ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра програмування

ЗВІТ

про проходження виробничої (обчислювальної) практики

з дисципліни "Програмування"

(Частина 1 - Python)

Студентка групи ПМО-21 Кравець Ольга

Керівник практики: доц. Селіверстов Р. Г.

Львів - 2021

Завдання 1

Дані про туристичні маршрути (не менше 10 записів) містяться у файлі у такому форматі:

старт, фініш, довжина_ділянки_1, довжина_ділянки_2, ...

Наприклад, запис

Кваси, Дземброня, 8, 8, 12, 18, 12

означає маршрут Кваси-Дземброня загальної довжини 8+8+12+18+12=58 км з 4-ма привалами (5-ма переходами).

Створити клас "Маршрут" з необхідними властивостями (початкова точка, кінцева точка, перелік довжин переходів) та методами (конструктор, метод __str__, отримання довжини маршруту, кількості привалів, перевантаження операторів порівняння). Створити список маршрутів. Упорядкувати його за протяжністю. Написати функції для виведення маршрутів з максимальною кількістю привалів, з найдовшим переходом, з початком/кінцем у певній точці.

#Файл "PMO21_lab1_var9.py"

```
from route_func import *
file_name = 'routes_data.txt'
with open(file_name, encoding = 'utf-8') as datafile:
    routes_list = []
    for r in datafile:
        d_list = []
        for i in range(2, len(r.split(', '))):
            d_list.append(int(r.split(', ')[i]))
        routes_list.append(Route(r.split(', ')[0], r.split(', ')[1], d_list))
print("\nМаршрути:")
print_on(routes_list)
print("\nМаршрути за протяжністю:")
routes_list.sort()
print_on(routes_list)
print("\nМаршрути з максимальною кількістю привалів:")
```

```
print_on(routes_max_stop(routes_list))
print("\nМаршрути з найдовшим переходом:")
print_on(longest_distance(routes_list))
print("\nМаршрути з початком у Чорногорі:")
print_on(start_in_certain_point('Чорногора', routes_list))
print("\nМаршрути з кінцем у Дземброні:")
print_on(end_in_certain_point('Дземброня', routes_list))
```

#Файл “route_func.py”

```
class Route:
    def __init__(self, start, end, distances):
        self.start = start
        self.end = end
        self.distances = distances
    def __str__(self) -> str:
        return f"{self.start} - {self.end} "\
                f"загальної довжини {sum(self.distances)} км "\
                f"з {len(self.distances)-1}-ма привалами"\
                f"({len(self.distances)}-ма переходами)"
    def get_route_distance(self): # отримання довжини маршруту
        return sum(self.distances)
    def get_stop_number(self): # кількість привалів
        return len(self.distances) - 1
    def __gt__(self, other):
        return self.get_stop_number() > other.get_stop_number()
    def get_max_stop(self): # отримання найдовшої дистанції
        return max(self.distances)
def print_on(routes_list):
    for r in routes_list:
        print(r)
def routes_max_stop(routes_list): # максимальна к-сть привалів
```

```
    routes_l = []
    for r in routes_list:
        if r.get_stop_number() ==  max(routes_list).get_stop_number():
            routes_l.append(r)
    return routes_l
def longest_distance(routes_list): # маршрут з найдовшим переходом
    l_routes = []
    for r in routes_list:
        if r.get_max_stop() == max(routes_list,\
            key=lambda x: x.get_max_stop()).get_max_stop():
            l_routes.append(r)
    return l_routes
def start_in_certain_point(point, routes_list): #з початком у певній точці
    start_l = []
    for r in routes_list:
        if r.start == point:
            start_l.append(r)
    return start_l
def end_in_certain_point(point, routes_list): #з кінцем у певній точці
    end_l = []
    for r in routes_list:
        if r.end == point:
            end_l.append(r)
    return end_l
```

#Файл “routes_data.txt”

