Проєкт 2. Черепашачі перегони

Етап 1. Організаційний

Завдання. Створити проєкт, який реалізує схему перегонів Черепашок.

Етап 2. Підготовчий

1. Підключення модулів:

- Модуль **turtle** — для створення вікна та змоги працювати з графічними об'єктами.
- Підключаємо модуль **random** — для вибору рандомних чисел, які відповідатимуть за рандомну швидкість Черепашок.

```
from turtle import *
from random import randint
```

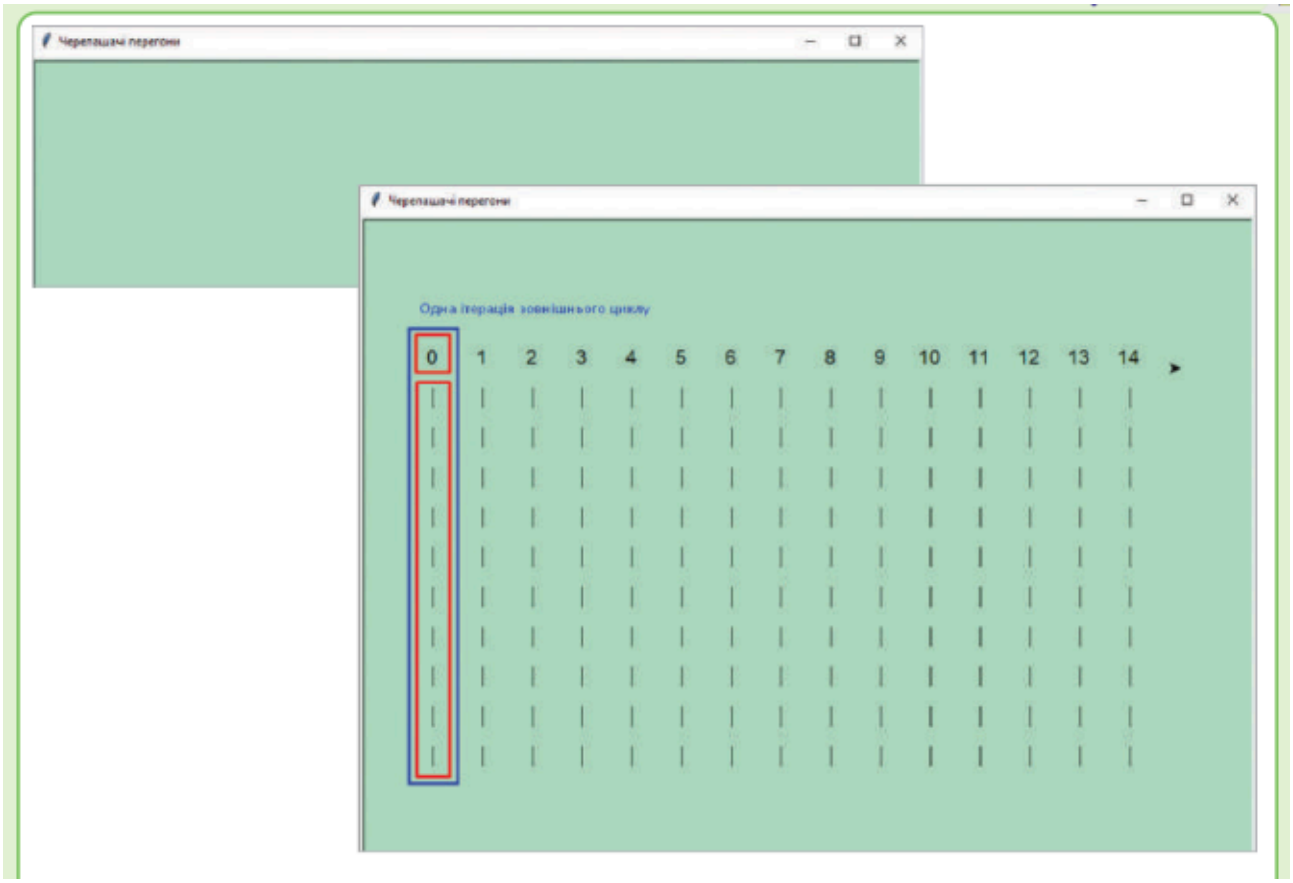
2. Створення вікна:

- Задати розміри вікна (900, 700).
- Задати заголовок вікна **Черепашачі перегони**.
- Задати колір фону вікна.

```
window = Screen()
window.setup(900, 700)
window.title('Черепашачі перегони')
window.bgcolor('#A4F0C8')
```

Ви можете обрати свій колір фону.

