Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

**ЗВІТ**

**з обчислювальної практики**

Вайнагія Данила Вікторовича

студента факультету математики та інформатики

Чернівецького національного університету імені Юрія Федьковича

спеціальності «Комп’ютерні науки»

курсу 2 групи 211

Чернівці – 2023

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Оцінка | 10 | 10 | 0 | 0 | 20 | 25 | 17 |
| Макс. | 10 | 10 | 10 | 5 | 20 | 25 | 20 |
| Назва | Завд.  1 | Пр.з.1 | Пр.з.2 | Презент. | Завд.  3 | Завд.  4 | М-К |
| Завд.2 | | |

Вайнагій Д.В.

студента 211 групи

факультету математики

та інформатики

**Варіант №4**

**Завдання 1.** Побудувати графічне зображення астроїди.

**Код Програми**

class Astroid:

    def \_\_init\_\_(self):

        self.x\_coords = None

        self.y\_coords = None

        self.color = None

        plt.figure(figsize=(5, 5))

    def calculate\_coords(self, a):

        t = np.linspace(0, 2\*np.pi, 1000)

        self.x\_coords = a \* np.cos(t)\*\*3

        self.y\_coords = a \* np.sin(t)\*\*3

    def build(self):

            plt.clf()

            plt.plot(self.x\_coords, self.y\_coords, color=self.color)

            plt.title("Астроїда")

            plt.xlabel("x")

            plt.ylabel("y")

            plt.axis("equal")

            plt.grid(True)

            plt.show()

class Application(tk.Tk):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("Графічна фігура")

        self.geometry("400x400")

        self.astroid = Astroid()

        self.button\_astroid = tk.Button(self, text="Побудувати астроїду", command=self.plot\_astroid, font=("Arial", 14))

        self.button\_astroid.pack(fill=tk.BOTH, padx=20, pady=10)

        self.button\_code2 = tk.Button(self, text="Повернути", command=self.rotate, font=("Arial", 14))

        self.button\_code2.pack(fill=tk.BOTH, padx=20, pady=10)

        self.button\_code3 = tk.Button(self, text="Масштабувати", command=self.resize, font=("Arial", 14))

        self.button\_code3.pack(fill=tk.BOTH, padx=20, pady=10)

        self.animation = tk.Button(self, text="Анімація", command=self.resize, font=("Arial", 14))

        self.button\_code3.pack(fill=tk.BOTH, padx=20, pady=10)

        self.button\_circle\_animation = tk.Button(self, text="Анімація кола", command=self.start\_circle\_animation, font=("Arial", 14))

        self.button\_circle\_animation.pack(fill=tk.BOTH, padx=20, pady=10)

    def plot\_astroid(self):

        dialog = simpledialog.askfloat("Масштабний параметр", "Введіть масштабний параметр астроїди:")

        if dialog is not None:

            self.astroid.calculate\_coords(dialog)

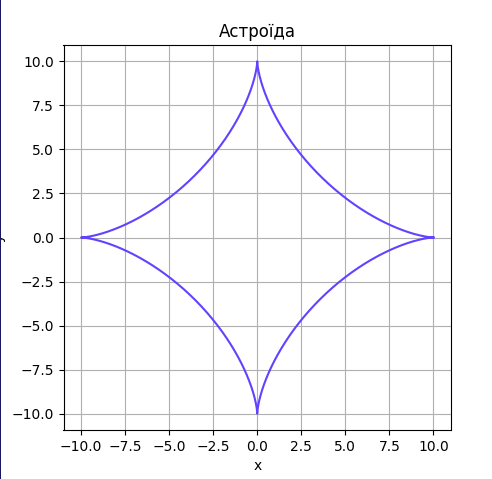
            color\_dialog = colorchooser.askcolor(title="Виберіть колір астроїди")

            if color\_dialog[1] is not None:

                self.astroid.color = color\_dialog[1]

                self.astroid.build()

Результат роботи програми:



**Алгоритм роботи**

1. Створюємо клас астрохди в якому зберігаються її координати, колір, а також обчислюються її параметри і будується сама астроїда
2. Створюємо GUI який використовуватимемо для виконання 1, 3, 4 завдань
3. Для завдання 1 створюємо кнопку яка викликатиме діалогове вікно для введення масштабу, а потім для вибору кольору.
4. Після цього визначаються координати за формулою і малюється астроїда.

**Завдання 2.** Задайте вектор V1, що складається з трьох елементів {2,3,1}, і вектор V2 = {4,1,7}. Виконайте наступні операції: V1\*3, V1-V2, V1\*V2, V1xV2, підсумуйте елементи V1, транспонуйте вектор V2, обчисліть норму вектора V1; використовуючи операцію векторизації, обчисліть sin(V1) та норму одержаного вектора.

**Код програми**

V1 = [2, 3, 1];

V2 = [4, 1, 7];

% V1 \* 3

result1 = V1 \* 3;

disp('V1 \* 3:');

disp(result1);

% V1 - V2

result2 = V1 - V2;

disp('V1 - V2:');

disp(result2);

% V1 .\* V2 (element-wise multiplication)

result3 = V1 .\* V2;

disp('V1 .\* V2 (element-wise multiplication):');

disp(result3);

% cross(V1, V2) (cross product)

result4 = cross(V1, V2);

disp('cross(V1, V2) (cross product):');

disp(result4);

% Sum of elements in V1

result5 = sum(V1);

disp('Sum of elements in V1:');

disp(result5);

% Transpose of vector V2

result6 = V2';

disp('Transposed vector V2:');

disp(result6);

% Norm of vector V1

result7 = norm(V1);

disp('Norm of vector V1:');

disp(result7);

% sin(V1) and norm of the resulting vector

result8 = sin(V1);

norm\_result8 = norm(result8);

disp('sin(V1):');

disp(result8);

disp('Norm of the vector sin(V1):');

disp(norm\_result8);

Результат роботи:

>>

V1 \* 3:

6 9 3

V1 - V2:

-2 2 -6

V1 .\* V2 (element-wise multiplication):

8 3 7

cross(V1, V2) (cross product):

20 -10 -10

Sum of elements in V1:

6

Transposed vector V2:

4

1

7

Norm of vector V1:

3.7417

sin(V1):

0.9093 0.1411 0.8415

Norm of the vector sin(V1):

1.2469

>>

**Завдання 3.** Масштабування фігури відносно геометричного центру.

Поворот фігури відносно геометричного центру.

