **39,** **39,** **39,** **39,** **41,** **39,** **39,** **39,** **37,** **36,** **34,** **32,** **28,** **28,**

**27,** **25,** **25,** **25,** **23,** **23,** **23,** **23,** **21,** **21,** **21,** **21,** **19,** **21,** **19,** **21,** **21,** **19,** **21,** **27,** **28,** **32,**

**36,** **36,** **37,** **39,** **39,** **39,** **39,** **39,** **41,** **41,** **41,** **41,** **41,** **41,** **41,** **41,** **41,** **39,** **37,** **36,** **36,** **34,**

**32,** **30,** **28,** **28,** **27,** **27,** **25,** **25,** **23,** **23,** **23,** **21,** **21,** **21,** **19,** **19,** **19,** **19,** **19,** **19,** **21,** **23,**

**23,** **23,** **25,** **27,** **30,** **36,** **37,** **37,** **39,** **39,** **41,** **41,** **41,** **39,** **39,** **41,** **43,** **43,** **43,** **43,** **43,** **43,**

**43,** **43,** **43,** **39,** **37,** **37,** **37,** **36,** **36,** **36,** **36,** **34,** **32,** **32,** **32,** **32,** **30,** **30,** **28,** **28,** **28,** **27,**

**27,** **27,** **27,** **25,** **27,** **27,** **27,** **28,** **28,** **28,** **30,** **32,** **32,** **32,** **34,** **34,** **36,** **36,** **36,** **37,** **37,** **37,**

**37,** **37,** **37,** **37,** **37,** **37,** **36,** **34,** **30,** **30,** **27,** **27,** **25,** **25,** **23,** **21,** **21,** **21,** **21,** **19,** **19,** **19,**

**19,** **19,** **18,** **18,** **18,** **18,** **18,** **19,** **23,** **27,** **30,** **32,** **32,** **32,** **32,** **32,** **32,** **34,** **34,** **34,** **34,** **34,**

**36,** **36,** **36,** **36,** **36,** **32,** **32,** **32,** **32,** **32,** **32,** **32,** **32,** **30,** **30,** **30,** **30,** **30,** **30,** **30,** **30,** **30,**

**30,** **30,** **30,** **30,** **28,** **28]**

**for** i **in** range**(**len**(**temp\_data**)):**

temp\_celcius **=** fahr\_to\_celsius**(**temp\_data**[**i**])**

temp\_class **=** temp\_classifier**(**temp\_celcius**)**

temp\_classes**.**append**(**temp\_class**)**

zeros **=** temp\_classes**.**count**(0)**

ones **=** temp\_classes**.**count**(1)**

twos **=** temp\_classes**.**count**(2)**

threes **=** temp\_classes**.**count**(3)**

# 1. How many 0 values exist in temp\_classes -list?

**print(**"temp\_classes -list contains {count: .0f} 0 values"**.**format**(**count**=**zeros**))**

# 2. How many 1 values exist in temp\_classes -list?

**print(**"temp\_classes -list contains {count: .0f} 1 values"**.**format**(**count**=**ones**))**

# 3. How many 2 values exist in temp\_classes -list?

**print(**"temp\_classes -list contains {count: .0f} 2 values"**.**format**(**count**=**twos**))**

# 4. How many 3 values exist in temp\_classes -list?

**print(**"temp\_classes -list contains {count: .0f} 3 values"**.**format**(**count**=**threes**))**

# Check that the functions have a single parameter

t\_params **=** list**(**inspect**.**signature**(**temp\_classifier**).**parameters**.**keys**())**

f\_params **=** list**(**inspect**.**signature**(**fahr\_to\_celsius**).**parameters**.**keys**())**

**if** ok\_**(**len**(**t\_params**)==1)** **and** ok\_**(**len**(**f\_params**)==1):**

**print(**'The functions have not a single parameter'**)**

**else:**

**print(**'The functions have a single parameter'**)**

# Check that functions are in the namespace (in the current directory)

**if** ok\_**(**inspect**.**isfunction**(**fahr\_to\_celsius**))** **and** ok\_**(**inspect**.**isfunction**(**temp\_classifier**)):**

**print(**'The functions are not in the namespace'**)**

**else:**

**print(**'The functions are in the namespace'**)**

# Check that variable has been created

**if** ok\_**(**'temp\_celcius' **in** locals**())** **and** ok\_**(**'temp\_class' **in** locals**())** **and** ok\_**(**'temp\_classes' **in** locals**()):**

