MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DO PIAUÍ

CENTRO DE CIÊNCIAS DA NATUREZA- CCE

Campus Universitário Ministro Petrônio Portella, Bloco 06 - Bairro Ininga,

CEP 64049-550 – Teresina-Piauí-Brasil – Fone: (86) 3215-5565 – Fax: (86) 3215-5560

Curso: Ciência da Computação

Disciplina: Laboratório de Programação

Professor: Armando Soares Sousa

Aluno: Otávio C. França

Relatório Referente a Segunda Lista

Introdução

O presente relatório descreve uma série de programas escritos em Python para resolver os problemas da segunda lista de exercícios referente a primeira avaliação. A linguagem utilizada é o "Python" que é uma linguagem de programação de alto nível conhecida por sua sintaxe simples e legibilidade, adequada para uma variedade de aplicações, desde desenvolvimento web até ciência de dados. Os códigos foram desenvolvidos utilizando a IDE "Visual Studio Code", que oferece uma gama de recursos robustos para o desenvolvimento em Python, incluindo realce de sintaxe, depuração e controle de versão.

Objetivos

Os objetivos dos códigos fornecidos são os seguintes:

- Programa 01: Criar um programa em Python que manipule uma lista de nomes de alunos de forma que exercite as operações de inserção, busca, ordenação e remoção de alunos dessa lista.
- Programa 02: Criar um programa em Python que manipule uma lista de tuplas de pontos GPS que represente a origem e destino de uma rota de forma que exercite as operações de inserção, remoção de pontos GPS dessa lista de tuplas.
- Programa 03: Criar um programa em Python que manipule três conjuntos de cores de forma que exercite as operações de união, intersecção, diferença entre dois conjuntos selecionados dentre os três conjuntos de cores.

Metodologia

A metodologia adotada para desenvolver os códigos apresentados neste relatório baseia-se em princípios de modularidade. Cada conjunto de funcionalidades está encapsulado em funções específicas, promovendo a reutilização do código e facilitando a manutenção. As funcionalidades relacionadas a cada parte do programa foram agrupadas em diferentes módulos, definidos por funções individuais. No sistema de gerenciamento de alunos, foram definidas funções separadas para adicionar, pesquisar, exibir e remover alunos da lista. O mesmo princípio foi aplicado aos sistemas de gerenciamento de pontos GPS e operações de conjuntos.

Reviw Code

Em todos os códigos, existem blocos if __name__ == "__main__": que definem o ponto de entrada do programa, onde a execução começa. Cada programa utiliza um loop while para fornecer um menu de opções ao usuário e executar as funções correspondentes com base na escolha do usuário.

1. Programa de Gerenciamento de Alunos

```python
def add_student():
  student_name = input("\nEnter the name of the student to be added: ")
  students.append(student_name)
  print(f"\nThe student {student_name} has been added to the list.\n")


def search_student():
  student_name = input("\nEnter the name of the student to search for: ")
  if student_name in students:
    print(f"\nThe student {student_name} is in the list.\n")
  else:
    print(f"\nThe student {student_name} is not in the list.\n")


def display_in_alphabetical_order():
  sorted_students = sorted(students)
  print("\nList of students in alphabetical order:\n")
  for student in sorted_students:
    print("-", student)


def remove_student():
  student_name = input("\nEnter the name of the student to be removed: ")
  if student_name in students:
    students.remove(student_name)
    print(f"\nThe student {student_name} has been removed from the list.\n")
  else:
    print(f"\nThe student {student_name} is not in the list.\n")
```

```python
if __name__ == "__main__":
    students = []
    option = 0
    while option != 5:
        print("\nOptions:")
        print("1 - Add student")
        print("2 - Search student")
        print("3 - Display list of students in alphabetical order")
        print("4 - Remove student")
        print("5 - Exit the program")

        option = int(input("Enter the number of the desired option: "))

        if option == 1:
            add_student()
        elif option == 2:
            search_student()
        elif option == 3:
            display_in_alphabetical_order()
        elif option == 4:
            remove_student()
        elif option == 5:
            print("Exiting the program...")
        else:
            print("\nInvalid option. Please choose a valid option.")
```

- add_student(): Esta função solicita ao usuário o nome de um aluno e o adiciona à lista de alunos.
- search_student(): Esta função permite ao usuário pesquisar se um determinado aluno está na lista de alunos.
- display_in_alphabetical_order(): Esta função exibe a lista de alunos em ordem alfabética.
- remove_student(): Esta função permite ao usuário remover um aluno da lista, se estiver presente.