Кваси, Дземброня, 8, 8, 12, 18, 12

Менчул, Верхнє, 8, 2, 5, 12, 5

Гутин, Дземброня, 18, 9, 14, 7, 6

Данцер, Копиця, 14, 17, 10, 4, 10

Шешул, Ребра, 6, 12, 8, 9, 5, 7

Чорногора, Туркул, 13, 17, 6, 5, 8, 12

Смотрич, Кваси, 3, 7, 18, 14, 10

Копиця, Дземброня, 15, 6

Петросул, Туркул, 14, 16, 20

Верхнє, Дземброня, 5, 9, 11, 14, 17

Чорногора, Шешул, 4, 8, 7, 4

```
Маршрути:
Кваси - Дземброня загальної довжини 58 км з 4-ма привалами(5-ма переходами)
Менчул - Верхнє загальної довжини 32 км з 4-ма привалами(5-ма переходами)
Гутин - Дземброня загальної довжини 54 км з 4-ма привалами(5-ма переходами)
Данцер - Копиця загальної довжини 55 км з 4-ма привалами(5-ма переходами)
Шешул - Ребра загальної довжини 47 км з 5-ма привалами(6-ма переходами)
Чорногора - Туркул загальної довжини 61 км з 5-ма привалами(6-ма переходами)
Смотрич - Кваси загальної довжини 52 км з 4-ма привалами(5-ма переходами)
Копиця - Дземброня загальної довжини 21 км з 1-ма привалами(2-ма переходами)
Петросул - Туркул загальної довжини 50 км з 2-ма привалами(3-ма переходами)
Верхнє - Дземброня загальної довжини 56 км з 4-ма привалами(5-ма переходами)
Чорногора - Шешул загальної довжини 23 км з 3-ма привалами(4-ма переходами)

Маршрути за протяжністю:
Копиця - Дземброня загальної довжини 21 км з 1-ма привалами(2-ма переходами)
Петросул - Туркул загальної довжини 50 км з 2-ма привалами(3-ма переходами)
Чорногора - Шешул загальної довжини 23 км з 3-ма привалами(4-ма переходами)
Кваси - Дземброня загальної довжини 58 км з 4-ма привалами(5-ма переходами)
Менчул - Верхнє загальної довжини 32 км з 4-ма привалами(5-ма переходами)
Гутин - Дземброня загальної довжини 54 км з 4-ма привалами(5-ма переходами)
Данцер - Копиця загальної довжини 55 км з 4-ма привалами(5-ма переходами)
Смотрич - Кваси загальної довжини 52 км з 4-ма привалами(5-ма переходами)
Верхнє - Дземброня загальної довжини 56 км з 4-ма привалами(5-ма переходами)
Шешул - Ребра загальної довжини 47 км з 5-ма привалами(6-ма переходами)
Чорногора - Туркул загальної довжини 61 км з 5-ма привалами(6-ма переходами)

Маршрути з максимальною кількістю привалів:
Шешул - Ребра загальної довжини 47 км з 5-ма привалами(6-ма переходами)
Чорногора - Туркул загальної довжини 61 км з 5-ма привалами(6-ма переходами)

Маршрути з найдовшим переходом:
Петросул - Туркул загальної довжини 50 км з 2-ма привалами(3-ма переходами)

Маршрути з початком у Чорногорі:
Чорногора - Шешул загальної довжини 23 км з 3-ма привалами(4-ма переходами)
Чорногора - Туркул загальної довжини 61 км з 5-ма привалами(6-ма переходами)

Маршрути з кінцем у Дземброні:
Копиця - Дземброня загальної довжини 21 км з 1-ма привалами(2-ма переходами)
Кваси - Дземброня загальної довжини 58 км з 4-ма привалами(5-ма переходами)
Гутин - Дземброня загальної довжини 54 км з 4-ма привалами(5-ма переходами)
Верхнє - Дземброня загальної довжини 56 км з 4-ма привалами(5-ма переходами)
```

