Instructions for the Instructors

Contents

The instructions in this document are provided to work on the local system only, and not possible to work on Google Colab as it doesn't have a cell type of $Raw\ NBConvert$.

I. Installation of otter grader

Step 1: Install otter-grader library

The otter library has to install once in the local system using the below command.

$pip\ install\ otter - grader$

The instructor might require installing MiKTeX if it required.

https://miktex.org/download

Step 2: Import necessary libraries

Once the otter-grader library is installed, run the below command to initiate auto-grading:

$import\ os$

$import\ otter$

$grader\ =\ otter.Notebook()$

II. Set up the file format

The instructor has to follow the standard file format to set up the file for auto-grading.

Step 1: Begin Question

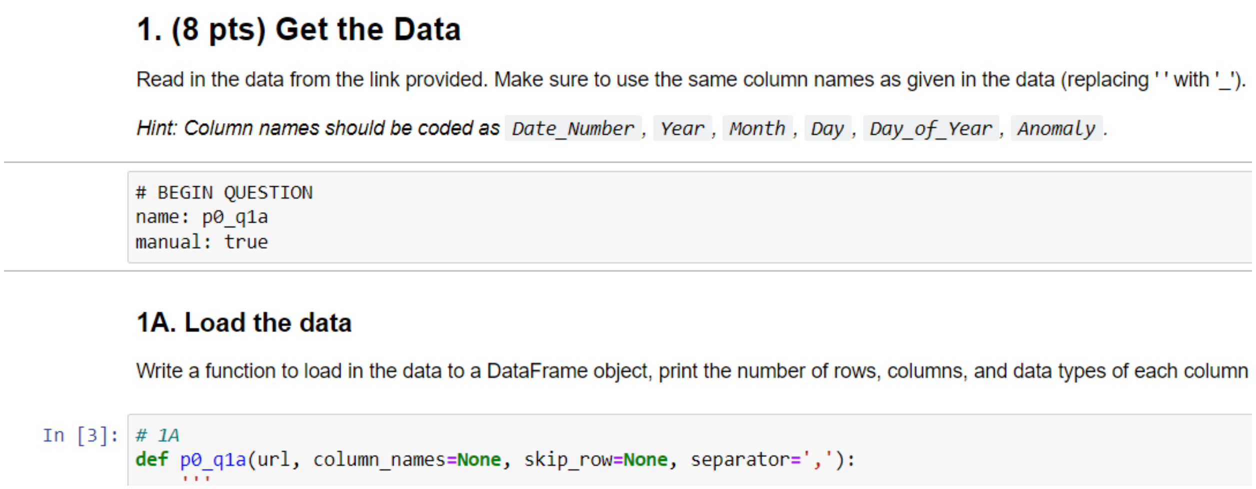
For every question, the below commands should copy into the $RawNBConvert$ cell type:

$\#\ BEGIN\ QUESTION$

$name: p0\_q1a$

$manual: true$

The $name$ of the question should match the name provided in the respective test case of that question. The $manual$ is optional, it provides an option to grade manually along with the auto-grading.


1. (8 pts) Get the Data

Read in the data from the link provided. Make sure to use the same column names as given in the data (replacing ' ' with '_').

Hint: Column names should be coded as `Date_Number`, `Year`, `Month`, `Day`, `Day_of_Year`, `Anomaly`.

```
# BEGIN QUESTION
name: p0_q1a
manual: true
```

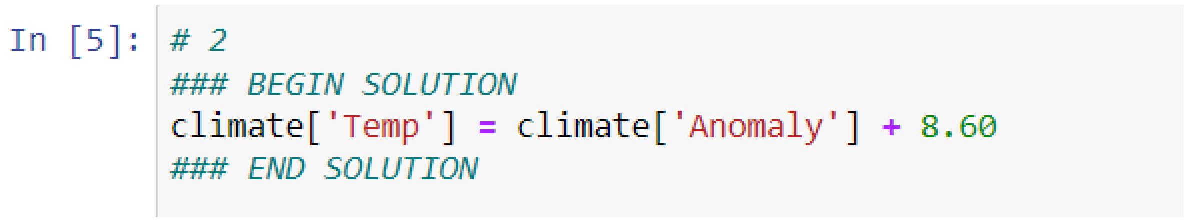
1A. Load the data

Write a function to load in the data to a DataFrame object, print the number of rows, columns, and data types of each column

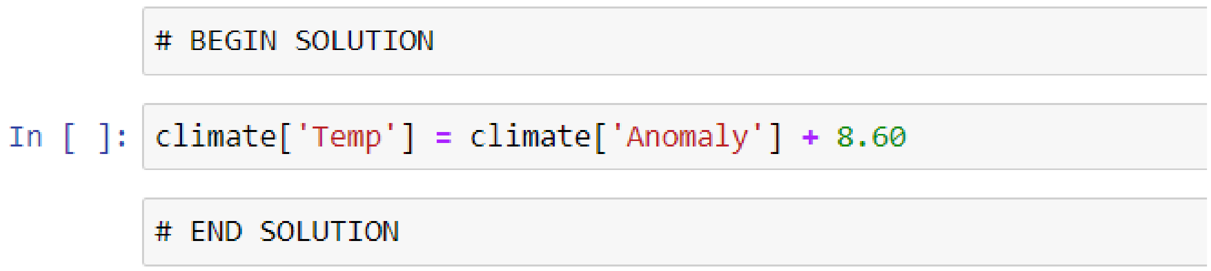
```
In [3]: # 1A
def p0_q1a(url, column_names=None, skip_row=None, separator=','):
```

Step 2: Begin and End Solution

Now the solution (code snippets) is provided in code cell type between the *BEGIN SOLUTION* and *END SOLUTION*. The *BEGIN SOLUTION* and *END SOLUTION* can be either provided in the *RawNBConvert* cell type or included in the code snippet.


```
In [5]: # 2
### BEGIN SOLUTION
climate['Temp'] = climate['Anomaly'] + 8.60
### END SOLUTION
```

Or

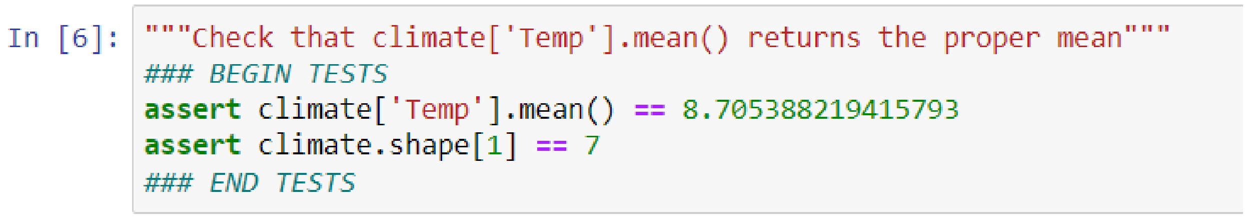

```
# BEGIN SOLUTION
```

```
In [ ]: climate['Temp'] = climate['Anomaly'] + 8.60
```