3. Створення траси для перегонів. У нашому випадку траса для перегонів складатиметься з 15-ти ліній-етапів.



Щоб реалізувати цю трасу, знадобиться цикл з 15-ти ітерацій, кожна з якої буде містити:

- текстове поле з текстом, який буде містити значення ітерації;
- цикл з 10-ти ітерацій, який малюватиме 10 вертикальних відрізків.

Тому робимо такі дії:

- створюємо Черепашку **t**, яка малюватиме трасу;
- піднімаємо олівець і переміщаємо **t** в точку **(-380, 200)** — верхній лівий куток, з якого буде починатися створення траси;
- задаємо швидкість Черепашки **5**;
- створюємо цикл, який відповідатиме за 15 етапів;
- усередині циклу задаємо команди створення одного етапу: текстове поле **write**, текст якого дорівнюватиме номеру ітерації, розмір тексту **14**;
- цикл, який буде створювати відрізки з пропуском, а саме: піднімає олівець, переміщається на **20** кроків, опускає олівець, переміщається на **20** кроків, малюючи за собою слід.

```
t = Turtle()
t.penup()
t.goto(-380, 200)
t.speed(5)
```

Після цього створюється перший етап. Щоб перейти до наступного етапу, потрібно підняти олівець, повернутися вгору на 400 кроків, повернути Черепашку вправо, рухати прямо на 50 кроків (відстань між етапами). Тоді починається виконання 2 ітерації зовнішнього циклу і т.д.

```
for i in range(15):
        t.write(i, align = 'center', font = ('Arial',
        14, 'normal'))
        t.right(90)

        for i in range(10):
                t.penup()
                t.forward(20)
                t.pendown()
                t.forward(20)

        t.penup()
        t.backward(400)
        t.left(90)
        t.forward(50)
```

Етап 3. Проєктний

Завдання. Додаємо учасників перегонів — чотирьох Черепашок.

У модулі, як Ви знаєте, є форма Черепашки **`turtle`**. Для створення учасника оберемо саме таку форму. Задамо колір червоний та розмір Черепашки 2.

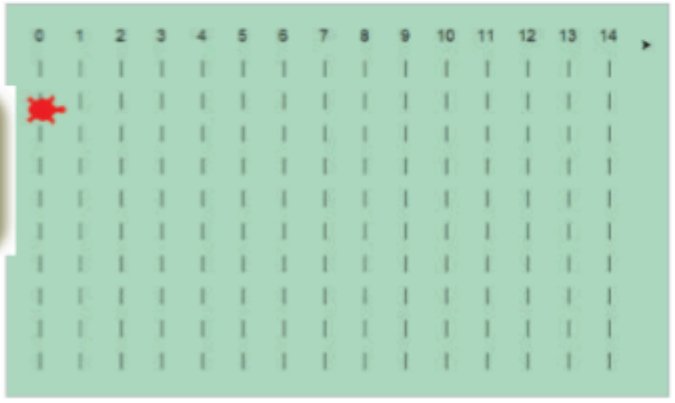
Перемістимо учасника на початок траси для перегонів в точку **`(-380, 120)`**.

Задамо ефект, як з'являтиметься учасник на початку, а саме робить оберт по колу.

```
player_1 = Turtle()
player_1.color('red')
player_1.shape('turtle')
player_1.shapesize(2)
```

