Name: Coco Jones
Date: February 26, 2023
Class: IT FDN 110 A
Module: 07
GitHub: https://github.com/cocopuffnagoya/ITFnd100-Mod07

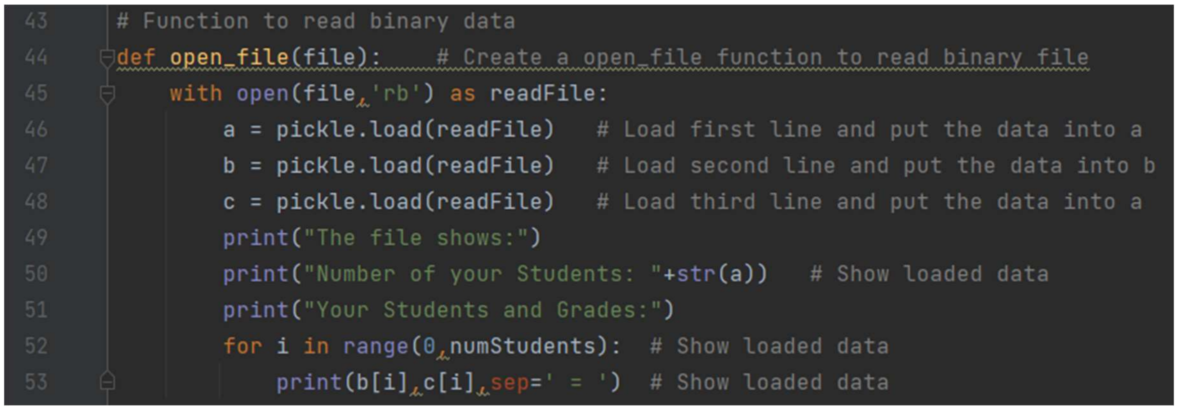
Pickling and Try-Except in Python

Introduction

In module 07, I researched how to handle errors using Try-Except and class and how to pickle data in Python. To complete this assignment, I watched a couple of YouTubes and read some articles on the web. I have pasted the URLs of the resources at the References section and written why I selected those resources for my assignment there. I decided to combine the two requirements (pickling and try-except) into one script.

What are the benefits of putting built-in Python command into functions?

Below is an example in which I put multiple Python commands into my open_file() function. This function has 8 commands. Once a function is defined, you can use this as many times as you want throughout the script. The 8 commands are in one block and this one function can be called multiple times later in the script. Benefits of putting built-in commands to functions are organizing our script and improving readability and reusability. (**Fig 1**)

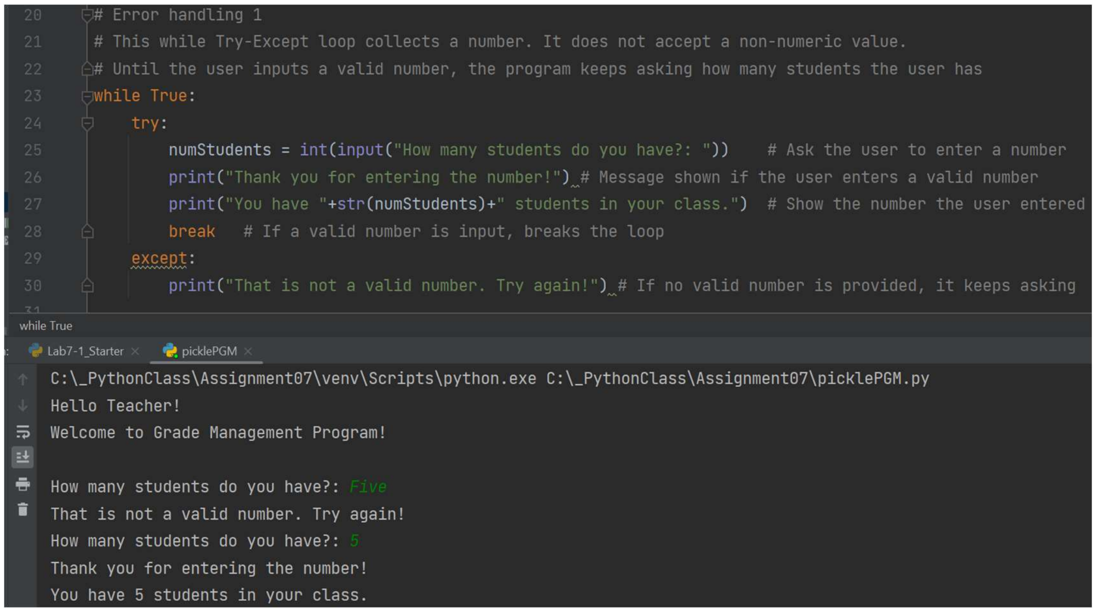


```
# Function to read binary data
def open_file(file):    # Create a open_file function to read binary file
    with open(file,'rb') as readFile:
        a = pickle.load(readFile)   # Load first line and put the data into a
        b = pickle.load(readFile)   # Load second line and put the data into b
        c = pickle.load(readFile)   # Load third line and put the data into a
        print("The file shows:")
        print("Number of your Students: "+str(a))   # Show loaded data
        print("Your Students and Grades:")
        for i in range(0,numStudents):  # Show loaded data
            print(b[i],c[i],sep=' = ')  # Show loaded data
```

Fig 1 - Multiple commands into one Function

What are the benefits of using structured error handling?

There are multiple benefits of structured error handling. One benefit is that you can use your custom error message instead of showing python built-in error message. Python's built-in error messages are not easy to understand for end users. Also, python's error message stops the whole program and does not allow the user to correct the mistake that she/he made. By using structured error handling, the user will know when there is an error, how he can correct and proceed. **Fig 2** below shows an Try-Except Error handling example in my script. **Fig 3** shows what happens if the error handling is not used.