Завдання 2

Використовуючи бібліотеки urllib та beautifulsoap, отримайте дані (ціна пального, курс валюти, кількість товару тощо) з принаймі трьох різних сайтів. Визначте середній час виконання запитів, здійснивши процедуру 5 разів. Після цього реалізуйте отримання інформації з кожного сайту в окремому потоці (використайте можливості модуля threading) і визначте, наскільки зменшився середній час виконання запитів.

```
from urllib.request import urlopen
from bs4 import BeautifulSoup as bs
import time
import threading

def get_price(siteUrl, f_all, fin):
  html = urlopen(siteUrl)
  obj = bs(html.read(), features='html.parser')
  all_divs = obj.find_all('div', {'class':f_all})
  for i in all_divs:
    price = i.find('span', {'class':fin}).get_text()
    print ('Ціна зі сайту:', price)

start_time = time.time()
for i in range(5):
  get_price('https://nashformat.ua/products/faktor-cherchyllya-yak-odna-lyudyna-zminyla-istoriyu-912095', 'product-price', 'fn-price')

  get_price('https://dumka.top/nehudozhnya-literatura/biografiyi-j-memuary/polityky-istorichni-diyachy/faktor-cherchyllya-yak-odna-lyudyna-zminyla-istoriyu-dzhonson-boris-9789669427960', 'price-cart', 'integer')

  get_price('https://book-ye.com.ua/catalog/biohrafiyi-vidomykh-lyudej/faktor-cherchyllya-yak-odna-lyudyna-zminyla-istoriyu/?gclid=Cj0KCQiA-qGNBhD3ARIsAO_o7yk0nXJTP_MYymuzRmExivezZRj3VcqvPPSYJpWYZe28zUxZC1u2Lu0aAticEALw_wcB', 'card__price', 'card_price-current-real')
```

```
        get_price('https://book24.ua/product/faktor-cherchillya-yak-odna-lyudina-zminila-istoriyu/?gclid=Cj0KCQiAnaeNBhCUARIsABEee8U1Ep8HkdHb6MV65Xu1kV4-wU0EUU-jzZkEqxx8H8tlwUcFELozQeYaAmloEALw_wcB', 'price font-bold font_mxs', 'price_value')

  print('---------------------------------')


end_time = time.time() - start_time

average_time = end_time/5

print("Час виконання усіх запитів 5 разів:", end_time)

print("\nСередній час виконання запиту:", average_time)

print('---------------------------------')


start_time2 = time.time()

th1 = threading.Thread(target=get_price, args=('https://nashformat.ua/products/faktor-cherchyllya-yak-odna-lyudyna-zminyla-istoriyu-912095', 'product-price', 'fn-price'))

th2 = threading.Thread(target=get_price, args=('https://dumka.top/nehudozhnya-literatura/biografiyi-j-memuary/polityky-istorichni-diyachy/faktor-cherchyllya-yak-odna-lyudyna-zminyla-istoriyu-dzhonson-boris-9789669427960', 'price-cart', 'integer'))

th3 = threading.Thread(target=get_price, args=('https://book-ye.com.ua/catalog/biohrafiyi-vidomykh-lyudej/faktor-cherchyllya-yak-odna-lyudyna-zminyla-istoriyu/?gclid=Cj0KCQiA-qGNBhD3ARIsAO_o7yk0nXJTP_MYymuzRmExivezZRj3VcqvPPSYJpWYZe28zUxZC1u2Lu0aAticEALw_wcB', 'card__price', 'card_price-current-real'))

th4 = threading.Thread(target=get_price, args=('https://book24.ua/product/faktor-cherchillya-yak-odna-lyudina-zminila-istoriyu/?gclid=Cj0KCQiAnaeNBhCUARIsABEee8U1Ep8HkdHb6MV65Xu1kV4-wU0EUU-jzZkEqxx8H8tlwUcFELozQeYaAmloEALw_wcB', 'price font-bold font_mxs', 'price_value'))

th1.start()

th2.start()

th3.start()

th4.start()

end_time2 = time.time() - start_time2

print("Час виконання запитів (threading):", end_time2)

print('---------------------------------')

print("Середній час виконання запитів зменшився на:", average_time - end_time2)
```

```
Ціна зі сайту: 230
Ціна зі сайту: 207
Ціна зі сайту: 221
Ціна зі сайту: 230
---------------------------------
Ціна зі сайту: 230
Ціна зі сайту: 207
Ціна зі сайту: 221
Ціна зі сайту: 230
---------------------------------
Ціна зі сайту: 230
Ціна зі сайту: 207
Ціна зі сайту: 221
Ціна зі сайту: 230
---------------------------------
Ціна зі сайту: 230
Ціна зі сайту: 207
Ціна зі сайту: 221
Ціна зі сайту: 230
---------------------------------
Ціна зі сайту: 230
Ціна зі сайту: 207
Ціна зі сайту: 221
Ціна зі сайту: 230
---------------------------------
```

```
Час виконання усіх запитів 5 разів: 66.829509973526

Середній час виконання запиту: 13.365901994705201
---------------------------------
Час виконання запитів (threading): 0.00751948356628418
---------------------------------
Середній час виконання запитів зменшився на: 13.358382511138917
Ціна зі сайту: 230
Ціна зі сайту: 221
Ціна зі сайту: 230
Ціна зі сайту: 207
```

Завдання 3

Реалізувати гру за такими правилами:

Двом гравцям роздається по 5 карток з цифрами від 1 до 9, які генеруються випадковим чином (можуть повторюватися). Гравці почергово викладають свої картки на стіл (картки викладаються з кінця списку). Якщо викладена гравцем картка співпадає з верхньою картою на столі, то вона передається іншому гравцеві (на початок списку його карт). Умова припинення гри обирається згідно з варіантом (якщо один з гравців виклав останню картку - гра теж припиняється). Якщо по завершенню гри на столі парна кількість очок - перемагає перший гравець, непарна - другий.

Умови припинення гри: на столі три підряд картки з непарними числами

На екран виводяться початкова ситуація (картки учасників), хід гри та переможець.

#Файл "PMO21_lab3_var9.py"

```
from player_func import *

from game_func import *

player_1 = Player()

player_1.create_card()

player_2 = Player()

player_2.create_card()

game = Game(player_1, player_2)

game.start()
```

#Файл "game_func.py"

```
from player_func import *

class Game:

    def __init__(self, player1, player2, table = []):

        self.player1 = player1

        self.player2 = player2

        self.table                              =                              table


    def print_on(self):
```

```
    print(f"STATE: Player 1: {self.player1.get_cards()}"
        f"--- Player 2: {self.player2.get_cards()}"
        f" --- Table: {self.table}")

def table_upper_identical(self, player):
    if len(self.table) and len(player.get_cards()):
        t = self.table[len(self.table)-1]
        return player.put_card() == t
    return                                                          False

def start(self):
    self.print_on()
    while len(self.player1.get_cards()) and len(self.player2.get_cards()):
        print(f"Player 1 put: {self.player1.put_card()}")
        if self.table_upper_identical(self.player1):
            self.player2.take_card(self.player1.put_card())
            print(f"Player 2 take: ", self.player1.cards.pop())
        else:
            self.table.append(self.player1.cards.pop())
        self.print_on()
        print(f"Player 2 put: {self.player2.put_card()}")
        if self.table_upper_identical(self.player2):
            self.player1.take_card(self.player2.put_card())
            print(f"Player 1 take: ", self.player2.cards.pop())
        else:
            self.table.append(self.player2.cards.pop())
        self.print_on()
    for i in range(len(self.table)-3):
        if self.table[i]%2 !=0 and self.table[i+1]%2 !=0\
             and self.table[i+2]%2 !=0:
            print ("GAME END!")
        break
```

```
        if sum(self.table)%2 != 0:
             print("Winer: Player 2")
        else:
            print("Winer: Player 1")
```

