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

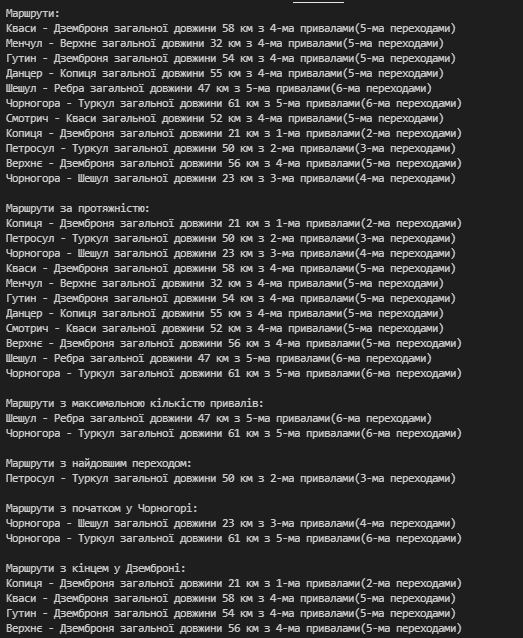
Смотрич, Кваси, 3, 7, 18, 14, 10

Копиця, Дземброня, 15, 6

Петросул, Туркул, 14, 16, 20

Верхнє, Дземброня, 5, 9, 11, 14, 17

Чорногора, Шешул, 4, 8, 7, 4



**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()

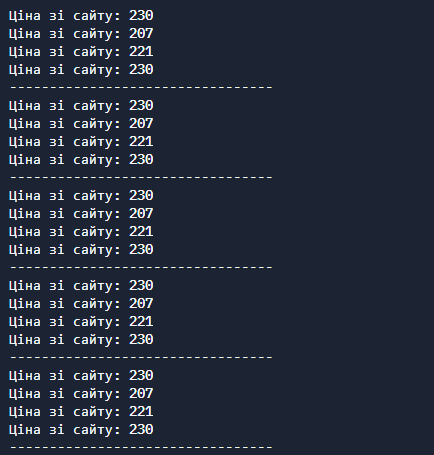
th4.start()

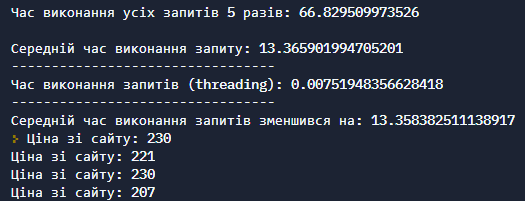
end\_time2 = time.time() - start\_time2

print("Час виконання запитів (threading):", end\_time2)

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

print("Середній час виконання запитів зменшився на:", average\_time - end\_time2)





**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):

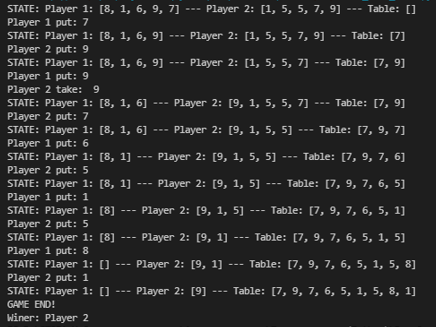
return self.cards

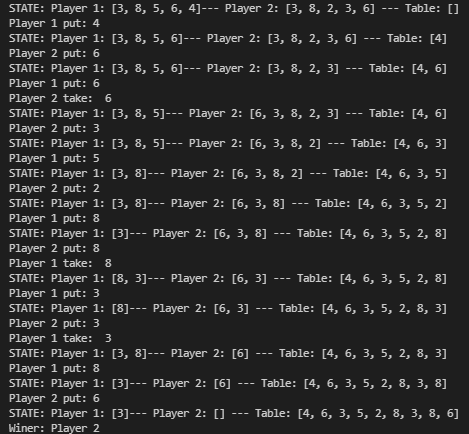
def put\_card(self):

return self.cards[len(self.cards)-1]

def take\_card(self, card):

self.cards.insert(0, card)