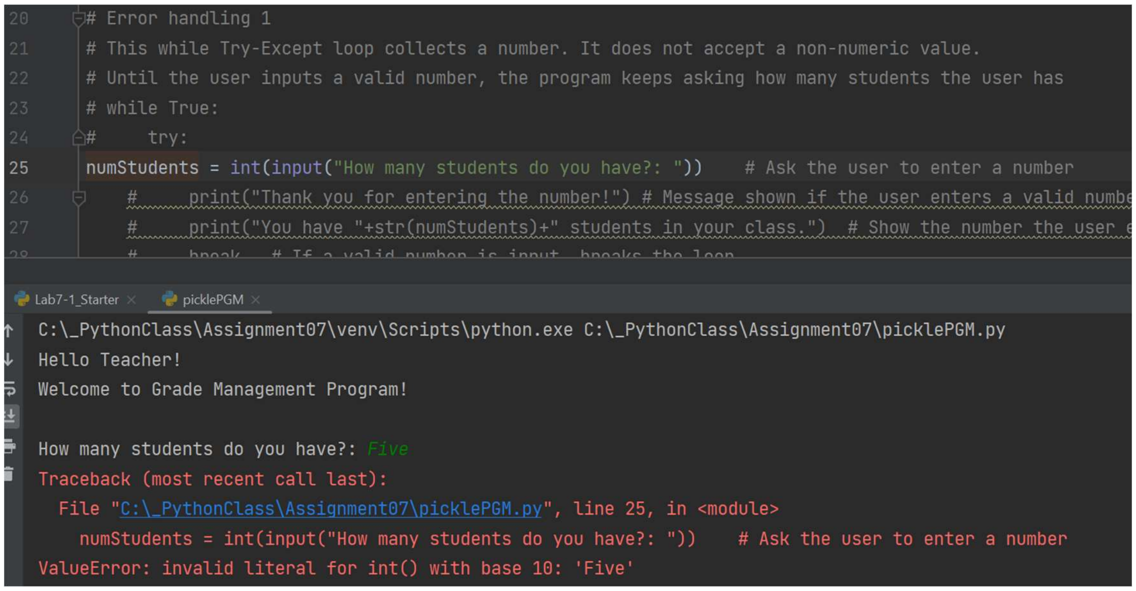
```
# Error handling 1
# This while Try-Except loop collects a number. It does not accept a non-numeric value.
# Until the user inputs a valid number, the program keeps asking how many students the user has
while True:
    try:
        numStudents = int(input("How many students do you have?: "))    # Ask the user to enter a number
        print("Thank you for entering the number!") # Message shown if the user enters a valid number
        print("You have "+str(numStudents)+" students in your class.")  # Show the number the user entered
        break   # If a valid number is input, breaks the loop
    except:
        print("That is not a valid number. Try again!") # If no valid number is provided, it keeps asking
```

while True

Lab7-1_Starter   picklePGM

```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\picklePGM.py
Hello Teacher!
Welcome to Grade Management Program!

How many students do you have?: Five
That is not a valid number. Try again!
How many students do you have?: 5
Thank you for entering the number!
You have 5 students in your class.
```

Fig 2 - Try-Except error handling example



```
# Error handling 1
# This while Try-Except loop collects a number. It does not accept a non-numeric value.
# Until the user inputs a valid number, the program keeps asking how many students the user has
# while True:
#     try:
numStudents = int(input("How many students do you have?: "))    # Ask the user to enter a number
    #     print("Thank you for entering the number!") # Message shown if the user enters a valid numbe
    #     print("You have "+str(numStudents)+" students in your class.")  # Show the number the user e
    #     break   # If a valid number is input, breaks the loop
```

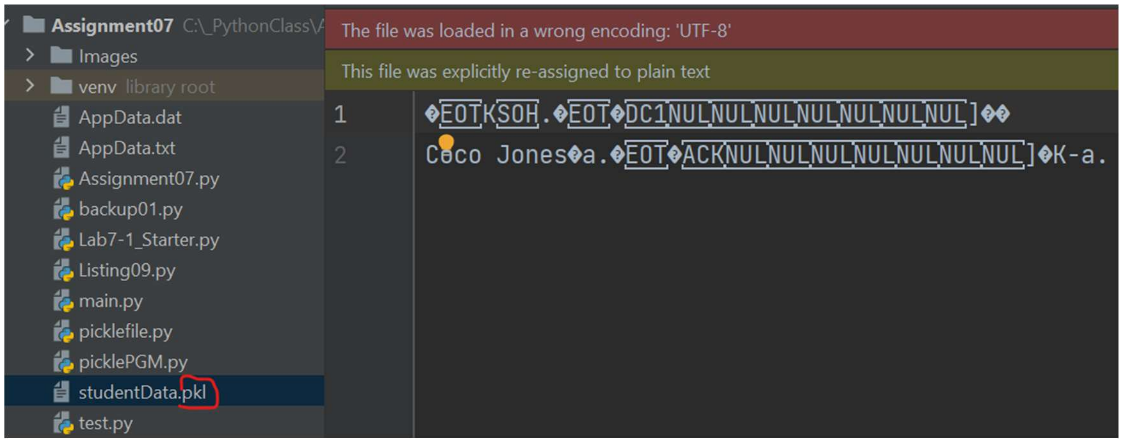
Lab7-1_Starter   picklePGM

```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\picklePGM.py
Hello Teacher!
Welcome to Grade Management Program!

How many students do you have?: Five
Traceback (most recent call last):
  File "C:\_PythonClass\Assignment07\picklePGM.py", line 25, in <module>
    numStudents = int(input("How many students do you have?: "))    # Ask the user to enter a number
ValueError: invalid literal for int() with base 10: 'Five'
```

Fig 3 - If no error handling is used, python error message is displayed and the program stops

What are the differences between a text file and a binary file?