#Файл “player_func.py”

```
from random import randint as rn
class Player:
    def __init__(self, cards = []):
        self.cards = cards
    def create_card(self):
        self.cards = [rn(1, 9) for i in range(5)]
    def get_cards(self):
        return self.cards
    def put_card(self):
        return self.cards[len(self.cards)-1]
    def take_card(self, card):
        self.cards.insert(0, card)
```

```
STATE: Player 1: [8, 1, 6, 9, 7] --- Player 2: [1, 5, 5, 7, 9] --- Table: []
Player 1 put: 7
STATE: Player 1: [8, 1, 6, 9] --- Player 2: [1, 5, 5, 7, 9] --- Table: [7]
Player 2 put: 9
STATE: Player 1: [8, 1, 6, 9] --- Player 2: [1, 5, 5, 7] --- Table: [7, 9]
Player 1 put: 9
Player 2 take:  9
STATE: Player 1: [8, 1, 6] --- Player 2: [9, 1, 5, 5, 7] --- Table: [7, 9]
Player 2 put: 7
STATE: Player 1: [8, 1, 6] --- Player 2: [9, 1, 5, 5] --- Table: [7, 9, 7]
Player 1 put: 6
STATE: Player 1: [8, 1] --- Player 2: [9, 1, 5, 5] --- Table: [7, 9, 7, 6]
Player 2 put: 5
STATE: Player 1: [8, 1] --- Player 2: [9, 1, 5] --- Table: [7, 9, 7, 6, 5]
Player 1 put: 1
STATE: Player 1: [8] --- Player 2: [9, 1, 5] --- Table: [7, 9, 7, 6, 5, 1]
Player 2 put: 5
STATE: Player 1: [8] --- Player 2: [9, 1] --- Table: [7, 9, 7, 6, 5, 1, 5]
Player 1 put: 8
STATE: Player 1: [] --- Player 2: [9, 1] --- Table: [7, 9, 7, 6, 5, 1, 5, 8]
Player 2 put: 1
STATE: Player 1: [] --- Player 2: [9] --- Table: [7, 9, 7, 6, 5, 1, 5, 8, 1]
GAME END!
Winer: Player 2
```

```
STATE: Player 1: [3, 8, 5, 6, 4]--- Player 2: [3, 8, 2, 3, 6] --- Table: []
Player 1 put: 4
STATE: Player 1: [3, 8, 5, 6]--- Player 2: [3, 8, 2, 3, 6] --- Table: [4]
Player 2 put: 6
STATE: Player 1: [3, 8, 5, 6]--- Player 2: [3, 8, 2, 3] --- Table: [4, 6]
Player 1 put: 6
Player 2 take:  6
STATE: Player 1: [3, 8, 5]--- Player 2: [6, 3, 8, 2, 3] --- Table: [4, 6]
Player 2 put: 3
STATE: Player 1: [3, 8, 5]--- Player 2: [6, 3, 8, 2] --- Table: [4, 6, 3]
Player 1 put: 5
STATE: Player 1: [3, 8]--- Player 2: [6, 3, 8, 2] --- Table: [4, 6, 3, 5]
Player 2 put: 2
STATE: Player 1: [3, 8]--- Player 2: [6, 3, 8] --- Table: [4, 6, 3, 5, 2]
Player 1 put: 8
STATE: Player 1: [3]--- Player 2: [6, 3, 8] --- Table: [4, 6, 3, 5, 2, 8]
Player 2 put: 8
Player 1 take:  8
STATE: Player 1: [8, 3]--- Player 2: [6, 3] --- Table: [4, 6, 3, 5, 2, 8]
Player 1 put: 3
STATE: Player 1: [8]--- Player 2: [6, 3] --- Table: [4, 6, 3, 5, 2, 8, 3]
Player 2 put: 3
Player 1 take:  3
STATE: Player 1: [3, 8]--- Player 2: [6] --- Table: [4, 6, 3, 5, 2, 8, 3]
Player 1 put: 8
STATE: Player 1: [3]--- Player 2: [6] --- Table: [4, 6, 3, 5, 2, 8, 3, 8]
Player 2 put: 6
STATE: Player 1: [3]--- Player 2: [] --- Table: [4, 6, 3, 5, 2, 8, 3, 8, 6]
Winer: Player 2
```

Завдання 4

Реалізувати у вигляді функції алгоритм пошуку чи сортування змішуванням (cocktail sort) та продемонструвати його на прикладі конкретних списків. Для алгоритмів сортування передбачити аргументом функції вибір напрямку впорядкування (за зростанням чи спаданням).