**Код програми:**

**//додано до завдання 1**

def \_\_add\_\_(self, angle):

        theta = np.radians(-angle)

        x\_new = self.x\_coords \* np.cos(theta) - self.y\_coords \* np.sin(theta)

        y\_new = self.x\_coords \* np.sin(theta) + self.y\_coords \* np.cos(theta)

        self.x\_coords = x\_new

        self.y\_coords = y\_new

    def \_\_mul\_\_(self, factor):

        self.x\_coords \*= factor

        self.y\_coords \*= factor

/\*

\*/

def rotate(self):

        change = simpledialog.askfloat("Поворот", "Введіть градус повороту астроїди:")

        if change is not None:

            self.astroid+change

            self.astroid.build()

    def resize(self):

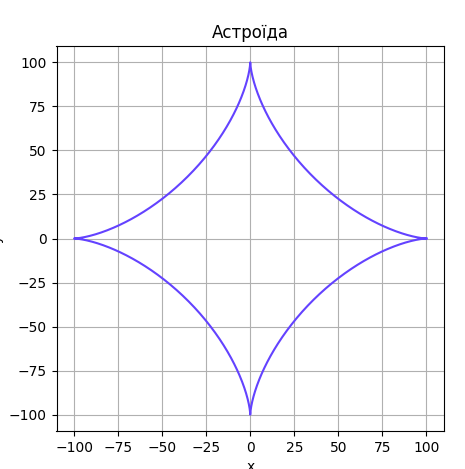
        change = simpledialog.askfloat("Маштабування", "Введіть коефіцієнт маштабування астроїди:")

        if change is not None:

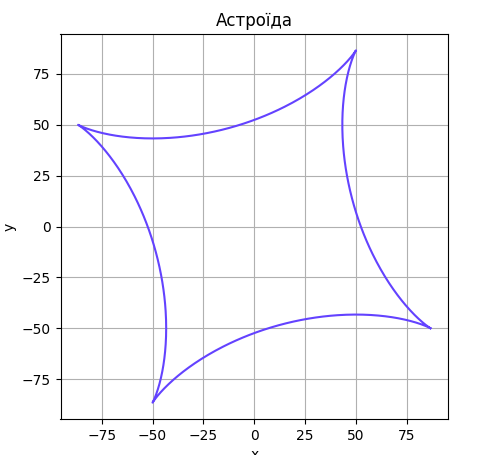
            self.astroid\*change

            self.astroid.build()

**Результат**



(Масштабування в 10 разів)



(Поворот на 30 градусів)

**Алгоритм виконання**

1. Перевизначаємо в створеному раніше класі операції додавання та множення
2. Додаємо код для кнопок масштабування та повороту (для масштабування використовуємо множення астроїди на число, для повороту додавання кута повороту)

**Завдання 4.** Зобразити коло, що рухається всередині прямокутника, відбиваючись від його сторін.

**Код програми**

class CircleAnimation:

    def \_\_init\_\_(self, width, height, radius, speed, color):

        pygame.init()

        self.width = width

        self.height = height

        self.radius = radius

        self.speed = speed

        self.screen = pygame.display.set\_mode((width, height))

        self.clock = pygame.time.Clock()

        self.circle\_color = color[0]

        self.circle\_position = [radius, radius]

        self.circle\_velocity = list(speed)

    def animate(self):

        running = True

        while running:

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    running = False

            self.circle\_position[0] += self.speed[0]

            self.circle\_position[1] += self.speed[1]

            if self.circle\_position[0] + self.radius > self.width or self.circle\_position[0] - self.radius < 0:

                self.speed[0] = -self.speed[0]

            if self.circle\_position[1] + self.radius > self.height or self.circle\_position[1] - self.radius < 0:

                self.speed[1] = -self.speed[1]

            self.screen.fill((255, 255, 255))

            pygame.draw.circle(self.screen, self.circle\_color, self.circle\_position, self.radius)

            pygame.display.flip()

            self.clock.tick(60)

        pygame.quit()

/\*

\*/

def start\_circle\_animation(self):

        width = 800

        height = 600

        radius = simpledialog.askfloat("Радіус", "Введіть радіус кола:")

        speed = [random.randint(1,5), random.randint(1,5)]

        color = colorchooser.askcolor(title="Виберіть колір астроїди")

        print(color)

        animation = CircleAnimation(width, height, radius, speed, color)

        animation.animate()

        sys.exit()



**Алгоритм виконання**

1. Створюємо клас для анімації
2. З допомогою pygame створюємо анімацію руху кола з швидкістю 60 кадрів/сек (на кожному кадрі обчислюються нові координати і малюється зображення, також перевіряється чи не дотикається коло до сторін, в цьому випадку змінюємо вектор швидкості на протилежний)
3. В графічному інтерфейсі додаємо вибір радіусу кола і його кольору