One of the differences between a text file and a binary file is that a binary file is not for humans to read. The data on a binary file is obscure. Below is a binary file example that was created by the assignment script. A text file is readable and understandable to humans. Another difference is that a text file uses the **.txt** extension. But binary files can have **.pkl** or **.pickle** or **.bat** extensions. You can make up an extension for your binary file. In my assignment script, I gave the .pkl extension to my binary file. (**Fig 4**)



```
The file was loaded in a wrong encoding: 'UTF-8'
This file was explicitly re-assigned to plain text
1  �EOTKSOH.�EOT�DC1NULNULNULNULNULNULNUL]��
2  Coco Jones�a.�EOT�ACKNULNULNULNULNULNULNUL]�K-a.
```

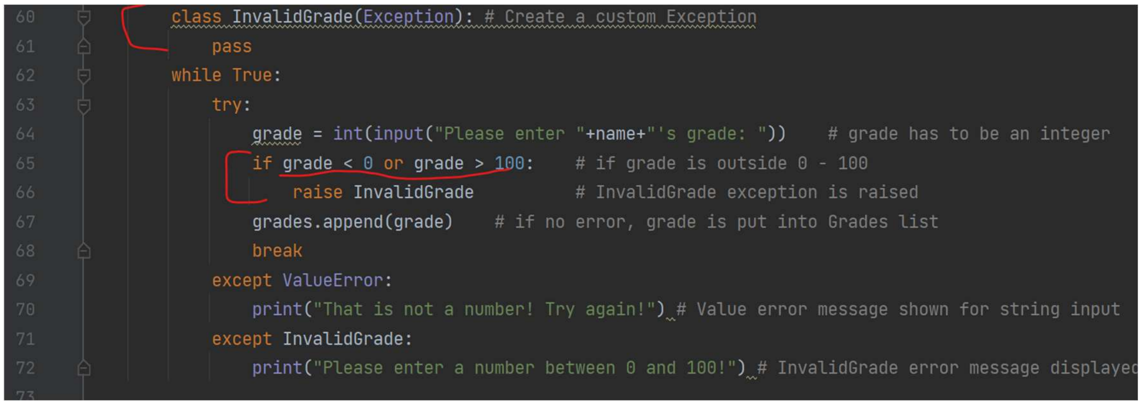
Fig 4 – What is in a binary file and a binary file can have an .pkl extension

How is the Exception class used?

The exception class is used to handle errors in Python. In **Fig 2** above, I showed the Try-Except block that is designed to catch an error. In the example, the user is expected to enter an integer to the program. If a non-integer value is entered, the Exception message "That is not a valid number. Try again!" is shown.

How do you "derive" a new class from the Exception class?

You can derive a new class by typing **class, your desired class name and (Exception)**. In my assignment script, I derived a new class from the Exception class. See **Fig 5**. I created a new class called InvalidGrade to raise an exception when the value the user inputs is smaller than 0 or larger than 100.

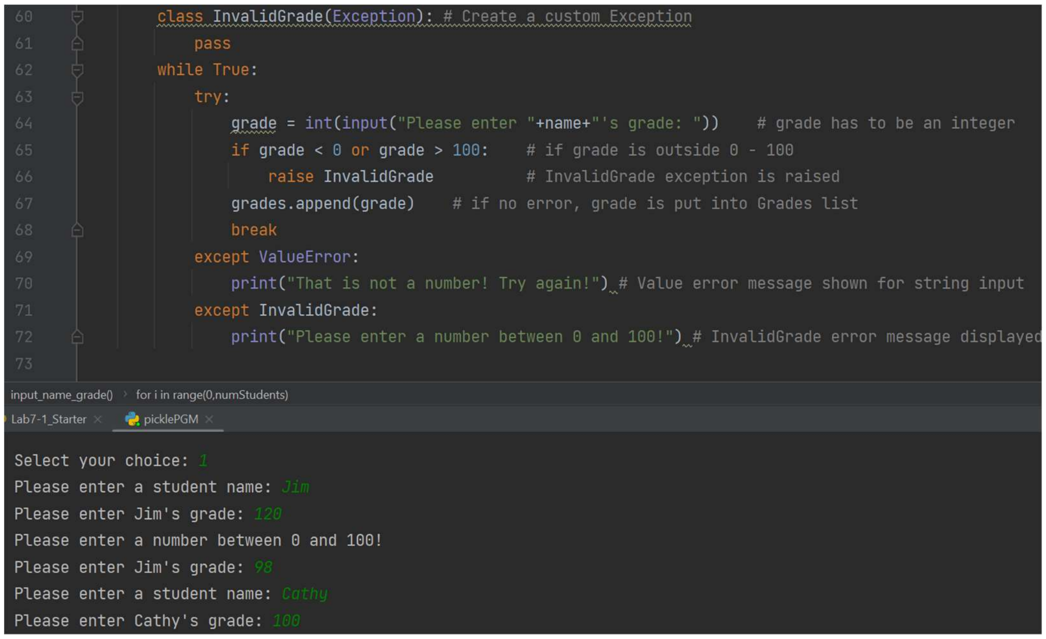


```python
class InvalidGrade(Exception): # Create a custom Exception
    pass
while True:
    try:
        grade = int(input("Please enter "+name+"'s grade: "))    # grade has to be an integer
        if grade < 0 or grade > 100:    # if grade is outside 0 - 100
            raise InvalidGrade          # InvalidGrade exception is raised
        grades.append(grade)    # if no error, grade is put into Grades list
        break
    except ValueError:
        print("That is not a number! Try again!") # Value error message shown for string input
    except InvalidGrade:
        print("Please enter a number between 0 and 100!") # InvalidGrade error message displayed
```

Fig 5 – Deriving a new class from Exception

When might you create a class derived from the Exception class?