#Файл "PMO21_lab4_var9.py"

```
from functions import *

list_1 = create_list()

print("\nПочатковий список: ")

n = len(list_1)

for i in range(n):

   print (list_1[i],end = " ")

direction_sort(list_1)

print("\nВідсортований список:")

for i in range(n):

   print (list_1[i],end = " ")

print('')
```

#Файл "functions.py"

```
from random import randint

def create_list():

    lis = []

    while len(lis) < 10:

        r = randint(0, 50)

        if r not in lis:

            lis.append(r)

    return lis

def cocktail_sort(elem):

    n = len(elem)

    is_swapped = True

    begin = 0
```

```
    end = n-1
    while is_swapped:
        print(elem)
        is_swapped = False
        for i in range (begin, end):
            if (elem[i] > elem[i+1]) :
                temp = elem[i]
                elem[i] = elem[i+1]
                elem[i+1] = temp
                print(elem)
                is_swapped=True
        if not(is_swapped):
            break
        print(elem)
        is_swapped = False
        end = end-1
        for i in range(end-1, begin-1,-1):
           if elem[i] > elem[i + 1]:
                temp = elem[i]
                elem[i] = elem[i+1]
                elem[i+1] = temp
                print(elem)
                is_swapped = True
        begin = begin+1
def direction_sort(elem):
    count = int(input("\nВведіть: 1 - зростання | 2 - спадання: "))
    if count == 1:
        cocktail_sort(elem)
    elif count == 2:
        cocktail_sort(elem)
        elem.sort(reverse = True)
```

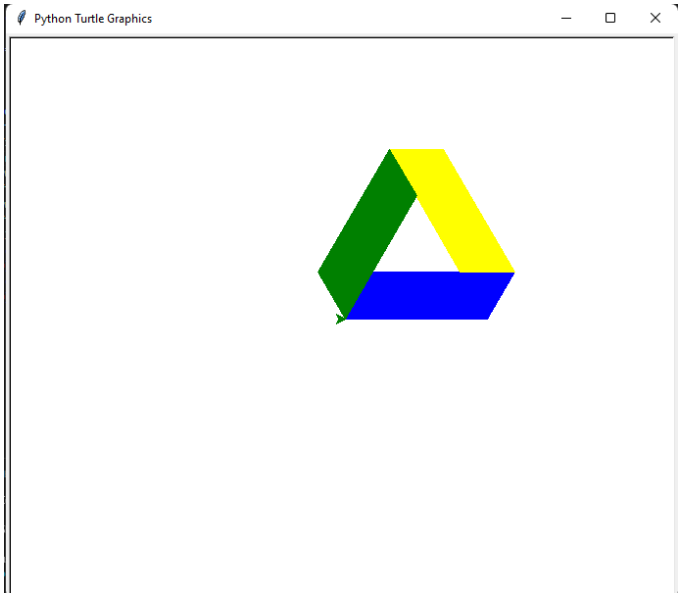
```
    else:
        print("\nНекоректно введені дані")
```

```
Початковий список:
-4 -7 -9 -2 3 -10
Введіть: 1 - зростання | 2 - спадання: 2
[-4, -7, -9, -2, 3, -10]
[-7, -4, -9, -2, 3, -10]
[-7, -9, -4, -2, 3, -10]
[-7, -9, -4, -2, -10, 3]
[-7, -9, -4, -2, -10, 3]
[-7, -9, -4, -10, -2, 3]
[-7, -9, -10, -4, -2, 3]
[-7, -10, -9, -4, -2, 3]
[-10, -7, -9, -4, -2, 3]
[-10, -7, -9, -4, -2, 3]
[-10, -9, -7, -4, -2, 3]
[-10, -9, -7, -4, -2, 3]

Відсортований список:
3 -2 -4 -7 -9 -10
```

Завдання 5

Використовуючи модуль turtle, написати програму для створення рисунка. Для рисування однакових елементів використати цикли та функції.

```
import turtle

t = turtle.Turtle()

def paral(colors = []):
    a= 145
    b = 55
    alpha = 60
    for item in colors:
        t.color(item)
        t.begin_fill()
        for i in range(2):
            t.forward(a)
            t.left(alpha)
            t.forward(b)
            t.left(180 - alpha)
        t.end_fill()
        t.forward(145)
        t.left(60)
        t.forward(55)
        t.left(60)

colors = ["blue", "yellow", "green"]
paral(colors)
input("")
```


Python Turtle Graphics

Завдання 6

Файл taxes.csv містить дані про тарифи на дзвінки (за 1 хвилину розмови) на мобільні телефони різних операторів та стаціонарні телефони. У файлі staff.csv містяться дані про співробітників фірми (їх прізвища та посади), а у файлах calls_1.csv, calls_2.csv, ... — дані про дзвінки (хто дзвонив, куди дзвонив, тривалість розмови).Використовуючи бібліотеки pandas, matplotlib та ipywidgets:

1. знайти, на яку суму здійснено дзвінків співробітниками певної посади (обирається з випадаючого списку);

2. побудувати гістограму загальної тривалості кожного типу дзвінків;

3. вивести найбільш балакучого співробітника та дані про усі його розмови.