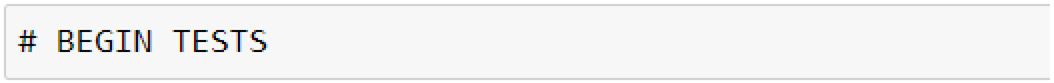
```
# END SOLUTION
```

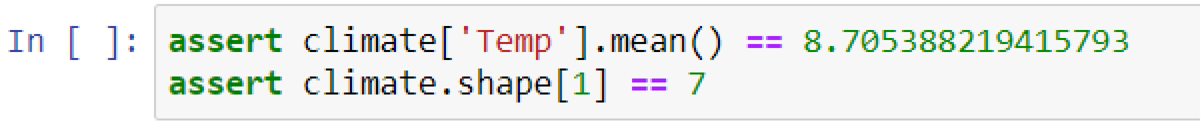
Step 3: Begin and End Tests

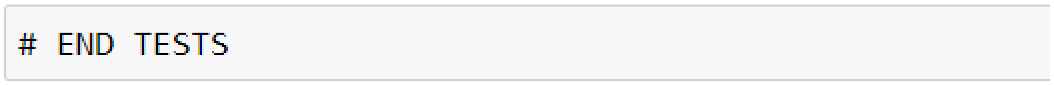
Now write the test cases using the assert function of the code snippet cell. The test cases should write in between *BEGIN TESTS* and *END TESTS*. The *BEGIN TESTS* and *END TESTS* can be either provided in the *RawNBConvert* cell type or included in the code snippet.


```
In [6]: """Check that climate['Temp'].mean() returns the proper mean"""
        ### BEGIN TESTS
        assert climate['Temp'].mean() == 8.705388219415793
        assert climate.shape[1] == 7
        ### END TESTS
```

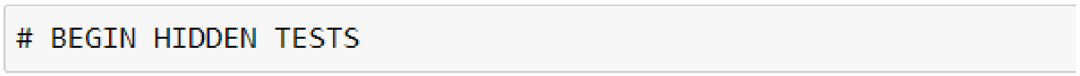
Or

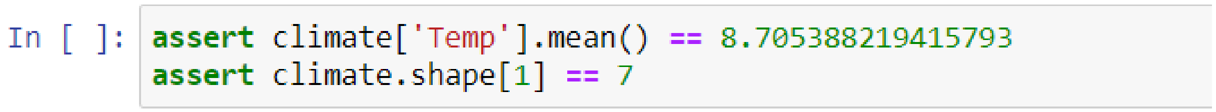

```
# BEGIN TESTS
```

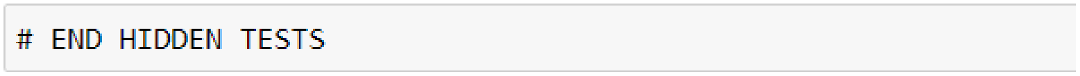

```
In [ ]: assert climate['Temp'].mean() == 8.705388219415793
        assert climate.shape[1] == 7
```


```
# END TESTS
```

To hide the test cases from the students, add the hidden keyword.


```
# BEGIN HIDDEN TESTS
```

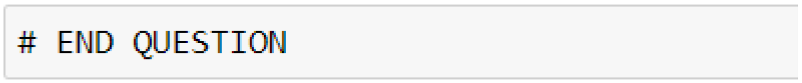

```
In [ ]: assert climate['Temp'].mean() == 8.705388219415793
        assert climate.shape[1] == 7
```


```
# END HIDDEN TESTS
```

Step 4: End Question

Similar to step 1, for every question, the below commands should copy into the *RawNBConvert* cell type:

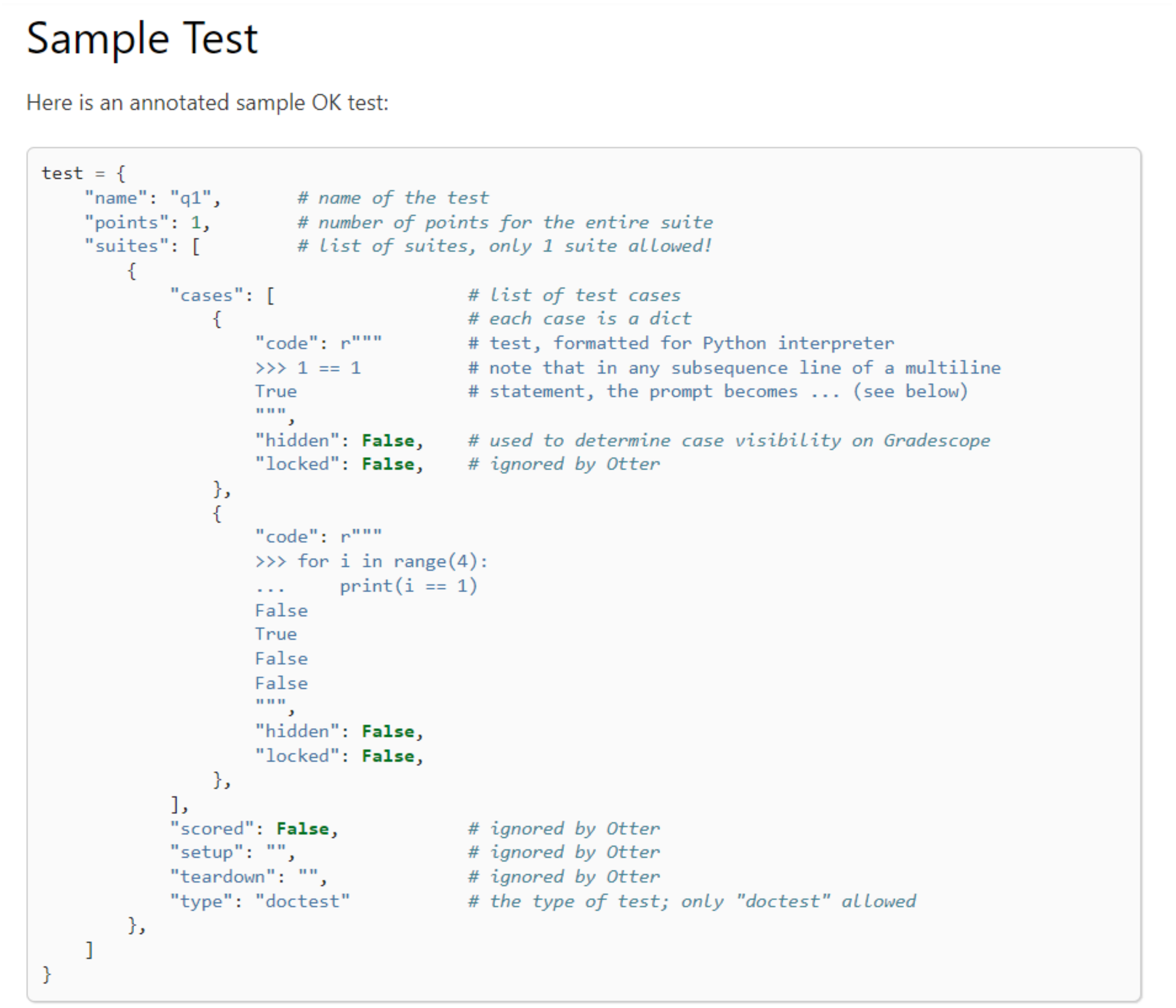
# END Question


```
# END QUESTION
```

Step 5: Repeat steps 1 to 4 to all the questions. Avoid step 3 for the manual questions.

