Abarna Sharankumar
Jan 28,2024
Foundations of Programming: Python
Assignment 06

FUNCTIONS

Introduction

This document explains the fundamental building blocks of programming which is functions and classes. Functions are reusable blocks of code that perform specific tasks. Classes are blueprints for creating objects. They encapsulate data (attributes) and behavior (methods). Also, the concept of "Separation of Concerns," shows how functions act as the glue that helps to create modular and, maintainable code.

Functions

A function is a reusable block of code that performs a specific task or a set of tasks. Functions are a fundamental concept in most programming languages and serve several important purposes. The two we will focus on for now are:
Modularity: Functions allows us to break down a large program into smaller, more manageable pieces. Each function can focus on a specific task, making the code easier to understand, maintain, and debug.
Reusability: Once a function is defined, we can use its code multiple times throughout your program without having to rewrite the same code. This promotes code reuse and helps prevent redundancy.

Defining functions

Use the def keyword followed by the function name and parentheses. Inside the parentheses, specify any parameters the function accepts.
The function body is indented and contains the actual code.
The area after the function is defined is called the "main body" of the script. The statements within the function run later in your program by “calling” the function from the main body. The "pass" keyword in Python is a placeholder statement or a no operation. The primary purpose of the "pass" is to indicate a place for code not yet implemented.

Using parameters

Parameters are special variables used in a function to refer to the data provided as input when the function is called. They act as placeholders for the

actual values (called arguments) that the function will operate on. When defining a function, you specify its parameters inside the parentheses. Parameters serve as input channels through which data flows into the function. The function can then process this data based on the parameter names. Parameters act as local variables within the function that are only used with the function is running.

When we pass an immutable object (e.g., integers, strings, tuples) as an argument to a function in Python, we are not actually passing the object itself, but a reference to that object. However, these objects are immutable, meaning their values cannot be changed in place. If we reassign the parameter variable inside the function, it creates a new reference to a different object, but the original object remains unchanged outside the function.

When we pass a mutable object (e.g., lists, dictionaries, sets) as an argument to a function, we are also passing a reference to that object. However, since these objects are mutable, changes made to the object inside the function will affect the same object outside the function. This is because the reference to the object is the same both inside and outside the function.

Global vs Local variables

Variables declared within a function are considered local to that function. These variables can only be accessed and used within the function where they are defined. They are inside of the function's scope. Whereas variables declared outside of any function, typically in the main body of the script, are considered global to the entire script. These variables can be accessed and modified from anywhere within the script. They are in scope throughout the entire script.

Return values

A **return value** is the result that a function produces after performing its task. When a function completes its execution, it can send back a value to the caller. Return values allow functions to communicate information or computed results.

In Python, you use the return keyword to specify the value that a function should return. The value can be a variable, expression, or any data type. Once a function encounters a return statement, it immediately exits and provides the specified value.

Classes

Classes are fundamental in object-oriented programming (OOP). They allow us to create your own data types with custom attributes and methods. Think of a class as a blueprint for creating objects. To define a class, use the class keyword followed by the class name. Inside the class, define attributes (variables) and methods (functions).
Classes are a way of grouping functions, variables, and constants by the name of the class. Using classes to group functions is a fundamental concept in programming. Grouping functions within classes creates a modular structure, making it easier to manage and maintain code. Classes provide a natural way to organize code by grouping functions and data needed by those functions, making code more structured and readable, especially in larger projects.
using the @staticmethod decorator Python will allow us to use the class functions directly, instead of creating an object first.

Separation of Concerns

Separation of concerns (SoC) is a fundamental design principle in software development. It advocates breaking down a program into distinct sections, each addressing a specific concern or set of information. SoC ensures that different aspects of a program remain independent and well-organized. Each section focuses on a specific task, such as user interface, business logic, or data access. This pattern encourages modularity, reduces code complexity, and makes it easier to manage and extend software systems.