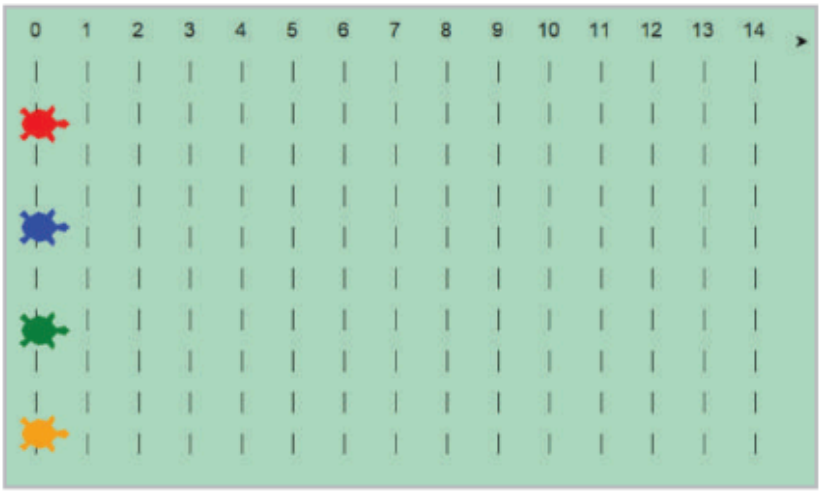
```
player_1.penup()
player_1.goto(-380, 120)
player_1.pendown()
```

```
for i in range(10):
  player_1.right(360/10)
```



Аналогічно створіть самостійно ще трьох учасників:

- **player_2**, Черепашка синього кольору, з початковим розміщенням **-380, 20**. Ефект появи — змінити кількість ітерацій;
- **player_2**, Черепашка зеленого кольору, з початковим розміщенням **-380, -80**. Ефект появи — змінити кількість ітерацій;
- **player_2**, Черепашка оранжевого кольору, з початковим розміщенням **-380, -180**. Ефект появи — змінити кількість ітерацій.



Можете самостійно змінити кількість учасників, їхні кольори та відстань між ними.

Створюємо рух учасників перегонів.

Щоб Черепашка рухалася, ми циклічно будемо переміщати при кожній ітерації вперед на рандомну відстань, наприклад, від **1** до **5**.

```
forward(randint(1, 5))
```

Проте, щоб перегони закінчилися, нам потрібно отримати координату **x** кожної з Черепашок.

```
xp1 = player_1.xcor()
xp2 = player_2.xcor()
xp3 = player_3.xcor()
xp4 = player_4.xcor()
print(xp1, xp2, xp3, xp4)
```

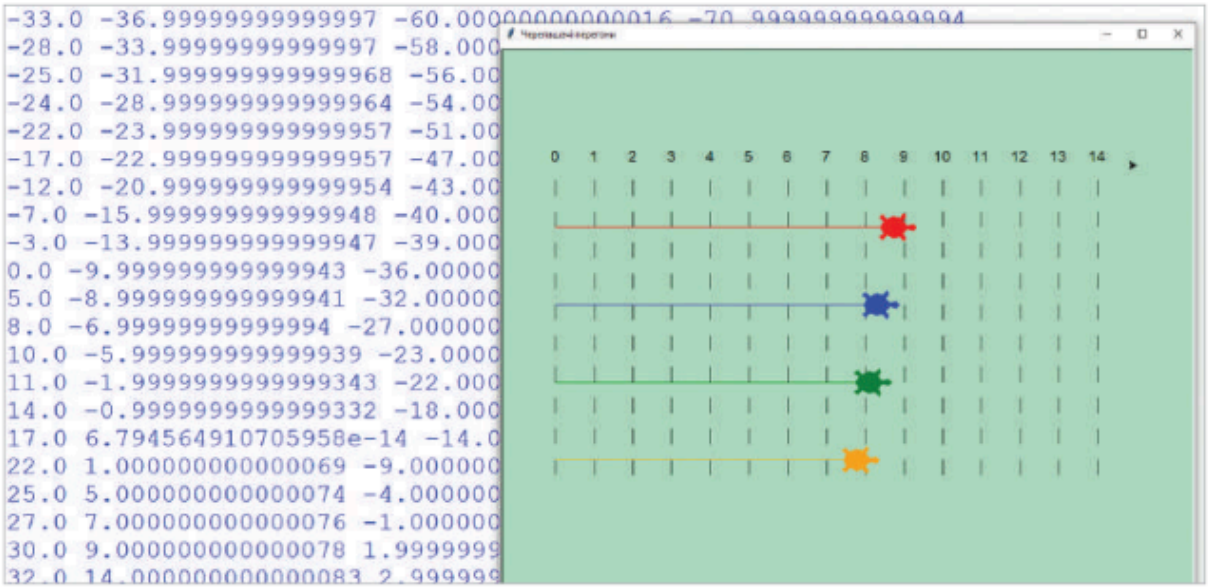
Виведемо ці значення в консоль, щоб протестувати, чи дійсно все правильно працює на цьому етапі програми. Зачекавши, доки виконання дійде до певного рядка, ми отримаємо потрібний результат в консолі.

```
-380 -380 -380 -380
```