III. Creation of test cases

The file format for the test cases depends on the programming language. The Python follows the OK test format.



Sample Test

Here is an annotated sample OK test:

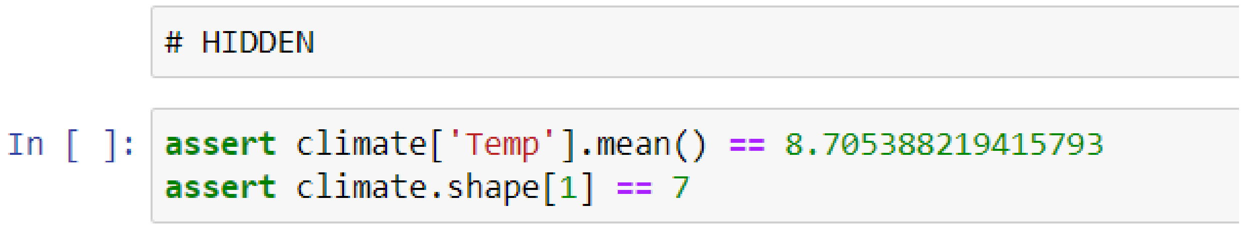
```
test = {
    "name": "q1",       # name of the test
    "points": 1,        # number of points for the entire suite
    "suites": [         # list of suites, only 1 suite allowed!
        {
            "cases": [                  # list of test cases
                {                       # each case is a dict
                    "code": r"""        # test, formatted for Python interpreter
                    >>> 1 == 1          # note that in any subsequence line of a multiline
                    True                # statement, the prompt becomes ... (see below)
                    """,
                    "hidden": False,    # used to determine case visibility on Gradescope
                    "locked": False,    # ignored by Otter
                },
                {
                    "code": r"""
                    >>> for i in range(4):
                    ...     print(i == 1)
                    False
                    True
                    False
                    False
                    """,
                    "hidden": False,
                    "locked": False,
                },
            ],
            "scored": False,            # ignored by Otter
            "setup": "",                # ignored by Otter
            "teardown": "",             # ignored by Otter
            "type": "doctest"           # the type of test; only "doctest" allowed
        },
    ]
}
```

Source: https://otter-grader.readthedocs.io/en/latest/test_files/ok_format.html

All the test case files are in $.py$ format. The name of the test file should match the name of the question given in the $RawNBConvert$ cell type of the python notebook.

There are two ways to generate test case files:

1) Manually write the test cases in the above-mentioned format.
2) Use the # $HIDDEN$ keyword in the jupyter notebook, to generate automatically.

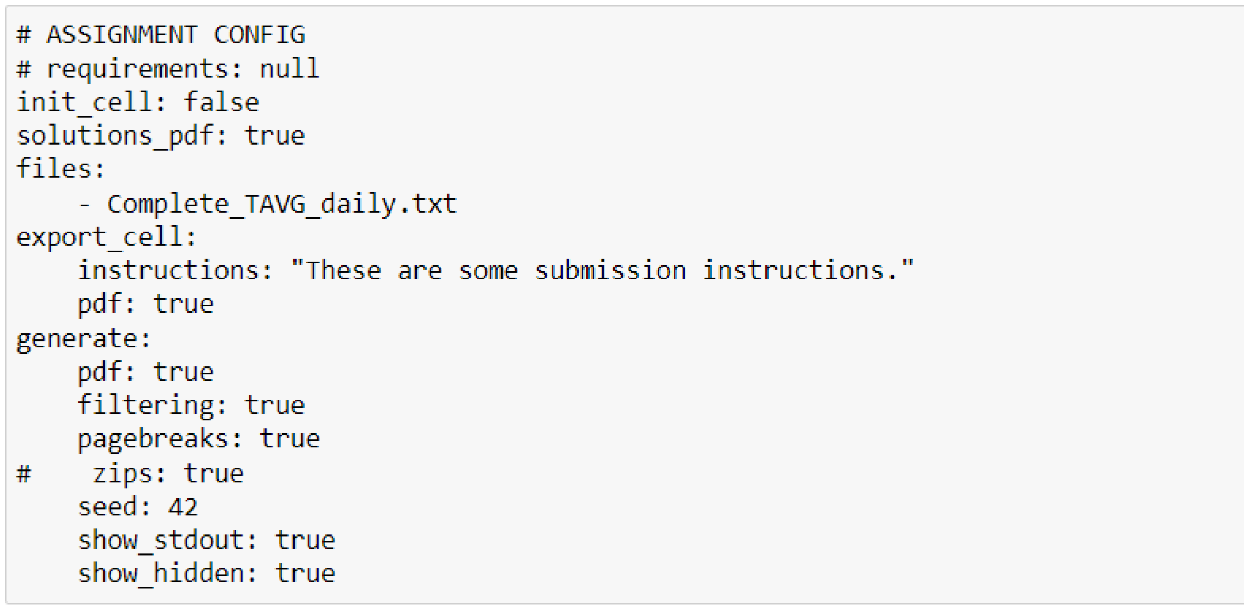


```
# HIDDEN
```

```
In [ ]: assert climate['Temp'].mean() == 8.705388219415793
        assert climate.shape[1] == 7
```

IV. Creation of Assignment

Step 1: The first cell of the notebook, places YAML-formatted configurations in a raw cell that begins with the comment # *ASSIGNMENT CONFIG*



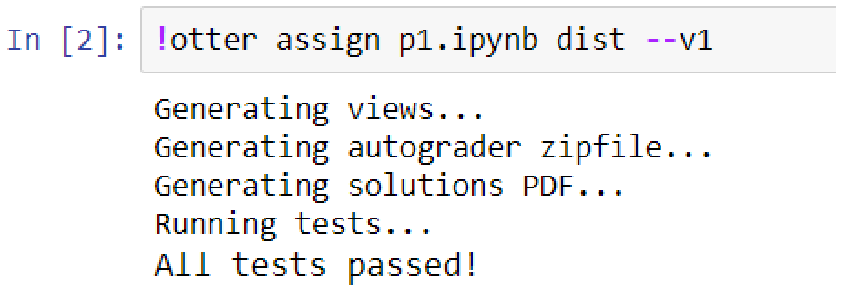
```
# ASSIGNMENT CONFIG
# requirements: null
init_cell: false
solutions_pdf: true
files:
    - Complete_TAVG_daily.txt
export_cell:
    instructions: "These are some submission instructions."
    pdf: true
generate:
    pdf: true
    filtering: true
    pagebreaks: true
#    zips: true
    seed: 42
    show_stdout: true
    show_hidden: true
```

