Junyan Lu

May 31, 2023

Foundation of Programming: Python

Assignment 07

GitHubURL:https://github.com/LU99IS99/IntroToProg-Python-Mod07.git

Creating scripts using custom functions, files, and structured error handling

Introduction

In this module, I delved into the use of custom functions and try-except blocks to organize file management code and provide custom error handling. By leveraging custom functions, I can break down complex tasks into manageable units, promoting code modularity and reusability. The implementation of try-except blocks allows me to gracefully handle errors and provide customized error messages, enhancing the user experience. Additionally, I explored ways to improve the appearance of GitHub webpages, emphasizing the importance of project structure, documentation clarity, and visual design in creating a more professional impression for potential collaborators and users.

The benefits of putting built-in Python command into functions

Putting built-in Python commands into functions offers several benefits:

- Reusability: Functions allow me to reuse code in multiple parts of my program without writing the same commands over and over again.
- Organization: Functions help me break down my code into smaller, self-contained units. This makes my code easier to read, understand, and maintain.
- Simplification: Functions abstract away the details of the built-in commands, allowing me to call a function instead of repeating the same code throughout my program. This simplifies the main flow of my program.
- Readability: Functions let me give meaningful names to blocks of code, making it easier to understand the purpose and functionality of each part of my program.
- Encapsulation: Functions allow me to combine built-in commands with additional logic and data manipulation, keeping related code together and improving code organization.

The benefits of using structured error handling

Using structured error handling, such as try-except blocks in Python, offers the following benefits:

- Error Control: It allows me to control and manage errors that might occur during the execution of my program. Instead of my program crashing abruptly when an error occurs, I can catch and handle the error in a controlled manner.
- Graceful Handling: Structured error handling enables me to handle errors gracefully by providing alternative actions or recovery mechanisms. This helps prevent my program from completely breaking and allows it to continue running smoothly.
- User-Friendly: By implementing structured error handling, I can provide meaningful error messages to users, explaining what went wrong and suggesting possible solutions. This improves the user experience by offering clear guidance when errors occur.
- Debugging Assistance: Structured error handling provides valuable information about the cause of errors, making it easier to identify and fix issues in my code. It helps my pinpoint the exact location and type of error, aiding in the debugging and troubleshooting process.
- Program Reliability: With structured error handling, my program becomes more reliable and robust. It can handle unexpected situations and recover from errors, ensuring that critical parts of my program can still execute even if errors occur.

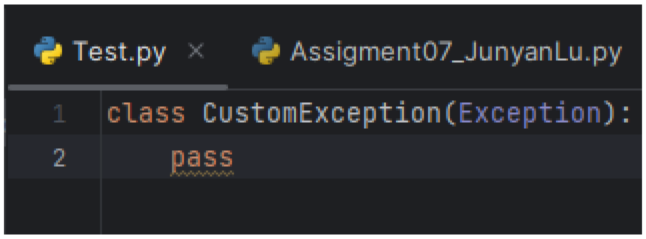
Difference between Parameters and Arguments

Text files and binary files differ in the way they store and represent data:

- Data Representation: Text files store data as human-readable text using characters, such as letters, numbers, and symbols. Binary files, on the other hand, store data as raw bytes, which are sequences of 0s and 1s.
- Readability: Text files can be opened and read using a text editor, and the content can be easily understood by humans. Binary files, however, are not meant to be read directly by humans as they contain encoded or complex data that requires specialized software or knowledge to interpret.
- Content Type: Text files are used for storing textual information, such as documents, code, or configuration files. Binary files are used for storing non-textual data, such as images, audio, video, or program executables.
- File Size: Text files tend to have larger file sizes compared to binary files. This is because text files store each character using a specific number of bytes, while binary files can represent data more efficiently by using the minimum number of bytes required.
- Editing: Text files can be easily edited and modified using a text editor or word processor. You can add, delete, or modify the text directly. Binary files, however, require specialized tools or programming to modify their content because they store data in a format that may have specific structures or encodings.

Exception class used and when to create a class derived from the Exception class

To derive a new class from the “Exception” class in Python, you can create a new class and make it a subclass of “Exception”. This allows you to define a custom exception type with specific behavior or properties. Here's is an example about “Exception” class:



```
class CustomException(Exception):
    pass
```

In this example, CustomException is a new class derived from Exception. The pass statement is used to indicate that the class doesn't have any additional methods or properties defined. I might create a class derived from the “Exception” class when I want to handle a specific type of error or exceptional situation in my code. It allows me to define my own custom exception with a unique name and behavior.

Markdown language

Markdown is a simple and easy-to-use language for formatting text. It allows me to add basic formatting elements like headers, lists, links, and bold or italic styles to my plain text. The great thing about Markdown is that I don't need to worry about complex formatting codes or tags. I can write my content using plain text, and when IA convert it to HTML or other formats, the Markdown processor will handle the formatting for me. It's widely used for creating documentation, writing blog posts, and formatting text on platforms like GitHub.

Use Markdown on a GitHub webpage

To use Markdown on a GitHub webpage, follow these easy steps:

- Create or open a file with a .md or .markdown extension on GitHub repository.
- Click the "Edit" button to enter the editing mode.
- Write content using simple Markdown syntax.
- Use # for headers, - or * for lists, [text](link) for links, **text** for bold, and _text_ for italic.
- Preview changes to see how the content will appear.
- Make any necessary adjustments.
- Provide a commit message and save changes.
- GitHub will render the Markdown content and display it on the webpage.