When you cannot use any of the built-in exceptions in Python to handle an expected error, you can create a custom exception. In my script, I created a custom Exception to handle an error. My program asks the user to enter students' names and grades. Grades are between 0 and 100. No negative values or values over 100 are invalid values. In my script, I created a class **InvalidGrade(Exception)** to raise an Exception when a value over 100 or below 0 are entered and have the user enter the. In the example below, the user enters 120 for Jim's grade, the custom exception catches it and tells the user to enter the value between 0 and 100. (**Fig 6)**



```
        class InvalidGrade(Exception): # Create a custom Exception
            pass
        while True:
            try:
                grade = int(input("Please enter "+name+"'s grade: "))    # grade has to be an integer
                if grade < 0 or grade > 100:    # if grade is outside 0 - 100
                    raise InvalidGrade          # InvalidGrade exception is raised
                grades.append(grade)    # if no error, grade is put into Grades list
                break
            except ValueError:
                print("That is not a number! Try again!") # Value error message shown for string input
            except InvalidGrade:
                print("Please enter a number between 0 and 100!") # InvalidGrade error message displayed
```

input_name_grade() › for i in range(0,numStudents)

Lab7-1_Starter × picklePGM ×

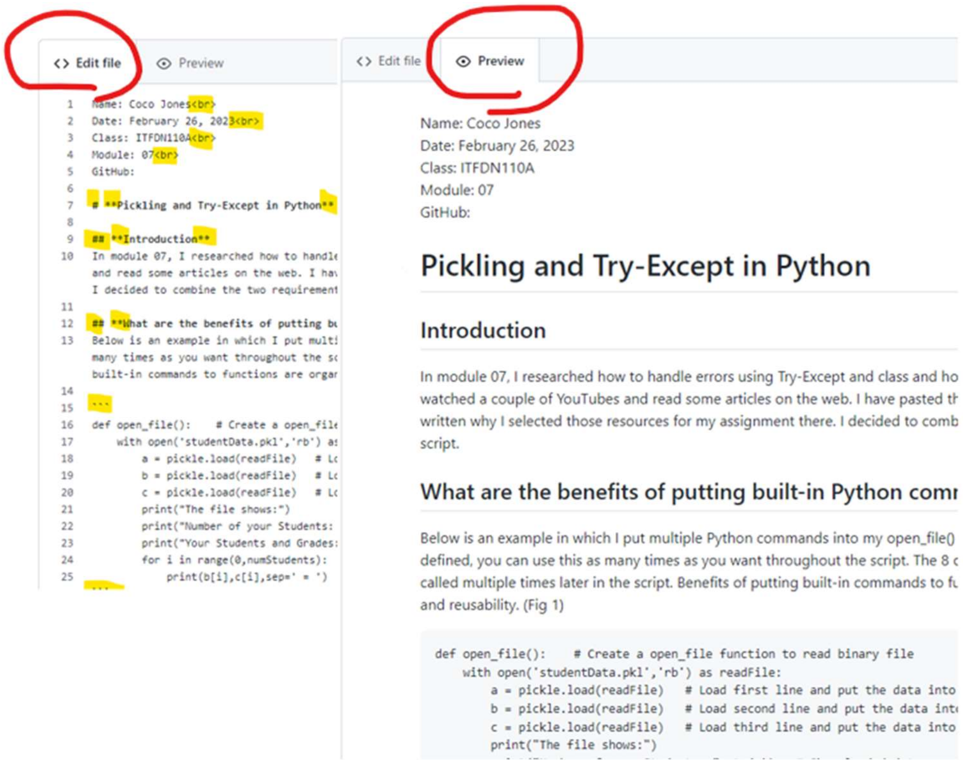
```
Select your choice: 1
Please enter a student name: Jim
Please enter Jim's grade: 120
Please enter a number between 0 and 100!
Please enter Jim's grade: 98
Please enter a student name: Cathy
Please enter Cathy's grade: 100
```

Fig 6 – Create a custom Exception to catch an unwanted value.

What is the Markdown language?

The Markdown language is used to format technical documents in plain text and is used on collaborative platforms like GitHub. This language is used to publish a copy of my Module 07 Assignment. The language is easily converted to HTML or PDF. That is one of the benefits of the Markdown language.

To make a header, you use #. To make your font bolder, you wrap the words with **. And to post your coding in a coding block, you wrap your coding with ```. <br> is used as carriage return. (**Fig 7)**



~~~
Name: Coco Jones<br>
Date: February 26, 2023<br>
Class: ITFDN110A<br>
Module: 07<br>
GitHub:

# **Pickling and Try-Except in Python**

## **Introduction**
In module 07, I researched how to handle
and read some articles on the web. I hav
I decided to combine the two requirement

## **What are the benefits of putting bu
Below is an example in which I put multi
many times as you want throughout the sc
built-in commands to functions are orgar

```
def open_file():    # Create a open_file
    with open('studentData.pkl','rb') as
        a = pickle.load(readFile)   # Lc
        b = pickle.load(readFile)   # Lc
        c = pickle.load(readFile)   # Lc
        print("The file shows:")
        print("Number of your Students:
        print("Your Students and Grades:
        for i in range(0,numStudents):
            print(b[i],c[i],sep=' = ')
```
~~~

Name: Coco Jones
Date: February 26, 2023
Class: ITFDN110A
Module: 07
GitHub:

Pickling and Try-Except in Python

Introduction

In module 07, I researched how to handle errors using Try-Except and class and ho
watched a couple of YouTubes and read some articles on the web. I have pasted th
written why I selected those resources for my assignment there. I decided to comb
script.