Refer to the website for additional instructions which can be included in # *ASSIGNMENT CONFIG*

https://otter-grader.readthedocs.io/en/latest/otter_assign/v1/notebook_format.html

Step 2: Otter Assign

The Otter Assign command generates the solution file in pdf format and autograder zip file.



```
In [2]: !otter assign p1.ipynb dist --v1

        Generating views...
        Generating autograder zipfile...
        Generating solutions PDF...
        Running tests...
        All tests passed!
```

The Otter Assign command creates a dist directory with two other subdirectories, autograder and student. The autograder directory contains the Gradescope autograder, solutions PDF, and the notebook with solutions. The student directory contains just the sanitized student notebook.

```
tutorial/dist
├── autograder
│   ├── autograder.zip
│   ├── demo-sol.pdf
│   ├── demo.ipynb
│   ├── otter_config.json
│   └── requirements.txt
└── student
    └── demo.ipynb
```

Source: https://otter-grader.readthedocs.io/en/latest/tutorial.html

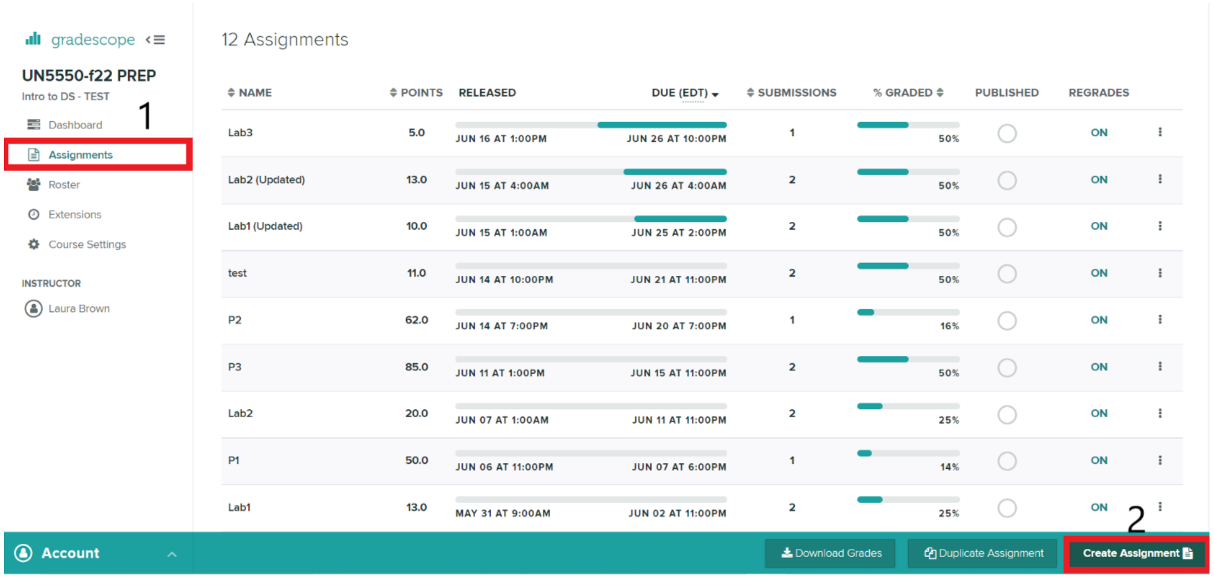
If the “tests” folder (having test cases) is not available in the zip file generated by the otter assign command, then copy the “tests” folder into the autograder zip file.

Executing the otter assign command from the terminal or command prompt included the “tests” folder automatically into the autograder zip file.

V. Assignment creation in Gradescope

Source: https://help.gradescope.com/article/ujutnle52h-instructor-assignment-programming

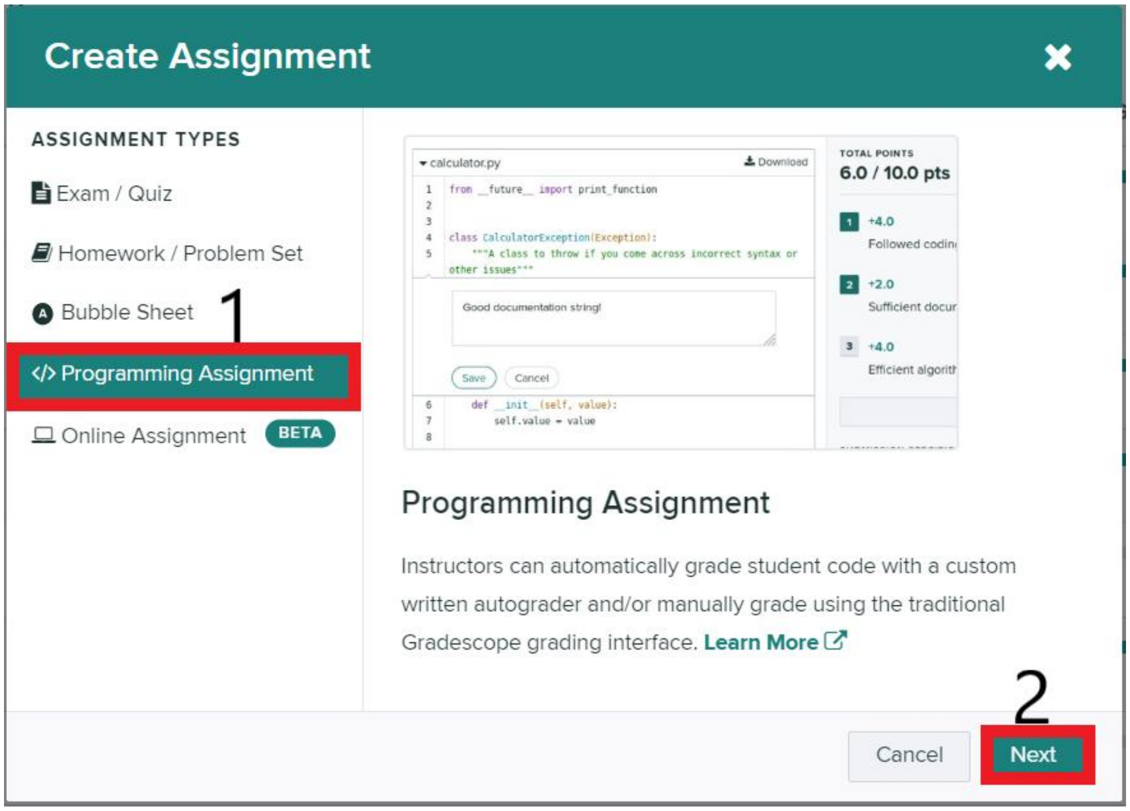
Step 1: Click on Assignments and choose to Create Assignment tab.



