Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Dezvoltare Aplicatiilor Web”

Аттестация No.1

Проверил: Плешка Наталья

Выполнил: Чобану Артём

Кишинев 2021

# Задание:

# “Управление данными ингредиентов блюд"

Напишите код Python, который выведет пользователю список блюд (мин. - 3, макс. - 10) и потом позволит пользователю:

1. **увидеть в каких блюдах встречается какой-то ингредиент, который вводит пользователь** (проверьте чтобы вводились только буквы – мин. 3 и макс. 15). Программа выводит количество найденных блюд, их название, типа: "Данный ингредиент был найден в N блюдах: B1, B2, ..., BN" и все комбинации ингредиентов в которых был найден указанный пользователем ингредиент. Поиск производится в текстовом файле данных (*items.txt*), в которых были сохранены уже другие комбинации ингредиентов. В случае если при поиске не находится ингредиент, или если файл пуст - выведите пользователю соответствующее сообщение. Для поиска используйте метод *upper()* или *lower()*...;

2. **выбрать из предоставленного списка какое-то блюдо и для него предложить какую-то собственную комбинацию ингредиентов**. Для этого он должен ввести сперва количество ингредиентов и потом их названия. Проверьте, при помощи регулярных выражений, чтобы пользователь при первом вводе, вводил одну либо 2 цифры, а для названий ингредиентов - только буквы (от 3 до 15). Список введенных ингредиентов сохраните в файл *items.txt*. При вводе данных в файл используйте метод *upper()* или *lower()*.

# Использование коллекций:

* **Словари** удобны в случае необходимости пары ключ – значение. Словарь реализует интерфейс ассоциативного массива. В Python он работает именно как Хеш-таблица. Он не является упорядоченным.

Плюсы:

1. Не имеют понятия индекса. Вместо него – ключ, который может быть любого типа данных.
2. Изменяемые (mutable) – возможно удалять или добавлять элементы.

* **Кортеж** – упорядоченная неизменяемая коллекция. Необходимы там, где заранее известно, что элементы не будут изменяться. Это позволяет воспользоваться их преимуществом – высокая скорость работы.
* **Списки** – упорядоченная изменяемая коллекция. Необходима где нужна динамически изменяемая коллекция, для того, чтобы можно было удалять и добавлять элменты.

# Скриншоты работы программы

Изначальные данные, считанные из файла:

Text

Description automatically generated

Поиск по всем блюдам:

Text

Description automatically generated

Добавление нового ингредиента:

Text

Description automatically generated

Добавление ингредиента, который повторяется в разных блюдах:

Text

Description automatically generated

Валидация входных данных:

Название блюда:

Text

Description automatically generated

Название ингредиента:

Graphical user interface, text

Description automatically generated

Валидация вводимых данных:

Text

Description automatically generated

# Код программы:

from os.path import isfile  
  
  
def read\_dishes\_from\_file(file\_name):  
 dishes\_from\_file = dict()  
  
 if not isfile(file\_name):  
 dishes\_file = open(file\_name, "r+")  
 dishes\_file.close()  
 else:  
 dishes\_file = open(file\_name, "r")  
  
 file\_lines = dishes\_file.readlines()  
 for line in file\_lines:  
 words = line.split()  
  
 dish\_name = words.pop(0)  
 dish\_ingredients = []  
  
 for i in words:  
 dish\_ingredients.append(i)  
  
 dishes\_from\_file[dish\_name] = dish\_ingredients  
  
 dishes\_file.close()  
  
 return dishes\_from\_file  
  
  
def write\_dishes\_to\_file(file\_name, dishes\_to\_write):  
 dishes\_file = open(file\_name, "w+")  
  
 lines = []  
  
 for dish\_name in dishes\_to\_write.keys():  
 current\_dish = dishes\_to\_write[dish\_name]  
  
 line = dish\_name  
 for i in current\_dish:  
 line += f" {i}"  
 lines.append(line + "\n")  
  
 dishes\_file.writelines(lines)  
 dishes\_file.close()  
  
  
def input\_with\_length(min\_length, max\_length, return\_type):  
 while 1:  
 input\_string = input()  
  
 try:  
 input\_value = return\_type(input\_string)  
 except ValueError:  
 print(f"Please enter the value of type: {return\_type}.")  
 continue  
  
 input\_length = len(input\_string)  
 if min\_length <= input\_length <= max\_length:  
 return input\_value  
 else:  
 print(f"Please enter a value with length between {min\_length} and {max\_length} characters: ")  
  
  
dishes = read\_dishes\_from\_file("items.txt")  
  
print("Initial Dishes: ")  
for dish in dishes.keys():  
 ingredients = dishes[dish]  
 print(f"{dish} : {ingredients}")  
  
actionNumber = 0  
while actionNumber != 3:  
 print("\nChoose an option:")  
 print("1 - Find an ingredient in a dish")  
 print("2 - Add an ingredient")  
 print("3 - Exit")  
  
 actionNumber = input\_with\_length(1, 50, int)  
  
 if actionNumber == 1:  
 print("Enter the ingredient name: ")  
 ingredientName = input\_with\_length(3, 15, str)  
  
 foundDishes = dict()  
 for dish in dishes.keys():  
 ingredients = dishes[dish]  
  
 for ingredient in ingredients:  
 if ingredient.upper() == ingredientName.upper():  
 foundDishes[dish] = dishes[dish]  
  
 foundDishAmount = len(foundDishes)  
 if foundDishAmount > 0:  
 dishText = "dishes" if foundDishAmount > 1 else "dish"  
  
 print(f"The ingredient was found in {foundDishAmount} {dishText}:")  
 for dishName in foundDishes.keys():  
 print(f"{dishName}. Ingredients: {foundDishes[dishName]}")  
 else:  
 print("No dish with this ingredient was found")  
  
 elif actionNumber == 2:  
 print("Enter the dish to add the ingredient to:")  
 dishName = input\_with\_length(3, 10, str)  
 if dishName not in dishes.keys():  
 dishes[dishName] = []  
 print("Enter the ingredient to add to the entered dish:")  
 ingredientName = input\_with\_length(3, 15, str)  
 dishes[dishName].append(ingredientName)  
 elif actionNumber == 3:  
 print("Writing the dishes into the file...")  
 write\_dishes\_to\_file("items.txt", dishes)  
 print("Exiting...")  
 break  
 else:  
 print("Please select a correct option:")