What are the benefits of putting built-in Python comr

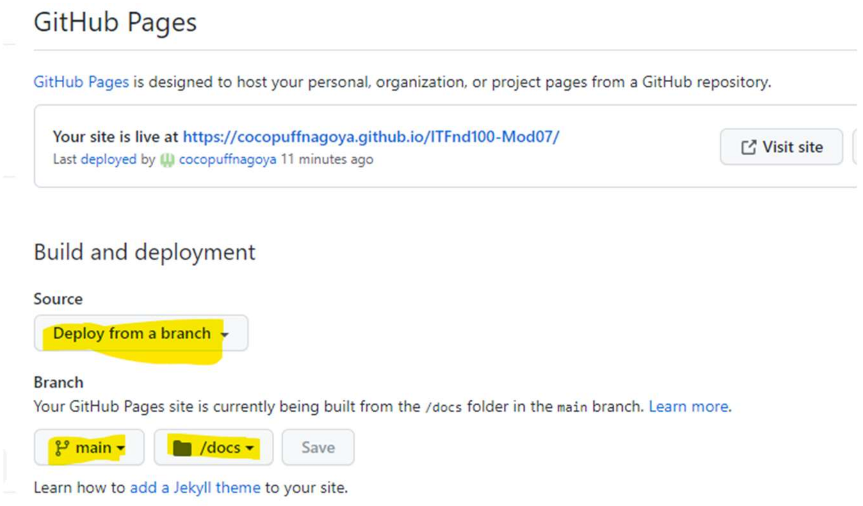
Below is an example in which I put multiple Python commands into my open_file()
defined, you can use this as many times as you want throughout the script. The 8 c
called multiple times later in the script. Benefits of putting built-in commands to fu
and reusability. (Fig 1)

```
def open_file():    # Create a open_file function to read binary file
    with open('studentData.pkl','rb') as readFile:
        a = pickle.load(readFile)   # Load first line and put the data into
        b = pickle.load(readFile)   # Load second line and put the data int
        c = pickle.load(readFile)   # Load third line and put the data into
        print("The file shows:")
```

Fig 7 – Markdown language to publish my homework

How do you use Markdown on a GitHub webpage?

First you need to create a webpage. Create a repository and go to Settings and Pages. Select Deploy from a branch, main and /docs and save the settings. (**Fig 8**) Your page will be available within 20 minutes.



GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at https://cocopuffnagoya.github.io/ITFnd100-Mod07/
Last deployed by cocopuffnagoya 11 minutes ago

Build and deployment

Source
Deploy from a branch

Branch
Your GitHub Pages site is currently being built from the /docs folder in the main branch. Learn more.

main
/docs

Learn how to add a Jekyll theme to your site.

Fig 8 – GitHub page settings

Then go to the <> Code tab. You can paste your plain text there and start editing using #, ##, <br>, ** as explained above in Fig 7 above. You need to go back and forth between Edit file and Preview. Once you are finished, click on Commit changes to save.

Summary

In module 07, I created one script that has pickling and Try-Except blocks. I used online resources listed below at the References. I will add comments why I used those resources and why I though they were relevant.

References

https://www.youtube.com/watch?v=LjtIKL17Bpw **Python Tutorial 15: Solution to Python Pickle Homework Example Problems (**Last accessed: 2/26/2023)

I found this material very useful to understand how to pickle.dump and pickle.load data and how pickling works in Python. I tweaked and used his script as part of my assignment script. Added Try-Except exceptions to it and put his commands into functions and added the menu options to complete my script.

https://www.youtube.com/watch?v=b0q9vVgAMq8 **Try Catch with Loop (Python)** (Last accessed: 2/26/2023)

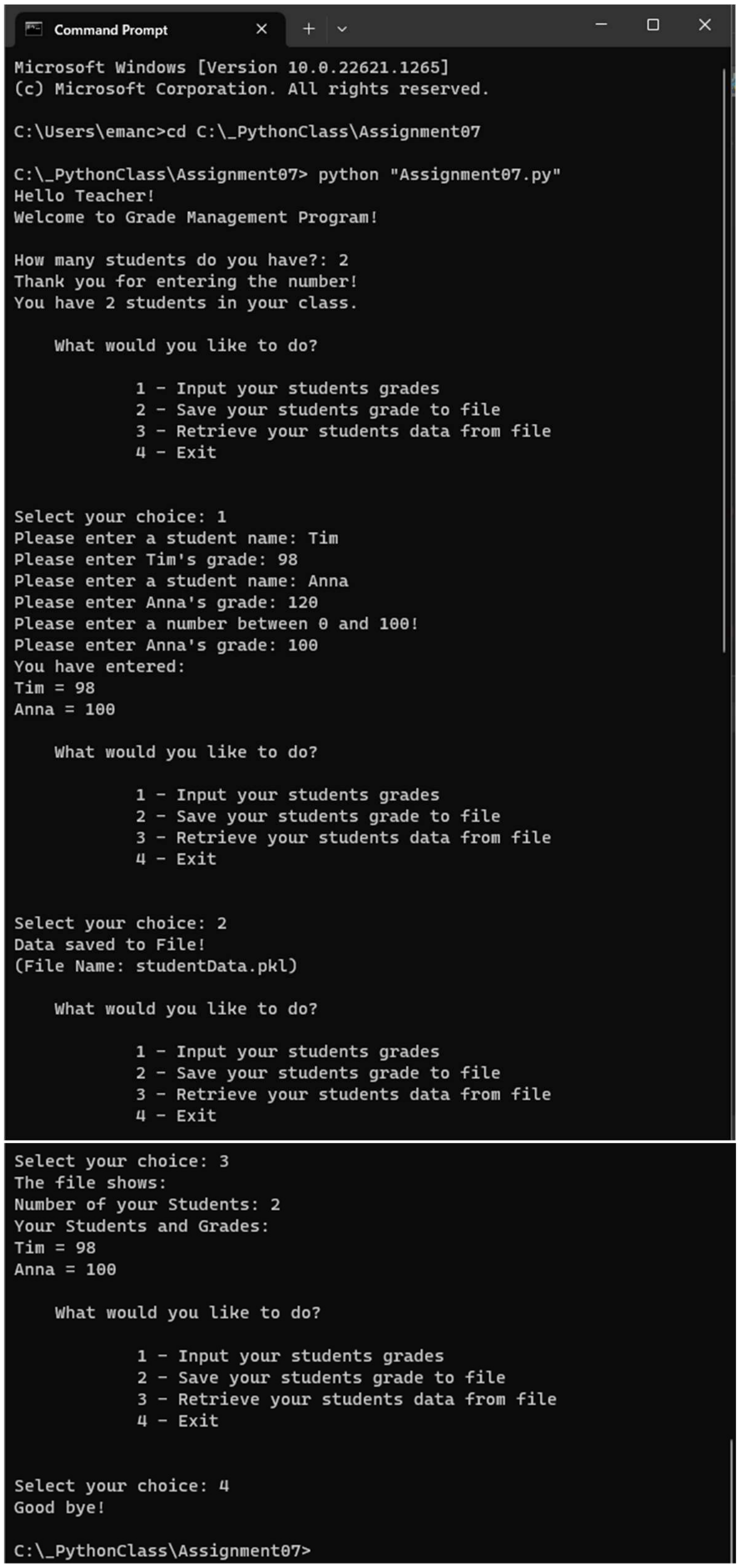
This material was useful for me to understand how to repeat the prompt until the user inputs a value in the correct format. With use of while True shown in this video, I was able to loop the step until the user inputs a value in an expected format.