 # PMO21_lab10_var9.ipynb

```
import pandas as pd
from ipywidgets import widgets
import matplotlib.pyplot as plt
%matplotlib inline

df_prices = pd.read_csv("taxes.csv")
df_prices.set_index("Tariff")
df_prices

df_st = pd.read_csv("staff.csv")
df_st = pd.DataFrame(df_st)
df_st

df_cal1 = pd.read_csv("calls_1.csv")
df_cal1 = pd.DataFrame(df_cal1)
df_cal1
```

```
df_cal2 = pd.read_csv("calls_2.csv")

df_cal2 = pd.DataFrame(df_cal2)

df_cal2


df_cal3 = pd.read_csv("calls_3.csv")

df_cal3 = pd.DataFrame(df_cal3)

df_cal3


df_calls = pd.concat(map(pd.read_csv, ['calls_1.csv',
'calls_2.csv','calls_3.csv']))

df_calls.set_index("Surname")

df_calls


def get_sum(index, table):

    for i in range(len(table)):

        if table.iloc[i, 0] == index:

            return table.iloc[i, 1]

df_calls["Sum"] = df_calls.apply(lambda x: x.Duration * get_sum(x.Tariff,
df_prices), axis = 1)

df_calls.set_index("Tariff")


df_calls = df_calls.merge(df_st)

df_calls


s = df_calls.groupby('Position')['Sum'].sum().reset_index(name='Sum of calls')

S


def drop_down_list(position):

    if position == 'assistant':

        print(s.iloc[0:1])

    elif position == 'director':
```

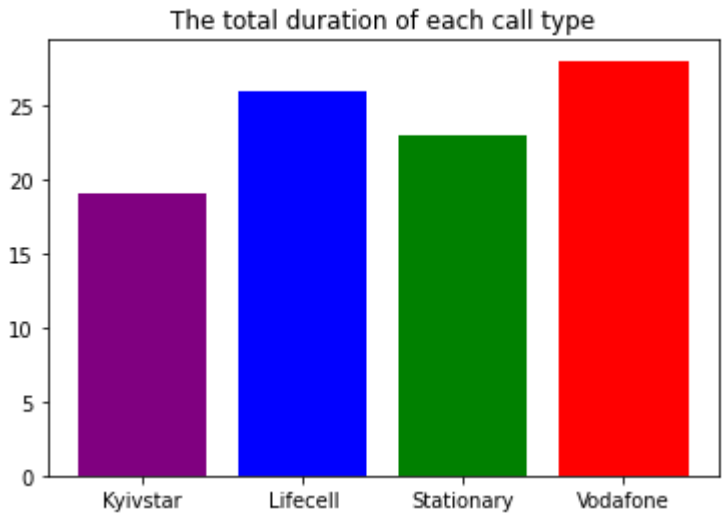
```
        print(s.iloc[1:2])
    elif position == 'operator':
        print(s.iloc[2:3])
    else:
        print('error!')


w = widgets.interactive(drop_down_list, position=['assistant', 'director',
'operator'])
W


total_talk =
df_calls.groupby('Tariff')['Duration'].sum().reset_index(name='Duration')
total_talk


def histogram():
    fig, ax = plt.subplots()
    ax.bar(total_talk['Tariff'][-4:], total_talk['Duration'][-4:],
color=['purple', 'blue', 'green', 'red'])
    ax.set_title("The total duration of each call type")
    plt.xticks(rotation=360)
    plt.show()


histogram()


sort_all_talk =
df_calls.groupby('Surname')['Duration'].sum().reset_index(name='Duration')
sort_all_talk = sort_all_talk.sort_values(by=['Duration'])[-4:]
print("The most talkative:")
sort_all_talk.tail(1)
```

position director

```
   Position  Sum of calls
1  director          9.01
```

position operator

```
   Position  Sum of calls
2  operator          15.0
```



The most talkative:

	Surname	Duration
2	Kravets	27

Завдання 7

Використовуючи модуль unittest, написати пакет тестів для тестування двох функцій. Передбачити валідацію аргументів інструкціями обробки винятків. Функції для тестування:

Функція отримує аргументом рік у вигляді цілого числа та повертає кількість днів у ньому.

Функція отримує аргументом список цілих чисел, викидає з нього парні і повертає кількість викинутих чисел.

