

ADVANCED COLLECTIONS AND ERROR HANDLING

Introduction

Python provides a module called collections that includes several advanced data structures which extend the functionality of python's built-in data structures. This document explains in about dictionary collections and error handling. A dictionary in Python is a data structure that stores values in key-value pairs. Dictionary elements are accessed via keys. Error handling is done using exceptions that are caught in try blocks and handled in except blocks.

Dictionaries

Dictionary is a collection of key-value pairs that are unordered, changeable, and indexed. Each key-value pair maps the key to its associated value. This assignment demonstrates how dictionary rows being added to the lists. Values in the dictionary can be of any datatype. A dictionary is defined by enclosing the key-value pairs in curly braces {}, with keys and values separated by a colon, and pairs separated by commas.

Working with Files

Reading, writing, and appending files in Python are fundamental operations that allows to interact with external files.

Opening and Closing Files: Python provides the `open()` function to open a file. This function returns a file object which can be used to read from or write to the file. Once done with the file, we can close the file using `close ()` function. With statement automatically closes the file when the block of code is exited.

Reading and Writing Files: `open()` function takes the filename and mode as arguments. The mode 'r' for reading, 'w' for writing, 'a' for appending and 'r+' for both reading and writing.

Write and append modes: When we open a file in write mode, the existing content of the file is erased, and the file is treated as a new empty file and then write content into the file. If the file does not exist, a new one will be created. In append mode, the file pointer is at the end of the file if the file exists. This means that when we write to the file, the new content will be added at the end of the existing content, preserving the original data. If the file does not exist, a new one will be created.

Exceptions

In Python, there are several built-in Python exceptions that can be raised when an error occurs during the execution of a program. Here are some of the most common types of exceptions in Python:

SyntaxError: This exception is raised when the interpreter encounters a syntax error in the code, such as a misspelled keyword, a missing colon, or an unbalanced parenthesis.

TypeError: This exception is raised when an operation or function is applied to an object of the wrong type, such as adding a string to an integer.

NameError: This exception is raised when a variable or function name is not found in the current scope.

IndexError: This exception is raised when an index is out of range for a list, tuple, or other sequence types.

KeyError: This exception is raised when a key is not found in a dictionary.

ValueError: This exception is raised when a function or method is called with an invalid argument or input, such as trying to convert a string to an integer when the string does not represent a valid integer.

AttributeError: This exception is raised when an attribute or method is not found on an object, such as trying to access a non-existent attribute of a class instance.

IOError: This exception is raised when an I/O operation, such as reading or writing a file, fails due to an input/output error.

ZeroDivisionError: This exception is raised when an attempt is made to divide a number by zero.

ImportError: This exception is raised when an import statement fails to find or load a module.

Errors and Exception

Errors are the problems in a program due to which the program will stop the execution. Exceptions are raised when internal events occur that change the normal flow of the program. Incorrect indentation can lead to an exception. Both errors and exceptions are a type of runtime error, which means they occur during the execution of a program.

Try-Except

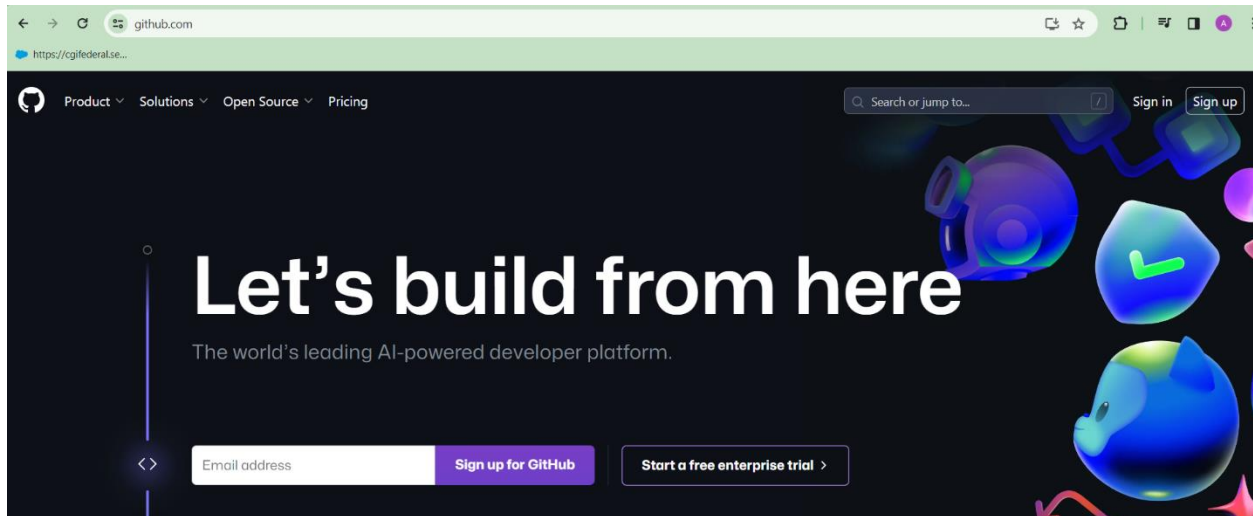
try: code with the exception(s) to catch. If an exception is raised, it jumps straight into the except block.

except: this code is only executed if an exception occurred in the try block. The except block is required with a try block, even if it contains only the pass statement. This prevents abrupt exits of the program on error.