12 Assignments

NAME	POINTS	RELEASED	DUE (EDT)	SUBMISSIONS	% GRADED	PUBLISHED	REGRADES
Lab3	5.0	JUN 16 AT 1:00PM	JUN 26 AT 10:00PM	1	50%		ON
Lab2 (Updated)	13.0	JUN 15 AT 4:00AM	JUN 26 AT 4:00AM	2	50%		ON
Lab1 (Updated)	10.0	JUN 15 AT 1:00AM	JUN 25 AT 2:00PM	2	50%		ON
test	11.0	JUN 14 AT 10:00PM	JUN 21 AT 11:00PM	2	50%		ON
P2	62.0	JUN 14 AT 7:00PM	JUN 20 AT 7:00PM	1	16%		ON
P3	85.0	JUN 11 AT 1:00PM	JUN 15 AT 11:00PM	2	50%		ON
Lab2	20.0	JUN 07 AT 1:00AM	JUN 11 AT 11:00PM	2	25%		ON
P1	50.0	JUN 06 AT 11:00PM	JUN 07 AT 6:00PM	1	14%		ON
Lab1	13.0	MAY 31 AT 9:00AM	JUN 02 AT 11:00PM	2	25%		ON

Step 2: A pop-up window is displayed and select Programming Assignment, Click Next.



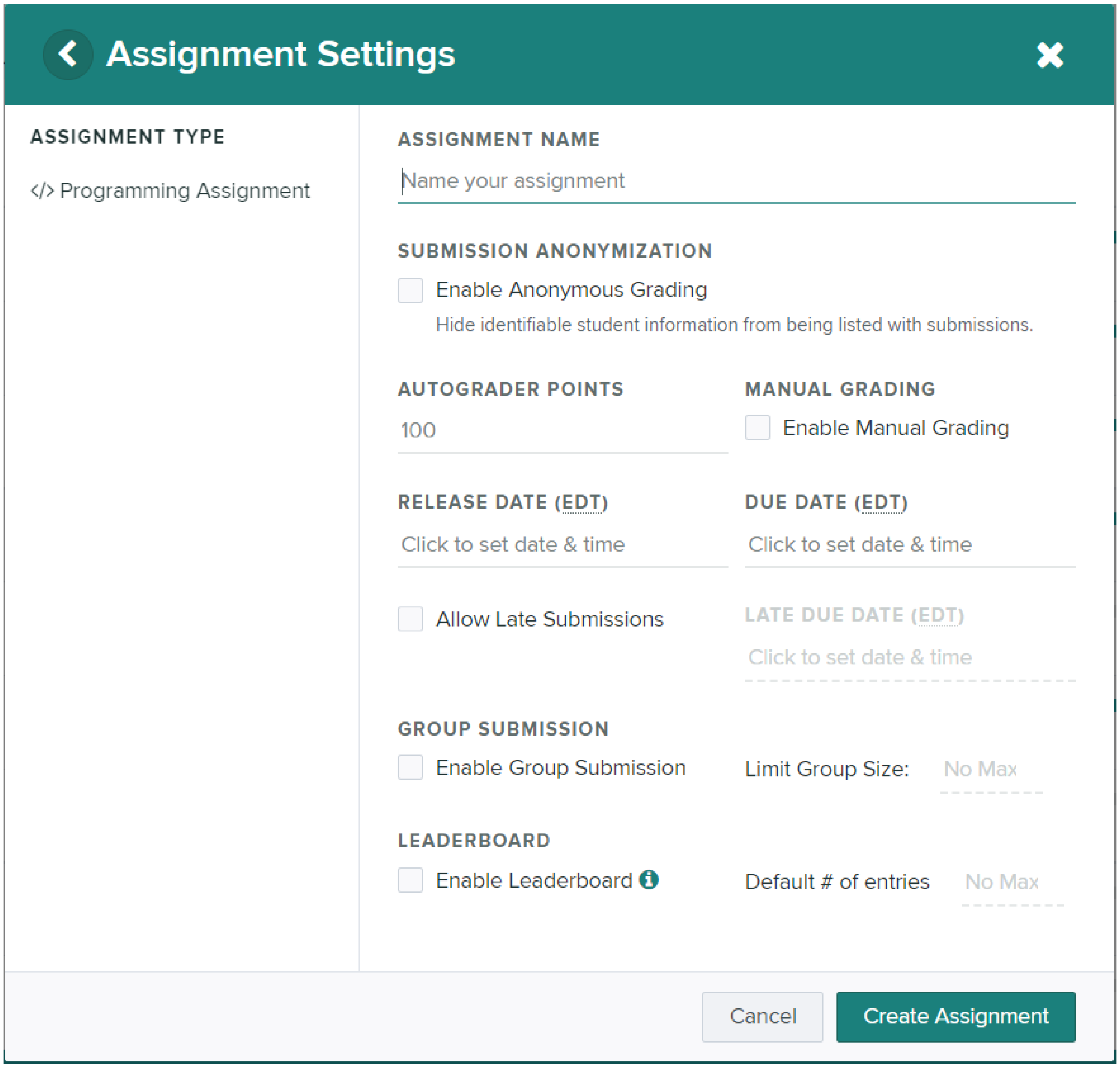
```
from __future__ import print_function


class CalculatorException(Exception):
    """A class to throw if you come across incorrect syntax or
other issues"""
    def __init__(self, value):
        self.value = value
```

Programming Assignment

Instructors can automatically grade student code with a custom written autograder and/or manually grade using the traditional Gradescope grading interface. Learn More

Step 3: A pop-up window is displayed and choose the respective Assignment Settings as required; Click Create Assignment.



Assignment Settings

ASSIGNMENT TYPE

</> Programming Assignment

ASSIGNMENT NAME

Name your assignment

SUBMISSION ANONYMIZATION

☐ Enable Anonymous Grading

Hide identifiable student information from being listed with submissions.