https://www.programiz.com/python-programming/user-defined-exception **Python Custom Exceptions** (Last accessed: 2/25/2023)

I read this article when I was looking for how to define my own Exception to use in my script. This article explains how to create a custom Exception very well.

Python Programming, Third Edition, Course Technology, Michael Dawson 2019 ISBN-13 978-1-4354-5500-9

Command Shell:

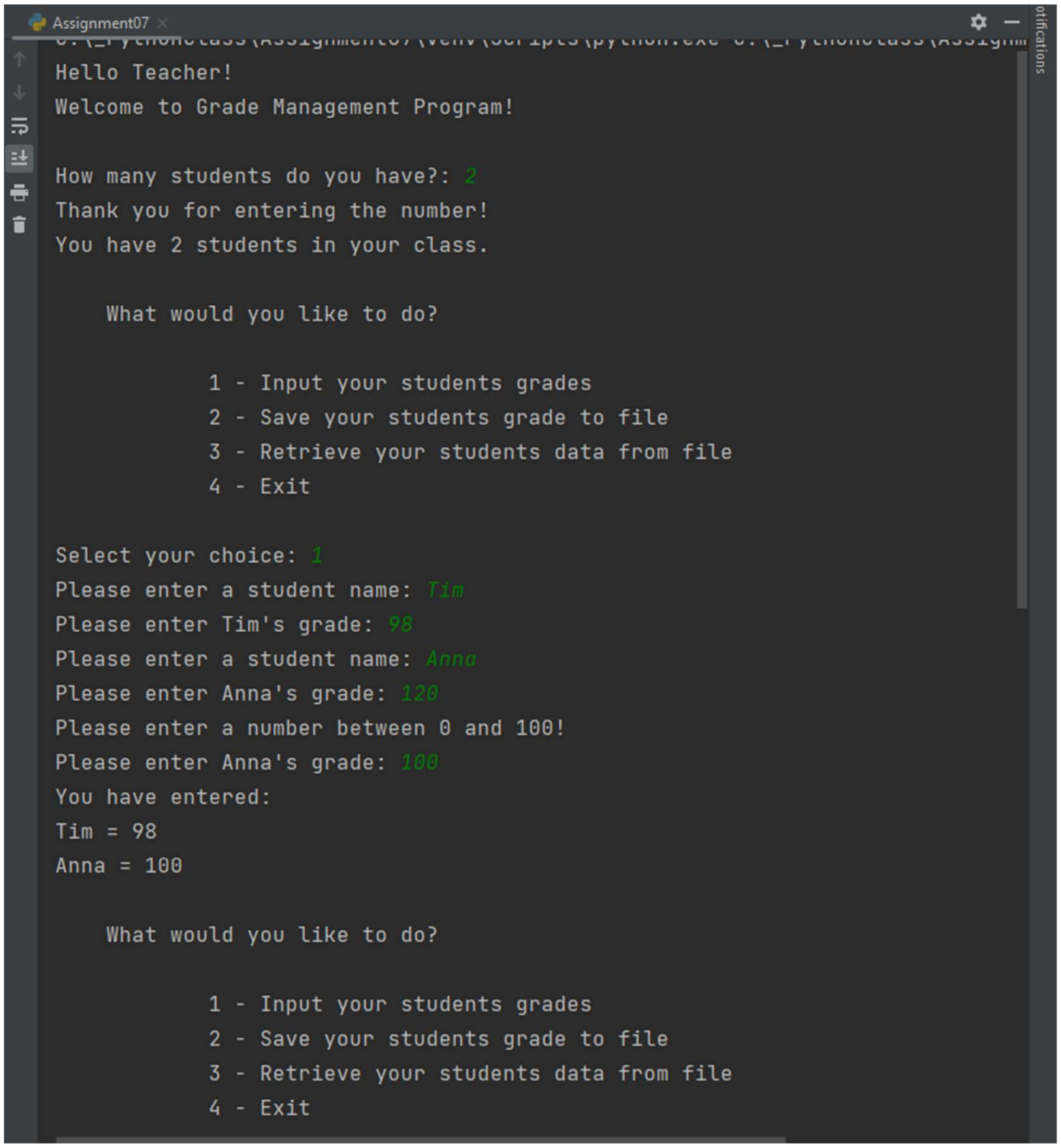


```
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emanc>cd C:\_PythonClass\Assignment07

C:\_PythonClass\Assignment07> python "Assignment07.py"
Hello Teacher!
Welcome to Grade Management Program!

How many students do you have?: 2
Thank you for entering the number!
You have 2 students in your class.

    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit


Select your choice: 1
Please enter a student name: Tim
Please enter Tim's grade: 98
Please enter a student name: Anna
Please enter Anna's grade: 120
Please enter a number between 0 and 100!
Please enter Anna's grade: 100
You have entered:
Tim = 98
Anna = 100

    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit


Select your choice: 2
Data saved to File!
(File Name: studentData.pkl)

    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit
Select your choice: 3
The file shows:
Number of your Students: 2
Your Students and Grades:
Tim = 98
Anna = 100

    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit


Select your choice: 4
Good bye!

C:\_PythonClass\Assignment07>
```