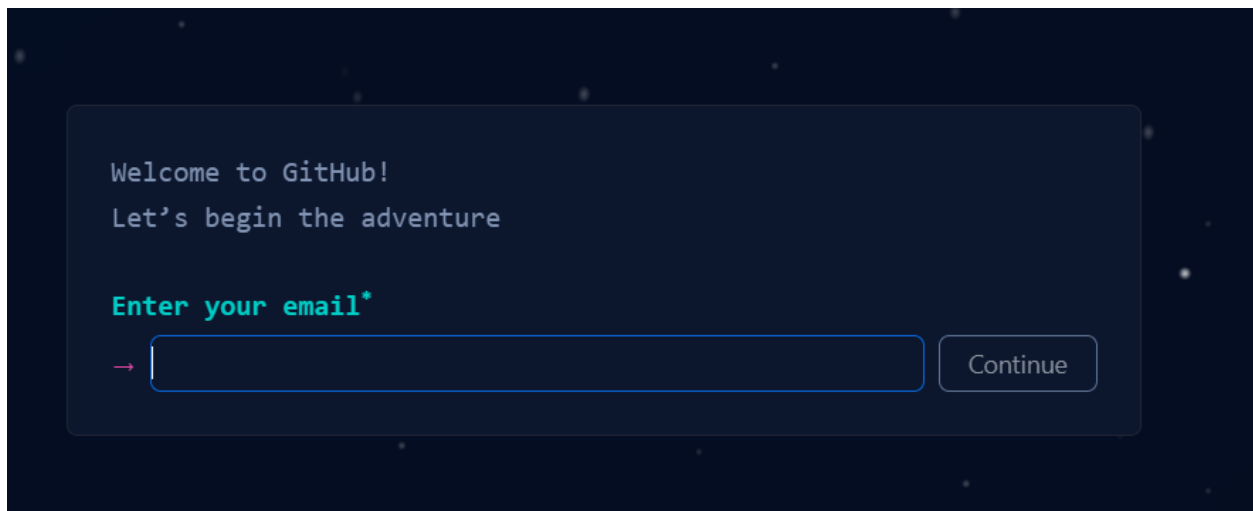
GitHub

GitHub is a website and service that hosts and tracks changes in software projects using Git, an open-source version control system. It allows developers to manage changes to their code efficiently. When developers work on a project, they make constant changes to the code. Git helps keep track of these revisions by storing modifications in a central repository. GitHub is often described as a cloud-based platform for code hosting and collaboration rather than a traditional cloud file sharing service. GitHub allows multiple developers to work on a single project at the same time, reduces the risk of duplicative or conflicting work, and can help decrease production time.

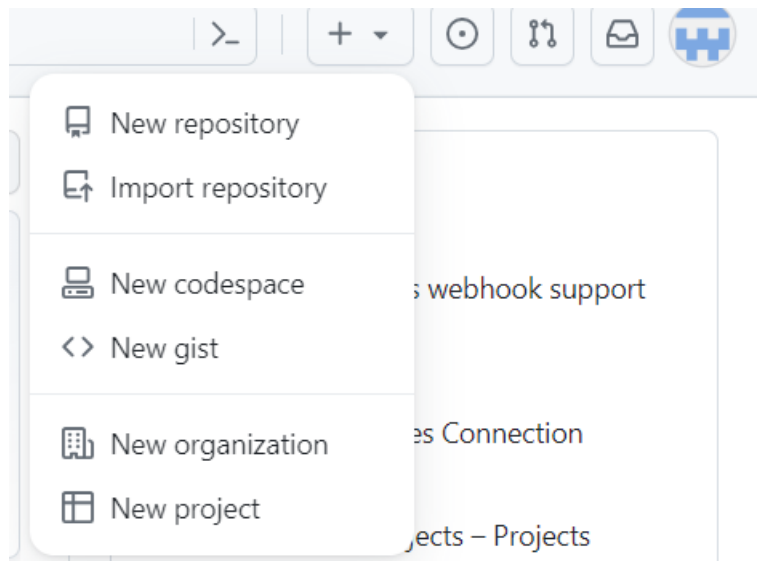
To get started using GitHub, we must create an account. Login to <https://github.com>



Click for signup and enter the necessary details to create an account.