#Файл "functions.py"

```
import calendar

#16.Функція отримує аргументом список цілих чисел, викидає з нього парні і повертає кількість викинутих чисел.

def count_even(myList):

    count = 0

    for item in myList:

        if int(item) % 2 == 0:

            count += 1

    return count

#8.Функція отримує аргументом рік у вигляді цілого числа та повертає кількість днів у ньому.

def days_in_year(year):

    days = 0

    if year > 0:

        if calendar.isleap(year):

            days = 366

        else:

            days = 365

    return days
```

#Файл "PMO21_lab7_var9.py"

```
from functions import *

from random import randint
```

```
list_1 = [randint(1, 100) for i in range(8)]
print("Список: ", list_1)
print("Кількість викинутих парних чисел:",  count_even(list_1))
print("---------------------------")
print("Кількість днів у 2021 році: ", days_in_year(2021))
print("Кількість днів у 2020 році: ",days_in_year(2020))
```

#Файл “test_runner.py”

```
from functions import *
import unittest
class FuncsTest(unittest.TestCase):
        def test_count_even(self):
                self.assertEqual(count_even([3, 6, 5, 4, 2]), 3)
                self.assertEqual(count_even([5, 3]), 0)
                self.assertEqual(count_even([0, -2, -4, -6, -3]), 4)
                self.assertEqual(count_even([0, 2, -4]), 3)
                self.assertNotEqual(count_even([0, 2, -4]), 2)

        def test_days_in_yearn(self):
                self.assertEqual(days_in_year(2020), 366)
                self.assertEqual(days_in_year(2022), 365)
                self.assertEqual(days_in_year(-2), 0)
                self.assertEqual(days_in_year(0), 0)
                self.assertNotEqual(days_in_year(2022), 366)
                self.assertRaises(TypeError, days_in_year, -2, 365)
if __name__ == '__main__':
```

```
Ran 2 tests in 0.001s

OK
```

Завдання 8

Розробити графічний інтерфейс користувача (GUI) для програми з будь-якого іншого завдання практики (номер завдання вказати у коментарі на початку програми). Графічний інтерфейс повинен містити і коректно (за призначенням) використовувати такі віджети:

рамка (Frame);

напис (Label);

кнопка (Button);

прапорець (Checkbutton);

група перемикачів (Radiobutton);

поле для введення даних (Entry);

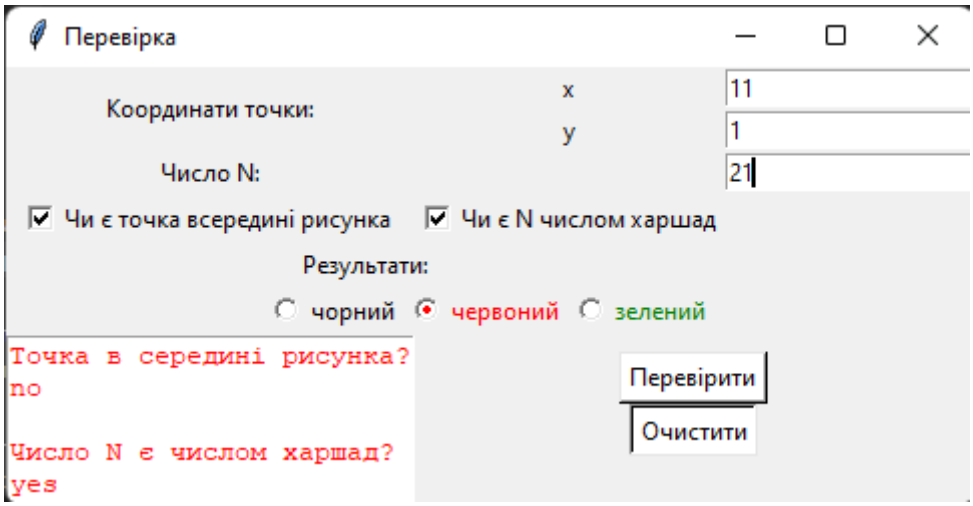
текстове поле для виведення результату (Text).

```
# для ЛР №2
from tkinter import *
from tkinter.messagebox import showwarning
def coordN(x, y):
    if ((x-1)**2+y**2<=4 and x<= 1) or (y <= -x+3 and y >= x-3):
        return 'yes'
    else :
        return 'no'
def harshad(number):
    copy_number = number
    digit_sum  = 0
    while number > 0 :
        digit_sum +=  number%10
        number = number//10
    if copy_number%digit_sum == 0:
       return 'yes'
    else:
        return 'no'
def verify():
```

```
    try:
        x_ = float(xEntry.get())
        y_ = float(yEntry.get())
        N_ = int(NEntry.get())
    except:
        showwarning('Некоректні дані', 'Значення повині бути числами')
        return None
    color = show.get()
    text = ''
    coord = coordN(x_, y_)
    numberN = harshad(N_)
    result['fg'] = color
    if color  == 'red':
        coord, numberN
    elif color == 'green':
        coord, numberN
    else:
        coord, numberN
    if showC.get():
        text += f'Точка в середині рисунка?\n{coord}\n\n'
    if showN.get():
        text += f'Число N є числом харшад?\n{numberN}\n\n'
    result.insert('1.0', text)
def clear():
    xEntry.delete(0, END)
    yEntry.delete(0, END)
    NEntry.delete(0, END)
    result.delete('1.0', END)
root = Tk()
root.title('Перевірка')
Label(root, text = 'Координати точки:').grid(row=0, column=0, rowspan=2)
```

```
Label(root, text = 'x').grid(row=0, column=1)
Label(root, text = 'y').grid(row=1, column=1)
Label(root, text = 'Число N:').grid(row=2, column=0)
xEntry = Entry(root)
xEntry.grid(row=0, column=2)
yEntry = Entry(root)
yEntry.grid(row=1, column=2)
NEntry = Entry(root)
NEntry.grid(row=2, column=2)
showC, showN = IntVar(), IntVar()
Checkbutton(root, text='Чи є точка всередині рисунка',\
     var=showC).grid(row=3, column=0)
Checkbutton(root, text='Чи є N числом харшад',\
     var=showN).grid(row=3, column=1)
Label(root, text='Результати:').grid(row=4, column=0, columnspan=2)
showing_frame = Frame(root)
showing_frame.grid(row=5, column=0, columnspan=3)
show = StringVar()
black = Radiobutton(showing_frame, var=show, value='black', fg="black")
black['text'] = 'чорний'
black.grid(row=0, column=0)
red = Radiobutton(showing_frame, var=show, value='red', fg="red")
red['text'] = 'червоний'
red.grid(row=0, column=1)
green = Radiobutton(showing_frame, var=show, value='green', fg="green")
green['text'] = 'зелений'
green.grid(row=0, column=2)
show.set('b')
result = Text(root, width=25, height=5)
result.grid(row=6, column=0, rowspan=2)
b_frame = Frame(root)
```

```
b_frame.grid(row=6, column=1, columnspan=2)
Button(b_frame, text='Перевірити', command=verify, bg="white", fg="black",\
     relief=RAISED, activebackground='green').grid(row=6, column=1)
Button(b_frame, text='Очистити', command=clear, bg="white", fg="black",\
     relief=SUNKEN, activebackground='red').grid(row=7, column=1)
root.mainloop()
```