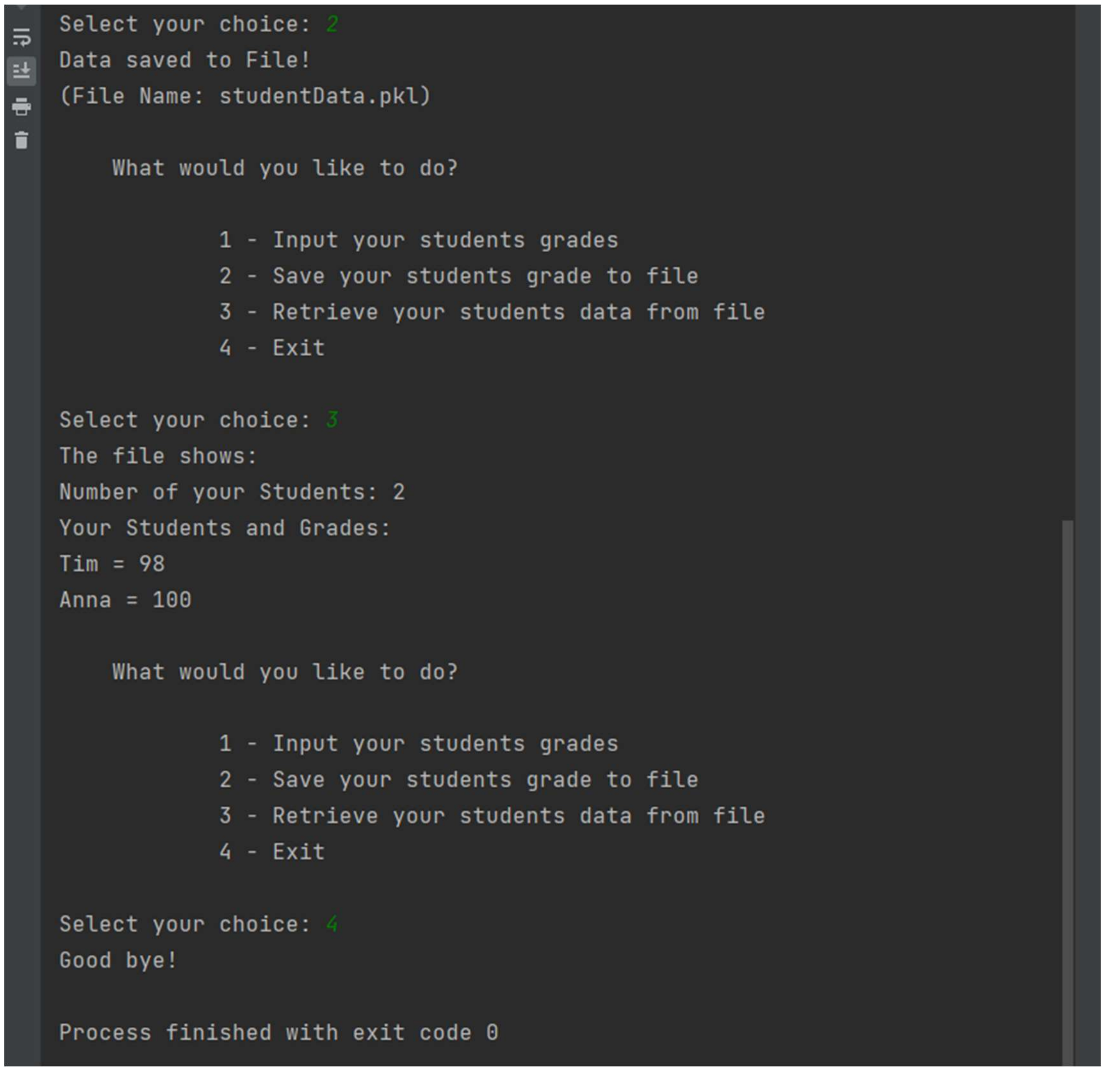
PyCharm:

```
# ------------------------------------------------- #
# Title: Homework Grade Management Program
# Description: This program was designed for teachers
# to save students names and grades to file and retrieve the saved data
# ChangeLog: (Who, When, What)
# CJones, 2/26/2023, created the script
# ------------------------------------------------- #

import pickle   # import pickle to be able to dump and load data in the
program

# -- Data -- #
name = ''    # Variable to assign student name. String
grade = 0    # Student's grade. Defaulted to 0. Integer
names = []   # List of students names
grades = [] # List of students grades
file = 'studentData.pkl'     # Binary file to save pickled data

# Show Welcome message to the user in the beginning of the program
print("Hello Teacher!\nWelcome to Grade Management Program!\n")

# Error handling 1
# This while Try-Except loop collects a number. It does not accept a non-
numeric value.
# Until the user inputs a valid number, the program keeps asking how many
students the user has
while True:
    try:
        numStudents = int(input("How many students do you have?: "))     # Ask
the user to enter an integer
        print("Thank you for entering the number!") # Message shown if the
user enters a valid number
        print("You have "+str(numStudents)+" students in your class.")   #
Show the number the user entered
        break   # Break the loop when a valid number is input
    except:
        print("That is not a valid number. Try again!") # Keep asking to try
again until a valid number is input

# -- Processing -- #
# Pickling
# Function to save data to binary file
def save_to_binary_file(file):
    with open(file, 'wb') as dataFile: # Open binary file
        pickle.dump(numStudents, dataFile)  # Dump number of students onto
dataFile
        pickle.dump(names,dataFile) # Dump students names to dataFile
        pickle.dump(grades,dataFile) # Dump students grades to dataFile
    print("Data saved to File!\n(File Name: studentData.pkl)")  # Show file
saved message

# Function to read binary data
def open_file(file):    # Create a open_file function to read binary file
    with open(file,'rb') as readFile:
        a = pickle.load(readFile)   # Load first line and put the data into a
        b = pickle.load(readFile)   # Load second line and put the data into
b
        c = pickle.load(readFile)   # Load third line and put the data into a
        print("The file shows:")
        print("Number of your Students: "+str(a))   # Show loaded data
        print("Your Students and Grades:")
        for i in range(0,numStudents):  # Show loaded data
            print(b[i],c[i],sep=' = ')  # Show loaded data

# -- Presentation (I/O) -- #
# Function to input name and grade
def input_name_grade(name, grade):
    for i in range(0,numStudents):  # Loop the following process numStudents
times
        name = input("Please enter a student name: ")   # Ask for student
name
        names.append(name)  # Put the name to names list
        # Error handling 2 - accepts only a number
        class InvalidGrade(Exception): # Create a custom Exception
            pass
        while True:
            try:
                grade = int(input("Please enter "+name+"'s grade: "))    #
grade has to be an integer
                if grade < 0 or grade > 100:    # if grade is outside 0 - 100
                    raise InvalidGrade          # InvalidGrade exception is
raised
                grades.append(grade)    # if no error, grade is put into
Grades list
                break
            except ValueError:
                print("That is not a number! Try again!") # Value error
message shown for string input
            except InvalidGrade:
                print("Please enter a number between 0 and 100!") #
InvalidGrade error message displayed

    print("You have entered: ") # Show the data the user has just entered
    for i in range(0,numStudents):  # Loop numStudents times
        print(names[i],grades[i],sep=' = ') # Show ith person's name and
grade

# Menu - Menu is shown to the user until 4 is selected to exit the program
while True:
    print('''
    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit
            ''')

    choice = int(input("Select your choice: ")) # Ask user to input a number
1-4
    if choice not in (1,2,3,4): # If user inputs other than 1,2,3,4
        print("Enter 1,2,3 or 4!")  # tell user to input a number 1,2,3 or 4
        continue    # Continue asking
    elif choice == 1:   # if choice is 1
        input_name_grade(name, grade)   # call input_name_grade() function
    elif choice == 2:   # if choice is 2
        save_to_binary_file(file)   # call save_to_binary_file() function
    elif choice == 3:   # if choice is 3
        open_file(file) # call open_file() function
    else:  # if choice is 4
        break   # exit the program
print("Good bye!")
```

