НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №*6*

із дисципліни «Основи програмування»

на тему

«Списки, кортежі, множини»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-73 | *асистент Дрозденко О.М.* |
| *Садченко М.В.* |  |

Київ — 2017

ЗМІСТ

[ВИЗНАЧЕННЯ МЕТИ ТА ЗАВДАННЯ ЛАБОРАТОРНОЇ РОБОТИ 3](#_Toc497412197)

[1.1 Мета роботи 3](#_Toc497412198)

[1.2 Завдання на лабораторну роботу 3](#_Toc497412199)

[1.3 Вимоги до програми 3](#_Toc497412200)

[ОСНОВНА ЧАСТИНА 4](#_Toc497412201)

[2.1 Опис програми 4](#_Toc497412202)

[2.2 Блок-схема 6](#_Toc497412203)

[ВИСНОВОК 7](#_Toc497412204)

[ДОДАТОК (текст програми) 8](#_Toc497412205)

# ВИЗНАЧЕННЯ МЕТИ ТА ЗАВДАННЯ ЛАБОРАТОРНОЇ РОБОТИ

* 1. Мета роботи

Набуття навичок у роботі з мовою Python, завдяки опанування темою «Списки, кортежі, множини».

## Завдання на лабораторну роботу

*Варіант №25*

«Створити список, елементами якого є вкладені списки. Кількість елементів не менше 15. Вкладений список складається з двох елементів: країна та місто в цій країні. Необхідно:

* знайти міста, що знаходяться в Італії;
* знайти країну, міста якої найбільше вказані в списку.»

1.3 Вимоги до програми

Програма повинна забезпечувати наступні операції:

1) створення списку із вкладеними списками;

2) виділення міст Італії зі списку;

3) визначення найбільш зустрічаючої країни зі списку;

4) виведення результату.

# ОСНОВНА ЧАСТИНА

## 2.1 Опис програми

Програма починається з того, що користувачу пропонується обрати список, з яким буде працювати. Реалізовано за допомогою команди *input*. Це або список за замовчуванням (складається з 16 пар країн та міст), або ж створити свій шляхом вбивання назв реальних країн та міст. Заради перевірки на правильність назв було імпортовано модуль *geonamescache* та застосовано його внутрішні команди *.get\_cities\_by\_name(назва\_міста)* та *.get\_countries\_by\_names()*. Через те, що модуль потрібно окремо завантажувати, він може бути не у всіх, тому я за допомогою try-except зробив виключення, що якщо комп’ютер не має модуля – користувачеві висвітиться попередження про це.

Уся перевірка була виконана в функції *def create\_list*., де відбувається створення користувацького двомірного масиву та наповнення його. Зроблено методом виключення та рекурсії. При неправильному вводі користувачеві буде висвічено повідомлення з попередженням та повторний ввід пари. Кількість пар вводиться користувачем. Валідацію на натуральні числа та для вибору з-поміж двох чисел: 1 чи 2 - теж було створено у окремих функціях *def natural\_num(num)* та *def choice(char).*

Далі вже у потрібному двомірному масиві проводиться пошук міст Італії. Реалізовано за допомогою ітераторів та перебирання елементів по двойним індексам: знаходження у списку країни Італії, та додавання міста з цього списку у порожній.

Після цього, коли покажчик рядка перемістився до команди *print* – там визивається функція *def max\_count(pair\_list),* де *pair\_list* – потрібний масив (передається користувачем), для вирахування країни, яка налічує в двомірному масиві найбільшу кількість міст. Знову ж таки, реалізовано за допомогою ітераторів та циклів. Зроблено таким чином: спочатку додаються усі країни у порожній список. Список сортується заради більш зручного обчислення. Далі підраховується кількість зустрічей кожної країни. Функція повертає ту, яка міститься у списках найбільше.

Після обрахувань користувачеві виводяться (за допомогою *print* та методу, що перетворює список у рядок, *‘’.join()*) у двох рядках, відповідно, міста Італії та найбільш зутрічаюча країна (застосовано перенос рядка *\n*).

В кінці користувачеві пропонується 2 варіанти: вийти з програми чи залишитися для продовження обрахувань. Це було реалізовано за допомогою циклу *while* та команди *exit()*, що виконує фактичний вихід з програми. Для виходу необхідно ввести ‘1’,для того, щоб залишитись – ‘2’. Для валідації використано функцію *def choice().* Ще у програмі мною було застосовано імпортований модуль *time* та його метод *.sleep()* для зручності сприймання тексту. Ще мною була написана перевірка, щоб користувач не ввів менше 15 пар за умовою задачі. Але я це вважаю непотрібним через незручність та довгий час вводу.

## 2.2 Блок-схема

ТАК

2

НІ

Введення кількості пар

Число натуральне?

Введення пар (країна + місто)

ТАК

Реальні назви країни та міста?

НІ

Пошук найбільш зустрічаючої країни у списку

Вирахування міст Італії із масиву

ні 1, ні 2

ТАК

НІ

Виведення

результату

Продовжити?

1

1 (список за замовчуванням) чи 2 (користувацький список)?

Обрання списку

# ВИСНОВОК

Отримав навички в розробці алгоритму для програм на мові Python. Також опанував знаннями з теми: *«Списки, кортежі, множини»*. Ознайомився з тонкощами роботи з цими типами данних. Покращив свої навички роботи з текстом.