AUTOGRADER POINTS

100

MANUAL GRADING

☐ Enable Manual Grading

RELEASE DATE (EDT)

Click to set date & time

DUE DATE (EDT)

Click to set date & time

☐ Allow Late Submissions

LATE DUE DATE (EDT)

Click to set date & time

GROUP SUBMISSION

☐ Enable Group Submission

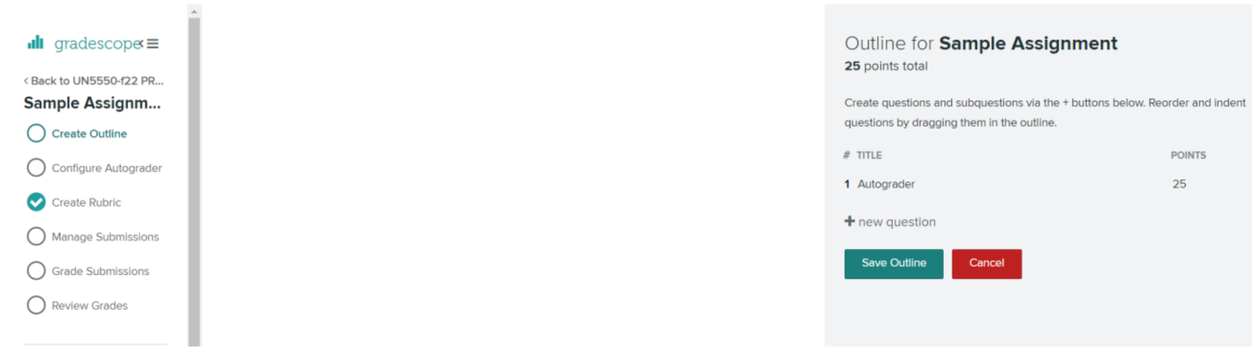
Limit Group Size: No Max

LEADERBOARD

☐ Enable Leaderboard

Default # of entries No Max

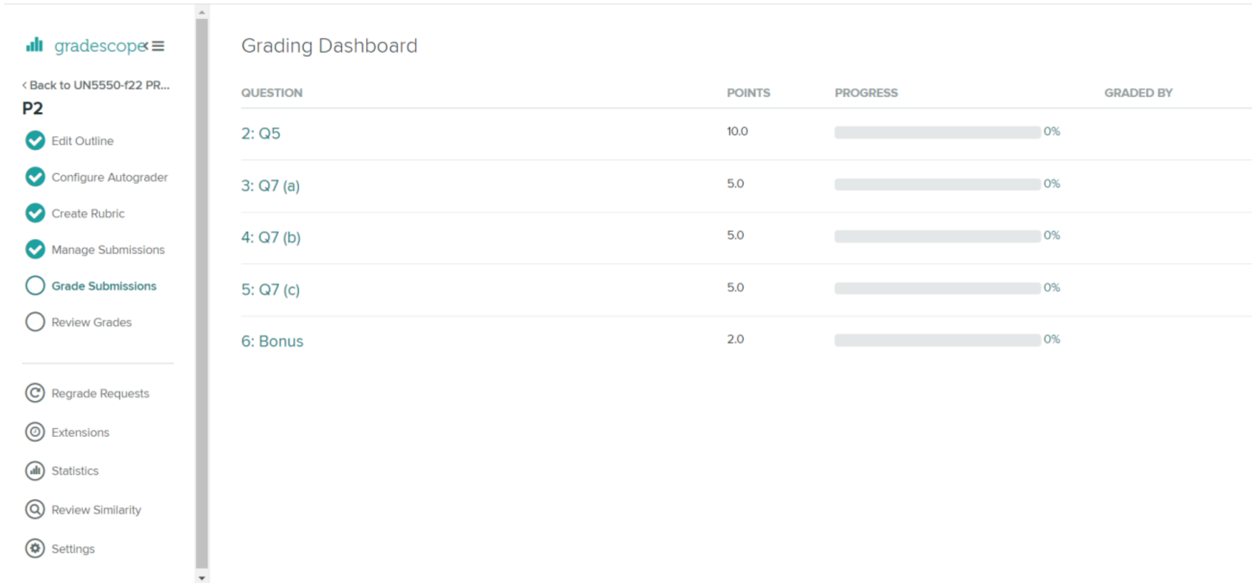
The instructor has to enable Manual Grading for manual grading of the questions and click on "Edit Outline" to add the manually graded questions.



Outline for **Sample Assignment**
25 points total

Create questions and subquestions via the + buttons below. Reorder and indent questions by dragging them in the outline.

#	TITLE	POINTS
1	Autograder	25

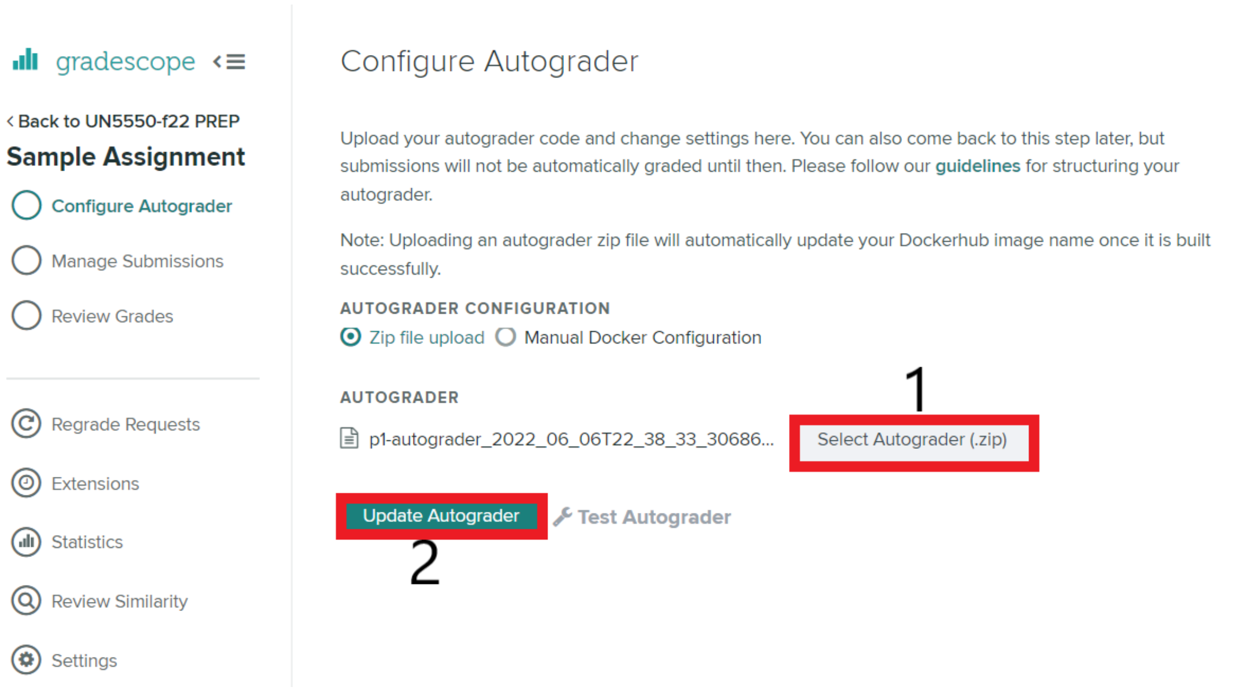


Grading Dashboard

QUESTION	POINTS	PROGRESS	GRADED BY
2: Q5	10.0	0%	
3: Q7 (a)	5.0	0%	
4: Q7 (b)	5.0	0%	
5: Q7 (c)	5.0	0%	
6: Bonus	2.0	0%	

Step 4: Configure Autograder

Select the autograder zip file from the dist directory generated using the otter assign command and click on “Update Autograder”.