To create a repository, click on new repository give the name and configure it to be either private or public. If the repository is public, it can easily be seen and shared with other people. One account can have many repositories, each one used for a different project of purpose.



From the drop-down menu, choose the option that says new repository. Ensure the option “add a README file” is selected. Click the Create repository button to complete the process and create your new repository.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)



Required fields are marked with an asterisk ().*

Owner * Repository name *

 abarna47 ▾ /

Great repository names are short and memorable. Need inspiration? How about **refactored-sniffle** ?

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

Once the repository is created, we can upload files to it. These files can be viewed, downloaded, and modified by anyone with a link. We can upload the file by clicking on add file → upload files.

+ Create new file

⬆ Upload files

43 minutes ago

43 minutes ago

```

# ----- #
# Title: Assignment05
# Desc: This assignment demonstrates data processing using dictionaries and exception handling
# Change Log: (Who, When, What)
# Abarna,2/12/2024, Created Script
# ----- #
#include sys module
import sys

#creating constants for a course registration program
MENU: str = """
----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----
"""
FILE_NAME: str = "Enrollments.csv"

#variable declaration
student_first_name: str = ""
student_last_name: str = ""
course_name: str = ""
file = None
csv_data: str = ""
menu_choice: str = ""
student_data: dict = {}
students: list = []

#Reading from Enrollments.csv
try:
    with open(FILE_NAME, 'r') as file:
        all_students = file.readlines()

#file not found error
except FileNotFoundError:
    print(f"Error : The file {FILE_NAME} was not found")

```

When the program starts, it automatically loads the contents of “Enrollments.csv” into a list of dictionaries.

On menu choice 1, the program prompts the user for the student’s first and last names, and the course name. These inputs are stored in dictionary and added to list.

On menu choice 2, the program formats the collected data into a comma-separated string and displays it.

On menu choice 3, the program writes the content of list of dictionaries to the file “Enrollments.csv”, then closes the file and displays the stored data.

On menu choice 4, the program terminates.

```

#list of strings to list of dictionary
for student in all_students:
    try:
        csv_data = student.strip().split(',')
        student_data = {
            "student_first_name" : csv_data[0],
            "student_last_name" : csv_data[1],
            "course_name" : csv_data[2]
        }
        students.append(student_data)

    # Index out of range
    except IndexError:
        print("Error: Not enough data fields")

#Creating a while loop
while True:
    print(MENU)
    menu_choice = input("what would you like to do: ")

    #using match-case statement
    match menu_choice:

        #collect user data and add to the list of dictionary rows
        case "1":
            try:
                student_first_name = input("Enter student's first name: ")
                student_last_name = input("Enter student's last name: ")

                #value error
                except ValueError:
                    print(f'Name should not contain numeric values')

                course_name = input("Enter the course name: ")
                student_data = {
                    "student_first_name" : student_first_name,
                    "student_last_name" : student_last_name,
                    "course_name" : course_name
                }
                students.append(student_data)

            #print data to the user
            case "2":
                for student in students:
                    print(f"{student['student_first_name']}\t\t{student['student_last_name']}\t\t{student['course_name']}")

        #writing data to the file
        case "3":
            try:
                file = open(FILE_NAME, 'w')
                for student in students:
                    file.write(f"{student['student_first_name']},{student['student_last_name']},{student['course_name']}\n")
                print(f"You have registered {student['student_first_name']} {student['student_last_name']} for {student['course_name']}")
                file.close()

            #Input/Output operation fails
            except IOError:
                print("Error: There was an issue writing to the file.")

        #exit out of the program
        case "4":
            print("program ends")
            sys.exit()
        case _:
            print("Choose one of the options from above")

```


The output,

```
----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----

what would you like to do: 1
Enter student's first name: abarna
Enter student's last name: sharankumar
Enter the course name: python100

----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----

what would you like to do: 1
Enter student's first name: max
Enter student's last name: goodwin
Enter the course name: python3.0

----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----

what would you like to do: 2
abarna          sharankumar          python100
max             goodwin             python3.0
```

```
what would you like to do: 3
You have registered abarna sharankumar for python100
You have registered max goodwin for python3.0
```

```
----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----
```

```
what would you like to do: 4
program ends
```

```

----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----

```

```

what would you like to do: 1
Enter student's first name: sarah
Enter student's last name: williams
Enter the course name: python100

```

```

----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----

```

```

what would you like to do: 2
abarna          sharankumar          python100
max             goodwin             python3.0
sarah           williams             python100