# ДОДАТОК (текст програми)

**try**:  
 **import** geonamescache  
 geo = geonamescache.GeonamesCache()  
**except** ImportError:  
 print(**"============================ Oh, no! =============================\n"  
 "====== Your computer do not support validation of geonames. ======\n"  
 "==================================================================\n"**)  
 geonamescache = **None  
import** time  
  
print(**'Вітаємо!\n'  
 'Програму виконав Садченко Микита, КМ-73.\n\n'  
 'Варіант 25.\n'  
 '"Створити список, елементами якого є вкладені списки. Кількість елементів не менше 15.\n'  
 'Вкладений список складається з двох елементів: країна та місто в цій країні.\n'  
 ' Необхідно:\n'  
 '- знайти міста, що знаходяться в Італії;\n'  
 '- знайти країну, міста якої найбільше вказані в списку."'**)  
city\_list = []  
user\_list = []  
time.sleep(2)  
  
  
error = **True  
while** error:  
 **def** natural\_num(num):  
 *""" NATURAL NUMBER VALIDATION """* **return** int(num) **if** num.isdigit() == 1 **and** int(num) != 0 **else** natural\_num(input(**"natural number!: "**))  
  
  
 **def** choice(char):  
 **return** int(char) **if** char == **'1' or** char == **'2' else** choice(input(**"'1' or '2': "**))  
  
  
 **def** create\_list(counter=1):  
 *""" CREATING of USER-LIST and CHECKING for REAL GEO-NAMES """* country = input(str(counter)+**".1. Enter name of country: "**)  
 **if** list(geo.get\_countries\_by\_names()).count(country) == 0:  
 print(**"Wrong name of country!"**)  
 time.sleep(1)  
 **return** create\_list(counter)  
 city = input(str(counter)+**".2. Enter name of city: "**)  
 **if** list(geo.get\_cities\_by\_name(city)):  
 user\_list.append([country, city])  
 **if** counter == amount:  
 **return** user\_list  
 **return** create\_list(counter + 1)  
 print(**"Wrong name of city!"**)  
 time.sleep(1)  
 **return** create\_list(counter)  
  
  
 **def** max\_count(pair\_list):  
 *""" DETERMINE COUNTRY THAT CONTAINS the BIGGEST NUMBER of CITIES """* country\_list = []  
 counter = 1  
 maxim = 0  
 max\_country = **"no!"  
 for** c1 **in** range(len(pair\_list)):  
 country\_list.append(pair\_list[c1][0])  
 sorted(country\_list)  
 **for** c2 **in** range(len(country\_list)):  
 **if** country\_list[c2] == country\_list[c2-1]:  
 counter += 1  
 **if** maxim < counter:  
 maxim = counter  
 max\_country = country\_list[c2-1]  
 **else**:  
 counter = 1  
 **return** max\_country  
  
  
 **if** choice(input(**"\nEnter '1' to work with default-list or '2' - to work with user-list: "**)) == 1:  
 default\_list = [[**"Ukraine"**, **"Kiev"**], [**"Uganda"**, **"Kampala"**], [**"Italy"**, **"Milano"**], [**"Ukraine"**, **"Kharkiv"**],  
 [**"Russia"**, **"Moscow"**], [**"Canada"**, **"Ottawa"**], [**"Poland"**, **"Warsaw"**], [**"Germany"**, **"Berlin"**],  
 [**"Poland"**, **"Lublin"**], [**"Italy"**, **"Rome"**], [**"Italy"**, **"Rome"**], [**"Spain"**, **"Madrid"**],  
 [**"Ukraine"**, **"Cherkasy"**], [**"Norway"**, **"Oslo"**], [**"Turkey"**, **"Ankara"**], [**"Ukraine"**, **"Khmelnytskyi"**]]  
 **for** i **in** range(len(default\_list)):  
 **if** default\_list[i][0] == **"Italy"**:  
 city\_list.append(default\_list[i][1])  
 time.sleep(1)  
 print(**"\nCities of Italy: "**+**', '**.join(set(city\_list)) +  
 **"\nThe most mentioned country: "** + max\_count(default\_list))  
 **else**:  
 amount = natural\_num(input(**"Enter amount of pairs: "**))  
 user\_list = create\_list()  
 **for** i **in** range(len(user\_list)):  
 **if** user\_list[i][0] == **"Italy"**:  
 city\_list.append(user\_list[i][1])  
 time.sleep(1)  
 print(**"\nCities of Italy: "**+**', '**.join(set(city\_list)) +  
 **"\nThe most mentioned country: "** + max\_count(user\_list))  
 time.sleep(2)  
 **if** choice(input(**"\nEnter '1' to exit, '2' - to stay: "**)) == 1:  
 error = **False** exit()  
  
**'''  
# Нижче зроблена перевірка, щоб користувач ввів не менше 15 пар.  
# Але це я вважаю непотрібним - тому закоментовано.  
  
def natural\_num(num):  
 """ VALIDATION for NATURAL NUMBER and AT LEAST 15 - NUMBER """  
 if num.isdigit() == 1 and int(num) != 0:  
 if int(num) < 15:  
 natural\_num(input("at least 15!:"))  
 return int(num)  
 else:  
 natural\_num(input("natural number!: "))  
'''**