**print(**'The variables have not been been created'**)**

**else:**

**print(**'The variables have been been created'**)**

# Check that temp\_classes is a list

**if** ok\_**(**type**(**temp\_classes**)** **==** list**):**

**print(**'The variables have not been been created'**)**

**else:**

**print(**'The variables have been been created'**)**

# Check that the variable "zeros" exists and is correct

**if** ok\_**(**'zeros' **in** locals**()):**

**print(**'the variable "zeros" doesn\'t exists or correct'**)**

**else:**

**print(**'the variable "zeros" exists and is correct'**)**

# Check that the variable "ones" exists and is correct

**if** ok\_**(**'ones' **in** locals**()):**

**print(**'the variable "ones" doesn\'t exists or correct'**)**

**else:**

**print(**'the variable "ones" exists and is correct'**)**

# Check that the variable "twos" exists and is correct

**if** ok\_**(**'twos' **in** locals**()):**

**print(**'the variable "twos" doesn\'t exists or correct'**)**

**else:**

**print(**'the variable "twos" exists and is correct'**)**

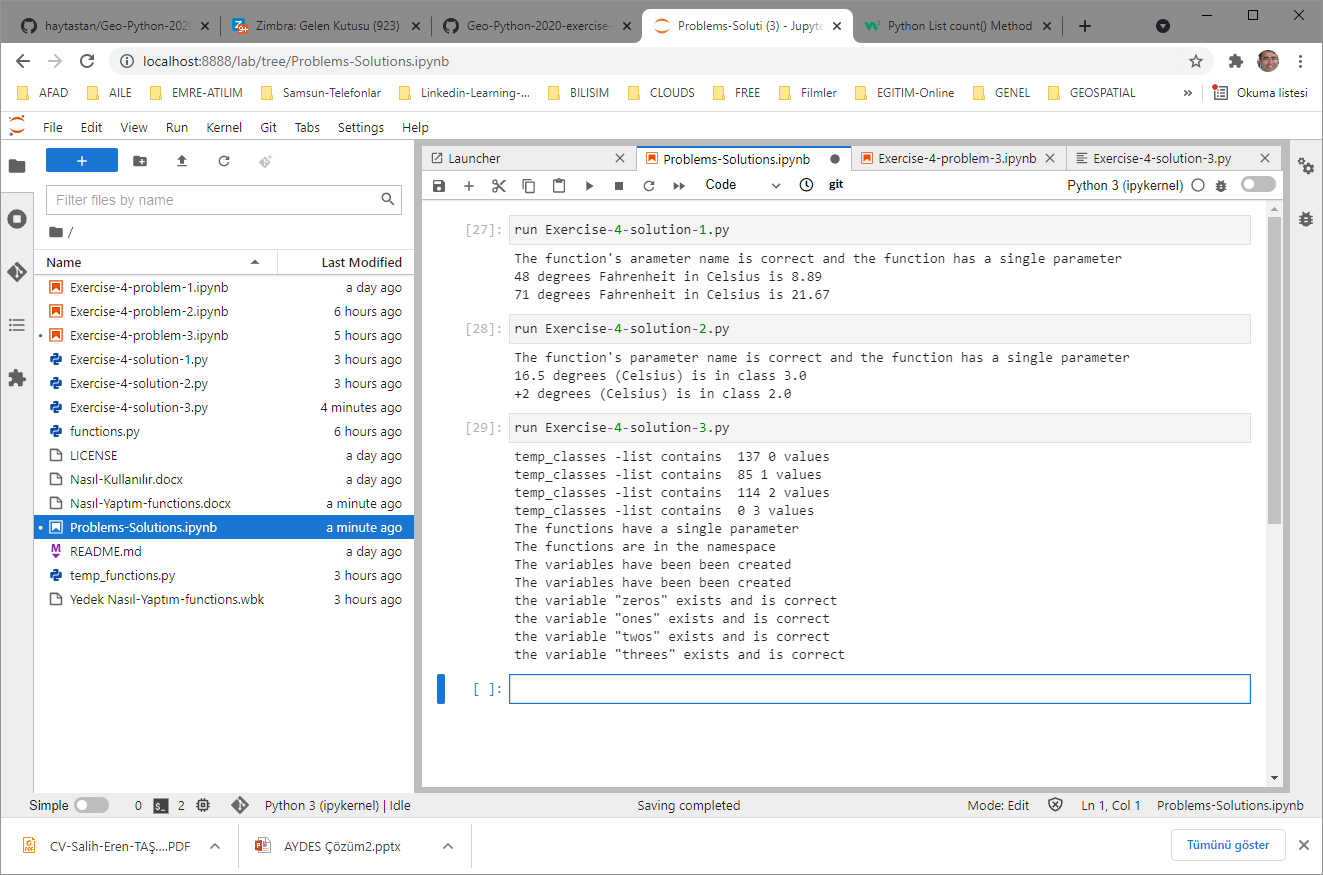
# Check that the variable "threes" exists and is correct

**if** ok\_**(**'threes' **in** locals**()):**

**print(**'the variable "threes" doesn\'t exists or correct'**)**

**else:**

**print(**'the variable "threes" exists and is correct'**)**

**problems-solutions.ipnyb** boş bir kutu içine yukarıdaki dosya içeriğini yaz ve SHIFT+ENTER; veya kutu içine sadece run Exercise-4-solution-3.py yaz.