JSON Files

A JSON (JavaScript Object Notation) file is a text-based data format used to store and manipulate data. It stores information in key-value pairs and arrays. JSON is human-readable and widely used for data interchange. Data is organized in key-value pairs. Commas separate data elements. Curly braces {} hold objects, and square brackets [] hold arrays. JSON files simplify data storage and exchange, making them essential in modern software development.

```python
# ------------------------------------------------------------------------
# Title: Assignment06
# Desc: This assignment demonstrates how to use functions and class
# Change Log: (Who, When, What)
#   Abarna,2/18/2024, Created Script
# ------------------------------------------------------------------------
import json
import sys
from typing import IO
from json import JSONDecodeError

#creating a constant 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.json'
menu_choice: str = ""
students: list = []


class FileProcessor:
    """
    A class that performs file operations
    """"
    global FILE_NAME
    global students
    global file

# reading the data from json file
```

```python
# reading the data from json file
    @staticmethod
    def read_data_from_file(FILE_NAME , students) -> dict:
        """
        reads data from specified json file
        :param FILE_NAME
        :param students
        :return: list of students
        """
        try:
            with open(FILE_NAME, 'r') as file:
                students = json.load(file)
            for student in students:
                print(f'{student["first_name"]} {student["last_name"]} {student["course_name"]}')
        except FileNotFoundError as e:
            IO.output_error_message("JSON file not found", e)
            print("Creating a file")
            with open(FILE_NAME, 'w') as file:
                json.dump(file, students)
        except JSONDecodeError as e:
            IO.output_error_message("Invalid file", e)
            print("Creating a valid file...")
            with open(FILE_NAME, 'w') as file:
                json.dump(file, students)
        except Exception as e:
            IO.output_error_message("unhandled exception", e)
        return students


# writing to the json file
    @staticmethod
    def add_data_to_file(FILE_NAME , students) -> None:
        """
        writing data to the json file
        :param FILE_NAME
        :param students
        """
        try:
            student_first_name = input("Enter student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("First name must contain only alphabets")
            student_last_name = input("Enter student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("Last name must contain only alphabets")
            course_name = input("Enter course name: ")
            student_data = {"first_name" : student_first_name, "last_name" : student_last_name, "course_name" : course_name}
            students.append(student_data)
            with open(FILE_NAME, 'w') as file:
                json.dump(students, file)
        except ValueError as e:
            IO.output_error_message("User entered invalid name")
        except Exception as e:
            IO.output_error_message("There was an error writing to the file", e)
        return students

    @staticmethod
    def write_data_to_file(FILE_NAME, students):
        """
        writing data to the json file
        :param FILE_NAME
        :param students
        prints the data in the file
        """
        try:
            with open(FILE_NAME, 'w') as file:
                json.dump(students, file)
                for student in students:
                    print(f"You have registered {student['first_name']} {student['last_name']} for {student['course_name']}")
       except ValueError as e:
            IO.output_error_message("Unhandled excetion ",e)
        except Exception as e:
            IO.output_error_message("There was an error writing to the file", e)
```

```python
    @staticmethod
    def input_student_data(students):
        """
        getting the student data from user
        :param students
        """
        try:
            student_first_name = IO.get_input("Enter student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("First name must contain only alphabets")
            student_last_name = IO.get_input("Enter student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("Last name must contain only alphabets")
            course_name = IO.get_input("Enter course name: ")
            student_data = {"first_name" : student_first_name, "last_name" : student_last_name, "course_name" : course_name}
            students.append(student_data)
        except ValueError as e:
            IO.output_error_message("User entered invalid name")


    @staticmethod
    def output_student_courses(students):
        """
        displaying the student information
        :param students
        :return student information
        """
        try:
            for student in students:
                print(f"{student['first_name']} {student['last_name']} registered for {student['course_name']}")
        except Exception as e:
            IO.output_error_message("unhandled exception",e)
```

```python
# writing to the json file
    @staticmethod
    def add_data_to_file(FILE_NAME , students) -> None:
        """
        writing data to the json file
        :param FILE_NAME
        :param students
        """
        try:
            student_first_name = input("Enter student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("First name must contain only alphabets")
            student_last_name = input("Enter student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("Last name must contain only alphabets")
            course_name = input("Enter course name: ")
            student_data = {"first_name" : student_first_name, "last_name" : student_last_name, "course_name" : course_name}
            students.append(student_data)
            with open(FILE_NAME, 'w') as file:
                json.dump(students, file)
        except ValueError as e:
            IO.output_error_message("User entered invalid name")
        except Exception as e:
            IO.output_error_message("There was an error writing to the file", e)
        return students

    @staticmethod
    def write_data_to_file(FILE_NAME, students):
        """
        writing data to the json file
        :param FILE_NAME
        :param students
        prints the data in the file
        """
        try:
            with open(FILE_NAME, 'w') as file:
                json.dump(students, file)
                for student in students:
                    print(f"You have registered {student['first_name']} {student['last_name']} for {student['course_name']}")
        except ValueError as e:
            IO.output_error_message("Unhandled excetion ",e)
        except Exception as e:
            IO.output_error_message("There was an error writing to the file", e)
```

```python
class IO:
    """
    This class performs io operations
    """
    global students

    @staticmethod
    def output_error_message(message, error: Exception = None) -> None:
        """
        prints the error message and exception
        :param message:
        :param exception
        """
        print(message)
        if Exception is not None:
            print("__Technical Information__")
            print(error, error.__doc__)

    def input_menu_choice(MENU):
        """
        Getting the menu choice from user
        """
        print(MENU)
        str_choice = input("what would you like to do:")
        return str_choice
        while str_choice not in ["1", "2", "3", "4"]:
            print("Please enter the choice between 1 and 4")
            str_choice = input(MENU)
        return str_choice

    @staticmethod
    def get_input(message):
        return input(message)
```

```
# reads the existing data from json file
students = FileProcessor.read_data_from_file(FILE_NAME, students)
while True:
    choice = IO.input_menu_choice(MENU)
    # getting input from user
    if choice == "1":
        IO.input_student_data(students)

    # displaying student information
    elif choice == "2":
        IO.output_student_courses(students)

    # writing data to the file
    elif choice == "3":
        FileProcessor.write_data_to_file(FILE_NAME,students)

    # exit the program
    elif choice == "4":
        sys.exit()

    else:
        print("user entered invalid choice")
```