2. Programa de Gerenciamento de Pontos GPS

```python
def add_gps_point():
    latitude = float(input("Enter the latitude of the GPS point: "))
    longitude = float(input("Enter the longitude of the GPS point: "))
    gps_point = (latitude, longitude)
    route.append(gps_point)
    print(f"\nThe GPS point {gps_point} has been added to the route.\n")

def remove_gps_point():
    if len(route) == 0:
        print("\nThe route is empty. There are no GPS points to remove.\n")
        return

    print("\nCurrent route:")
    display_route()

    index = int(input("\nEnter the index of the GPS point to remove: "))
    if index < 0 or index >= len(route):
        print("\nInvalid index. Please enter a valid index.\n")
        return

    removed_point = route.pop(index)
    print(f"\nThe GPS point {removed_point} has been removed from the route.\n")

def display_route():
    for i, gps_point in enumerate(route):
        print(f"{i}: {gps_point}")


if __name__ == "__main__":
    route = []
    option = 0
    while option != 4:
        print("\nOptions:")
        print("1 - Add GPS point to the route")
        print("2 - Remove GPS point from the route")
        print("3 - Display current route")
        print("4 - Exit the program")

        option = int(input("Enter the number of the desired option: "))

        if option == 1:
            add_gps_point()
        elif option == 2:
            remove_gps_point()
        elif option == 3:
            print("Current route:")
            display_route()
        elif option == 4:
            print("Exiting the program...")
        else:
            print("\nInvalid option. Please choose a valid option.")
```

- add_gps_point(): Esta função solicita ao usuário as coordenadas (latitude e longitude) de um ponto GPS e o adiciona à rota.

- remove_gps_point(): Esta função permite ao usuário remover um ponto GPS específico da rota com base no índice fornecido.

- display_route(): Esta função exibe a rota atual, mostrando todos os pontos GPS que foram adicionados.

3. Programa de Operações de Conjuntos:

```python

def perform_union(set1, set2):
    return set1.union(set2)

def perform_intersection(set1, set2):
    return set1.intersection(set2)

def perform_difference(set1, set2):
    return set1.difference(set2)

def display_sets():
    print("Set 1:", set1)
    print("Set 2:", set2)
    print("Set 3:", set3)

def display_result(operation, result):
    print(f"\nThe {operation} result is: {result}\n")

if __name__ == "__main__":
    set1 = {"red", "blue", "green", "yellow"}
    set2 = {"blue", "orange", "purple", "yellow"}
    set3 = {"green", "purple", "pink", "orange"}

    option = 0
    while option != 4:
        print("\nOptions:")
        print("1 - Perform union between two sets of colors")
        print("2 - Perform intersection between two sets of colors")
        print("3 - Perform difference between two sets of colors")
        print("4 - Exit the program")

        option = int(input("Enter the number of the desired option: "))

        if option == 1:
            display_sets()
            set_choice1 = int(input("Enter the number of the first set (1, 2, or 3): "))
            set_choice2 = int(input("Enter the number of the second set (1, 2, or 3): "))
            result = perform_union(globals()[f"set{set_choice1}"], globals()[f"set{set_choice2}"])
            display_result("union", result)
        elif option == 2:
            display_sets()
            set_choice1 = int(input("Enter the number of the first set (1, 2, or 3): "))
            set_choice2 = int(input("Enter the number of the second set (1, 2, or 3): "))
            result = perform_intersection(globals()[f"set{set_choice1}"], globals()[f"set{set_choice2}"])
            display_result("intersection", result)
        elif option == 3:
            display_sets()
            set_choice1 = int(input("Enter the number of the first set (1, 2, or 3): "))
            set_choice2 = int(input("Enter the number of the second set (1, 2, or 3): "))
            result = perform_difference(globals()[f"set{set_choice1}"], globals()[f"set{set_choice2}"])
            display_result("difference", result)
        elif option == 4:
            print("Exiting the program...")
        else:
            print("\nInvalid option. Please choose a valid option.")
```

- perform_union(set1, set2): Esta função retorna a união dos dois conjuntos passados como argumento.

- perform_intersection(set1, set2): Esta função retorna a interseção dos dois conjuntos passados como argumento.
- perform_difference(set1, set2): Esta função retorna a diferença entre o primeiro conjunto e o segundo conjunto passados como argumento.
- display_sets(): Esta função exibe os três conjuntos de cores predefinidos.
- display_result(operation, result): Esta função exibe o resultado de uma operação de conjunto (união, interseção ou diferença).

Discussões

Os referentes códigos estão disponíveis no Github: https://github.com/otaviofranca/programming-laboratory. E também no replit: https://replit.com/@otaviofranca?path=folder/fist_test_otavio_franca . **Gostaria, se possível, que avaliasse os programas e sugerisse possíveis melhorias.**