Configure Autograder

Upload your autograder code and change settings here. You can also come back to this step later, but submissions will not be automatically graded until then. Please follow our guidelines for structuring your autograder.

Note: Uploading an autograder zip file will automatically update your Dockerhub image name once it is built successfully.

AUTOGRADER CONFIGURATION

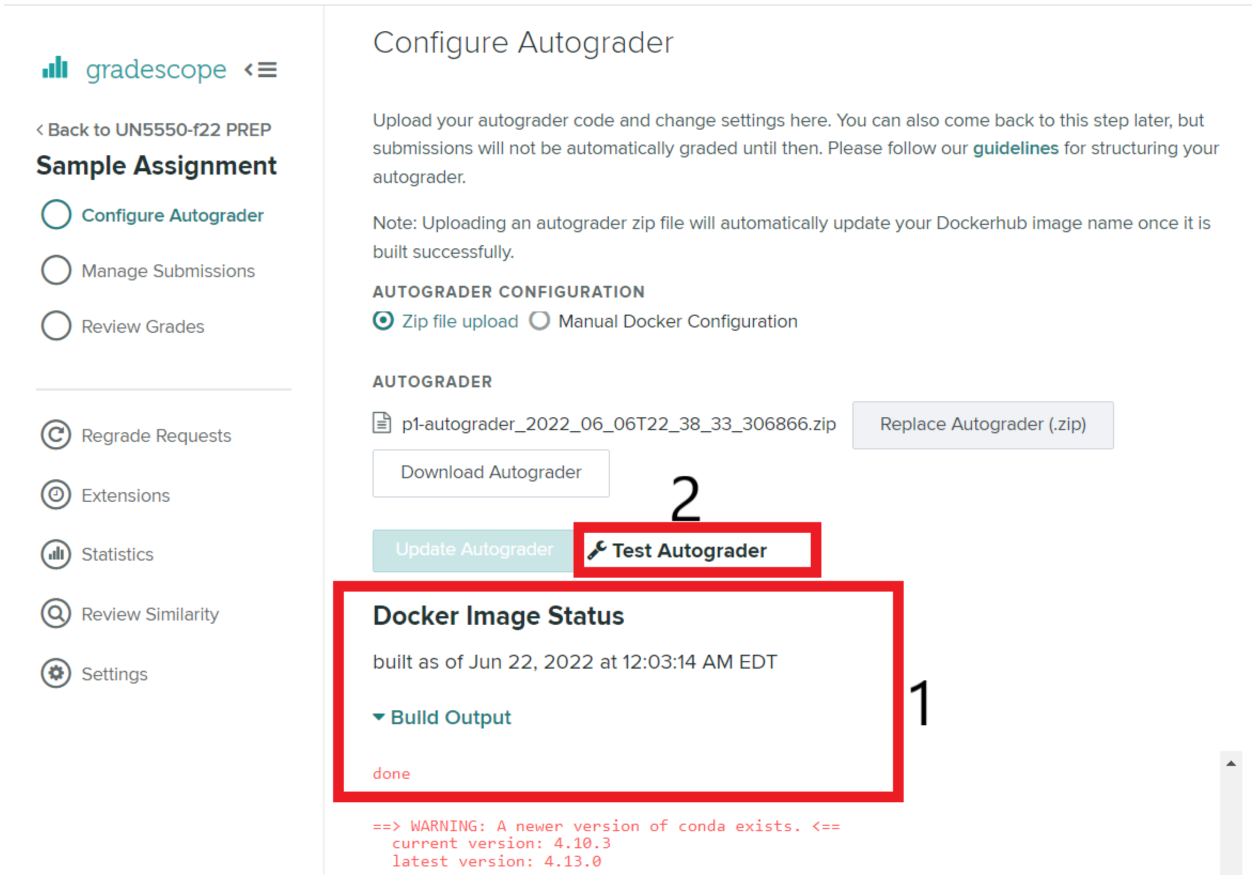
[✓ Zip file upload] Manual Docker Configuration

AUTOGRADER

p1-autograder_2022_06_06T22_38_33_30686...

Now the autograder setup runs for 5-7 minutes depending on the file size and requirements. The status of the setup is shown in the docker image status.

You can test the autograder with the sample submission by selecting Test Autograder once the autograder setup is done.



Configure Autograder

Upload your autograder code and change settings here. You can also come back to this step later, but submissions will not be automatically graded until then. Please follow our **guidelines** for structuring your autograder.

Note: Uploading an autograder zip file will automatically update your Dockerhub image name once it is built successfully.

AUTOGRADER CONFIGURATION
[✓ Zip file upload] Manual Docker Configuration

AUTOGRADER
p1-autograder_2022_06_06T22_38_33_306866.zip

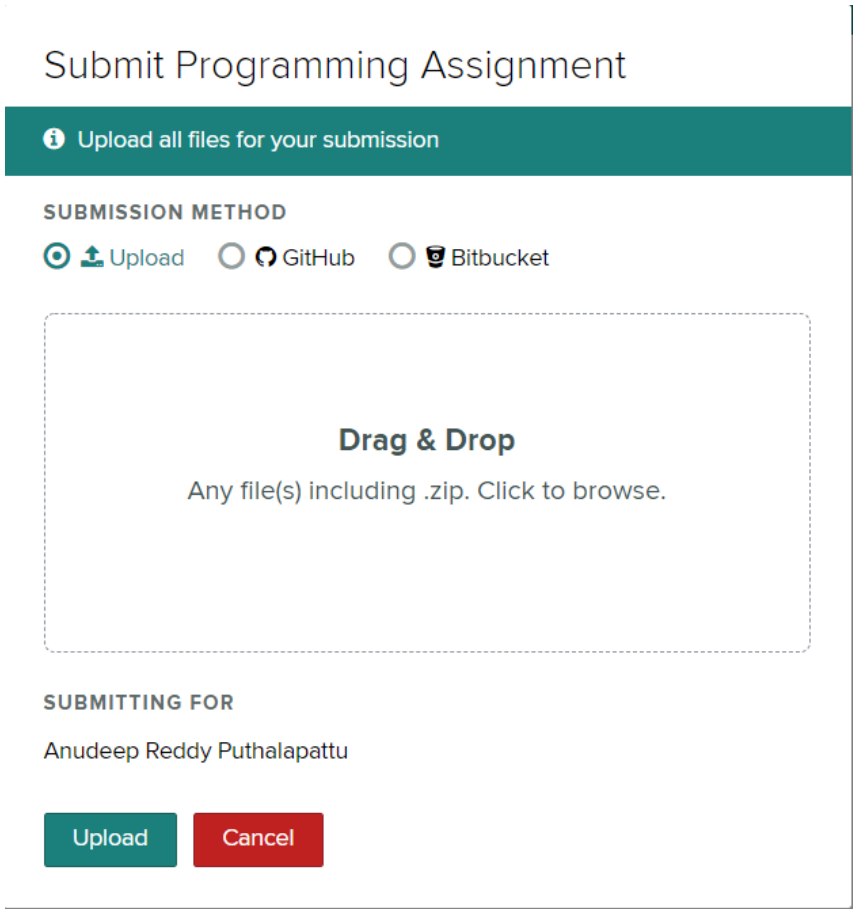
Docker Image Status
built as of Jun 22, 2022 at 12:03:14 AM EDT

Build Output

```
done

==> WARNING: A newer version of conda exists. <==
  current version: 4.10.3
  latest version: 4.13.0
```

Click on “upload” and select the file to upload.