```

```

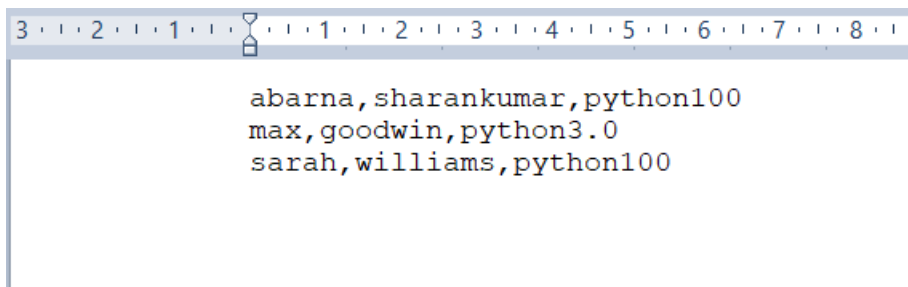
----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----

```

```

what would you like to do: 3
You have registered abarna sharankumar for python100
You have registered max goodwin for python3.0
You have registered sarah williams for python100

```



In PyCharm,

```
what would you like to do: 1
Enter student's first name: Max
Enter student's last name: goodwin
Enter the course name: python100

----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----

what would you like to do: 1
Enter student's first name: sarah
Enter student's last name: williams
Enter the course name: python2.0
```

```
what would you like to do: 2
Max      goodwin      python100
sarah    williams     python2.0

----Course Registration Program----
Select from the following menu
1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program
-----

what would you like to do: 3
You have registered Max goodwin for python100
You have registered sarah williams for python2.0
```

```
what would you like to do: 1
Enter student's first name: abarna
Enter student's last name: sharankumar
Enter the course name: python100
```

```
----Course Registration Program----
```

```
Select from the following menu
```

1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program

```
-----
```

```
what would you like to do: 2
Max      goodwin      python100
sarah    williams     python2.0
abarna   sharankumar  python100
```

```
what would you like to do: 3
You have registered Max goodwin for python100
You have registered sarah williams for python2.0
You have registered abarna sharankumar for python100
```

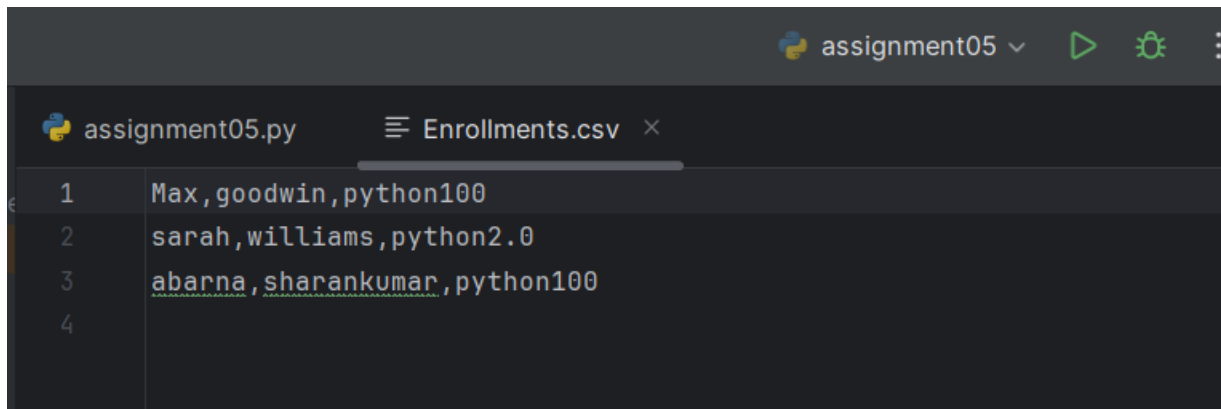
```
----Course Registration Program----
```

```
Select from the following menu
```

1. Register a student for a course
2. Show current data
3. Save data to the file
4. Exit the program

```
-----
```

```
what would you like to do: 4
program ends
```



The screenshot shows a code editor window with two tabs: 'assignment05.py' and 'Enrollments.csv'. The 'Enrollments.csv' tab is active, displaying a CSV file with three rows of data. The first row is 'Max,goodwin,python100', the second is 'sarah,williams,python2.0', and the third is 'abarna,sharankumar,python100'. The third row is highlighted with a green selection bar. The editor has a dark theme and includes standard icons for running, debugging, and settings in the top right corner.

Line	Student Name	Course
1	Max,goodwin	python100
2	sarah,williams	python2.0
3	abarna,sharankumar	python100

Summary

This Python program is a course registration system with four main functionalities. It allows users to register a student for a course, view the current data, save the data to a file, and exit the program. On startup, it reads data from an “Enrollments.csv” file into a list of dictionaries. The program shows a list of options to the user. When you add or change information in the program the program saves these changes to a file. This way, the information is not lost when you close the program, and we can see it again the next time we open the program. The program ends when the user chooses to exit.