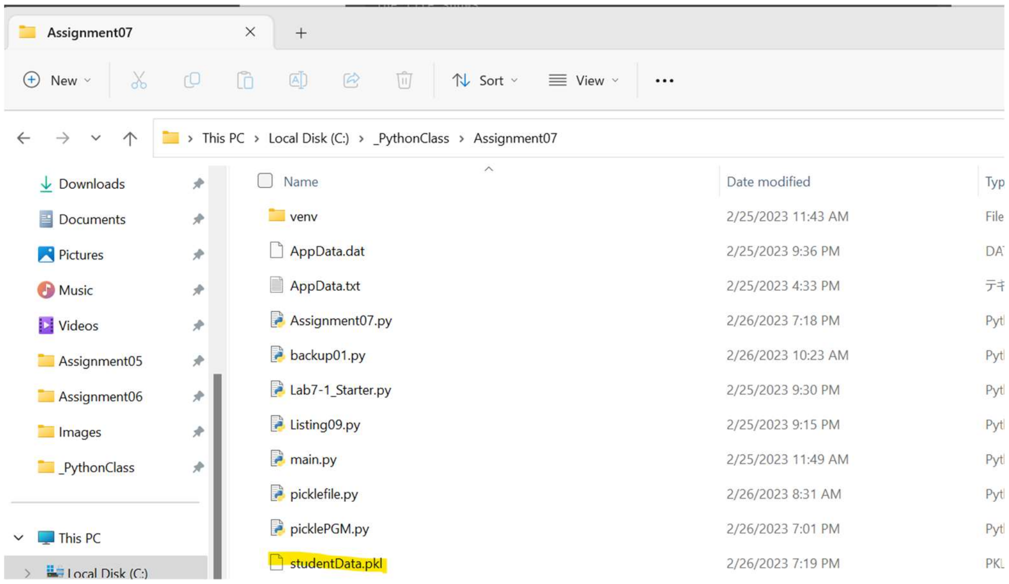
Terminal:



```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignm
Hello Teacher!
Welcome to Grade Management Program!

How many students do you have?: 2
Thank you for entering the number!
You have 2 students in your class.

    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit

Select your choice: 1
Please enter a student name: Tim
Please enter Tim's grade: 98
Please enter a student name: Anna
Please enter Anna's grade: 120
Please enter a number between 0 and 100!
Please enter Anna's grade: 100
You have entered:
Tim = 98
Anna = 100

    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit
```



```
Select your choice: 2
Data saved to File!
(File Name: studentData.pkl)

    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit

Select your choice: 3
The file shows:
Number of your Students: 2
Your Students and Grades:
Tim = 98
Anna = 100

    What would you like to do?

            1 - Input your students grades
            2 - Save your students grade to file
            3 - Retrieve your students data from file
            4 - Exit

Select your choice: 4
Good bye!

Process finished with exit code 0
```



Name	Date modified	Typ
venv	2/25/2023 11:43 AM	File
AppData.dat	2/25/2023 9:36 PM	DA
AppData.txt	2/25/2023 4:33 PM	テキ
Assignment07.py	2/26/2023 7:18 PM	Pyt
backup01.py	2/26/2023 10:23 AM	Pyt
Lab7-1_Starter.py	2/25/2023 9:30 PM	Pyt
Listing09.py	2/25/2023 9:15 PM	Pyt
main.py	2/25/2023 11:49 AM	Pyt
picklefile.py	2/26/2023 8:31 AM	Pyt
picklePGM.py	2/26/2023 7:01 PM	Pyt
studentData.pkl	2/26/2023 7:19 PM	PKL