Submit Programming Assignment

Upload all files for your submission

SUBMISSION METHOD

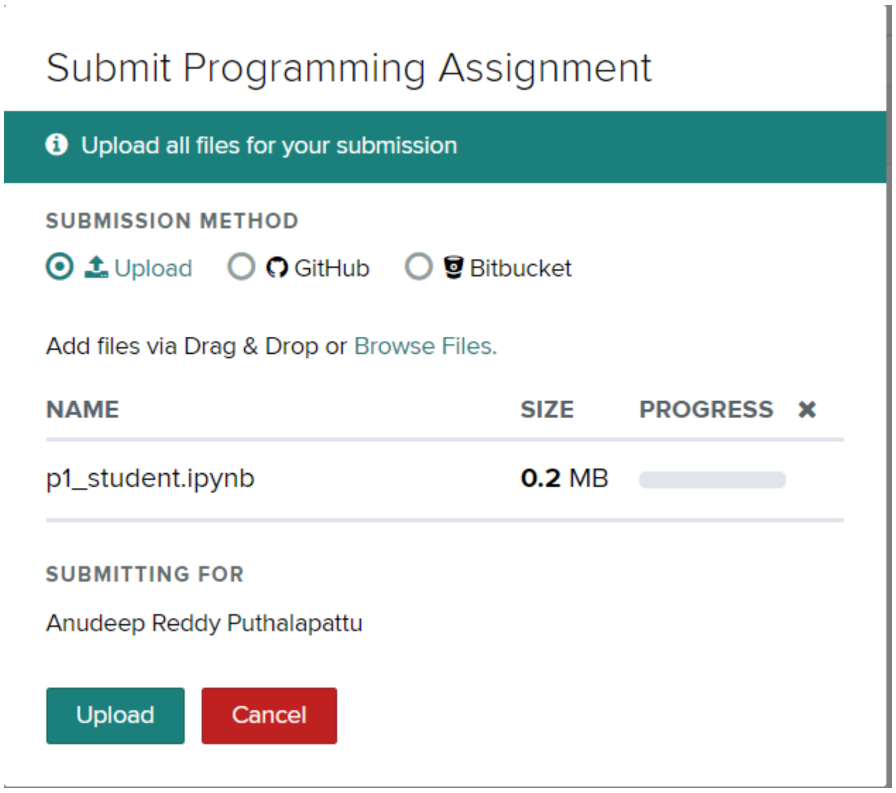
[✓ Upload] GitHub Bitbucket

Drag & Drop

Any file(s) including .zip. Click to browse.

SUBMITTING FOR

Anudeep Reddy Puthalapattu



Submit Programming Assignment

Upload all files for your submission

SUBMISSION METHOD

[✓ Upload] GitHub Bitbucket

Add files via Drag & Drop or Browse Files.

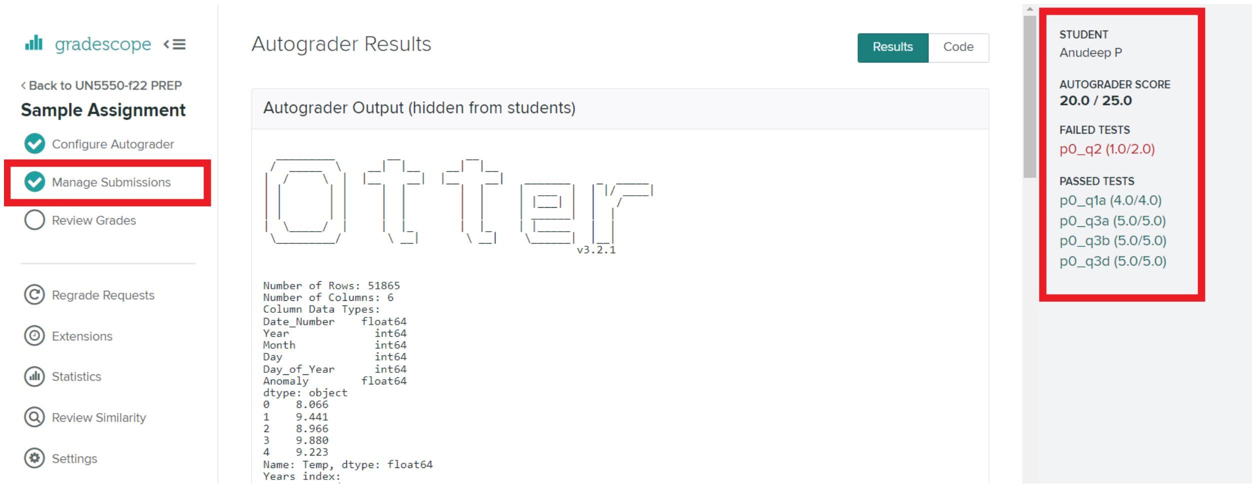
NAME	SIZE	PROGRESS ✖
p1_student.ipynb	**0.2** MB	

SUBMITTING FOR

Anudeep Reddy Puthalapattu

Wait for 2-3 minutes to auto-grade the submitted notebook and the results are displayed on the right side of the webpage.

The instructor can manage/view all submissions of students by selecting "Manage Submissions" on the left side of the webpage.



Autograder Results

Autograder Output (hidden from students)

```
                                        v3.2.1

Number of Rows: 51865
Number of Columns: 6
Column Data Types:
Date_Number    float64
Year             int64
Month            int64
Day              int64
Day_of_Year      int64
Anomaly        float64
dtype: object
0    8.066
1    9.441
2    8.966
3    9.880
4    9.223
Name: Temp, dtype: float64
Years index:
```

STUDENT
Anudeep P

AUTOGRADER SCORE
20.0 / 25.0

FAILED TESTS
p0_q2 (1.0/2.0)

PASSED TESTS
p0_q1a (4.0/4.0)
p0_q3a (5.0/5.0)
p0_q3b (5.0/5.0)
p0_q3d (5.0/5.0)