**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)

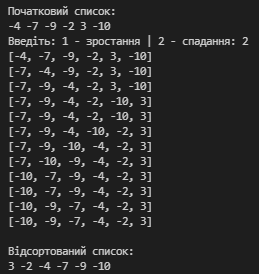
elif count == 2:

cocktail\_sort(elem)

elem.sort(reverse = True)

else:

print("\nНекоректно введені дані")



**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)

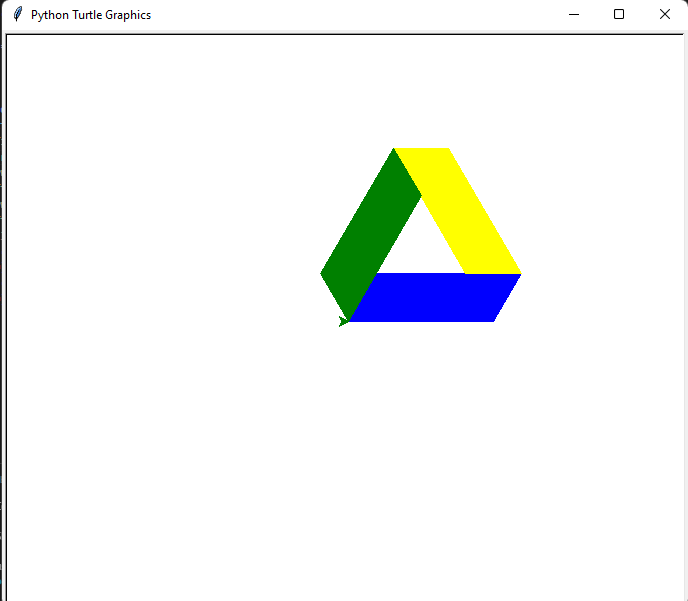
t.forward(55)

t.left(60)

colors = ["blue", "yellow", "green"]

paral(colors)

input("")



**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()

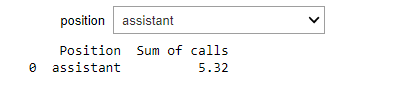
histogram()

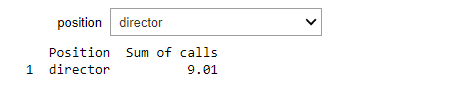
sort\_all\_talk = df\_calls.groupby('Surname')['Duration'].sum().reset\_index(name='Duration')

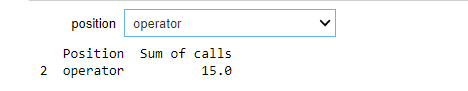
sort\_all\_talk = sort\_all\_talk.sort\_values(by=['Duration'])[-4:]

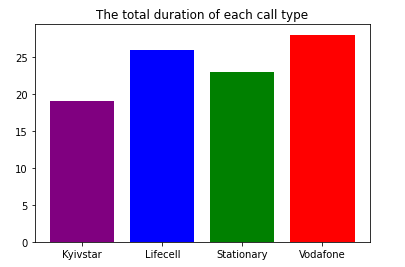
print("The most talkative:")

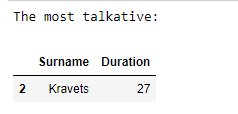
sort\_all\_talk.tail(1)











**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)

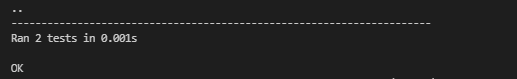
self.assertEqual(days\_in\_year(-2), 0)

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\_\_':

unittest.main()  
  


**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)

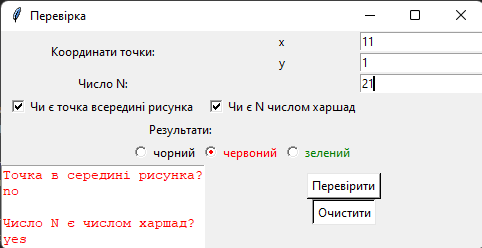
Button(b\_frame, text='Перевірити', command=verify, bg="white", fg="black",\

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))