The output will be,

```
[{"first_name": "abarna", "last_name": "sharankumar",
"course_name": "python"}, {"first_name": "Max", "last_name":
"Goodwin", "course_name": "python"}, {"first_name": "Sarah",
"last_name": "Williams", "course_name": "python"}]
```

```
what would you like to do:3
You have registered abarna sharankumar for python
You have registered Max Goodwin for python

----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

=== RESTART: C:/Users/abarn/Documents/Python/Pythoncourse/A06/assignment06.py ==
abarna sharankumar python
Max Goodwin python
closing the file file

----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 course name: python
```

```
what would you like to do:2
abarna sharankumar registered for python
Max Goodwin registered for python
Sarah Williams registered for python

----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 python
You have registered Max Goodwin for python
You have registered Sarah Williams for python

----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
```

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

----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 course name: python

----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 registered for python
Max Goodwin registered for python
```

In PyCharm,

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

 ----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 course name: python
what would you like to do:2
abarna sharankumar registered for python
Max Goodwin registered for python

----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 python
You have registered Max Goodwin for python
closing the file
```

```
what would you like to do:1
Enter student's first name: sarah
Enter student's last name: Williams
Enter course name: python

----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

Process finished with exit code 0
```

```
C:\Users\abarn\Documents\PycharmProjects\Helloworld\pythonProject\venv\Scripts\python.exe
abarna sharankumar python
Max Goodwin python

----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: vic
Enter student's last name: vu
Enter course name: python
```

```
what would you like to do:2
abarna sharankumar registered for python
Max Goodwin registered for python
vic vu registered for python

----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 python
You have registered Max Goodwin for python
You have registered vic vu for python
closing the file
```

Summary

Thus, Python program is created for course registration 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.json” 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.