Завдання 9

Написати власний клас для структури: Хеш-таблиця (для розв'язання колізій використати рехешування).

#Файл "PMO21_lab9_var9.py"

```
from hash_table_func import *


h = HashTable()
h.addKV('Богдан', '(096) 030-48-89')
h.addKV('Марія', '(063) 789-45-11')
h.addKV('Марія', '(095) 145-24-14')
h.addKV('Аня', '(098) 254-69-15')
h.addKV('Аліна', '(097) 369-96-45')
h.addKV('Аліса', '(073) 741-36-36')
h.addKV('Мирослава', '(066) 852-87-01')
h.addKV('Аліна', '(099) 258-22-00')
h.printTable()


h.deleteKey('Богдан')
print("\nТелефонний довідник після видалення 'Богдан':")
h.printTable()


print("\nОтримане значення ключа 'Марія':")
print('Марія: ' + h.getKey('Марія'))
```

#Файл "hash_table_func.py"

```
class HashTable:
    def __init__(self):
            self.size = 10
            self.data = [None] * self.size
```

```
def getHash(self, key):
    hash = 0
    for char in str(key):
        hash += ord(char)
    return hash % self.size

def addKV(self, key, value):
    key_hash = self.getHash(key)
    key_value = [key, value]
    if self.data[key_hash] is None:
        self.data[key_hash] = list([key_value])
        return True
    else:
        for pair in self.data[key_hash]:
            if pair[0] == key:
                pair[1] = value
                return True
        self.data[key_hash].append(key_value)
        return True

def getKey(self, key):
    key_hash = self.getHash(key)
    if self.data[key_hash] is not None:
        for pair in self.data[key_hash]:
            if pair[0] == key:
                return pair[1]
    return None

def deleteKey(self, key):
    key_hash = self.getHash(key)
    if self.data[key_hash] is None:
        return False
    for i in range (0, len(self.data[key_hash])):
```

```
            if self.data[key_hash][i][0] == key:
                self.data[key_hash].pop(i)
                return True
        return False


    def printTable(self):
        print('Телефонний довідник:')
        for item in self.data:
            if item is not None:
                print(str(item))
```

```
Телефонний довідник:
[['Аліна', '(099) 258-22-00']]
[['Аліса', '(073) 741-36-36']]
[['Богдан', '(096) 030-48-89'], ['Марія', '(095) 145-24-14']]
[['Мирослава', '(066) 852-87-01']]
[['Аня', '(098) 254-69-15']]

Телефонний довідник після видалення 'Богдан':
Телефонний довідник:
[['Аліна', '(099) 258-22-00']]
[['Аліса', '(073) 741-36-36']]
[['Марія', '(095) 145-24-14']]
[['Мирослава', '(066) 852-87-01']]
[['Аня', '(098) 254-69-15']]

Отримане значення ключа 'Марія':
Марія: (095) 145-24-14
```