Зробимо ще один тест для перевірки, яка координата по **x** нам потрібна для Черепашки, щоб рух Черепашок зупинився. Для цього запустимо цикл на близько 220-230 ітерацій, у якому при кожній ітерації кожна Черепашка рухатиметься вперед, і в консоль будуть виводитися координати кожної Черепашки. Щоб зміни координат змінювалися — то цю частину коду з визначенням координат перемістимо в тіло циклу:

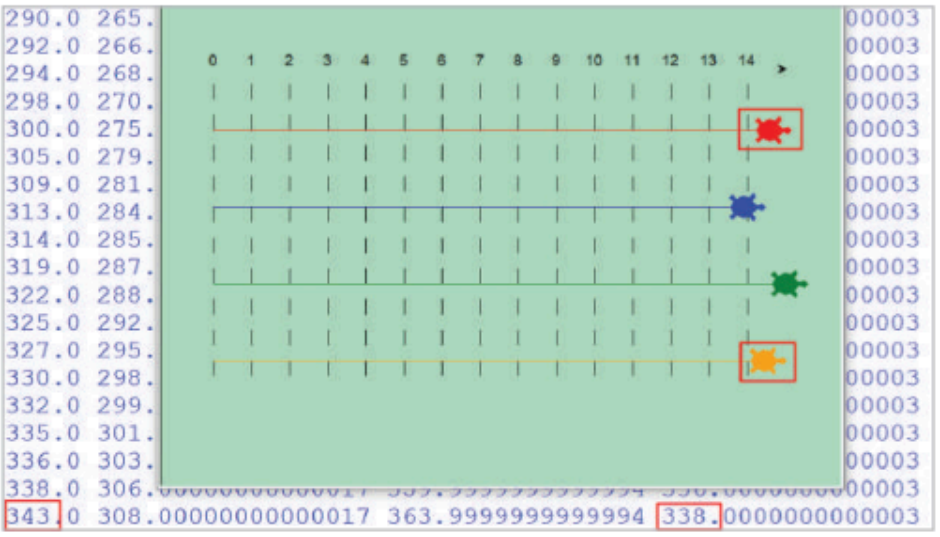
```
for i in range(220):
    xp1 = player_1.xcor()
    xp2 = player_2.xcor()
    xp3 = player_3.xcor()
    xp4 = player_4.xcor()
    print(xp1, xp2, xp3, xp4)
    player_1.forward(randint(1, 5))
    player_2.forward(randint(1, 5))
    player_3.forward(randint(1, 5))
    player_4.forward(randint(1, 5))
```

Запустивши код на виконання, ми побачимо, як в консолі змінюватимуться числа — координати наших Черепашок.



Запускаємо код і тестимо до тих пір, доки не визначимо потрібну координату **x**.

Після кількох тестів у поданому випадку помічаємо, що координати червоної та оранжевої Черепашок більш-менш підходять нам для перемоги. В консолі їхні координати по **x** дорівнюють **343** та **338**. Проте для перемоги ми візьмемо число **340**.



Це значення нам дозволить запустити цикл уже **while**, а не **for**, щоб наші учасники рухалися до тих пір, доки координати у всіх будуть менші за **340**. Як тільки умова перестане бути істинною — рух зупиниться.

```
xp1 = player_1.xcor()
xp2 = player_2.xcor()
xp3 = player_3.xcor()
xp4 = player_4.xcor()

while xp1 < 340 and xp2 < 340 and xp3 < 340 and xp4 < 340:
    xp1 = player_1.xcor()
    xp2 = player_2.xcor()
    xp3 = player_3.xcor()
    xp4 = player_4.xcor()
    print(xp1, xp2, xp3, xp4)
    player_1.forward(randint(1, 5))
    player_2.forward(randint(1, 5))
    player_3.forward(randint(1, 5))
    player_4.forward(randint(1, 5))
```

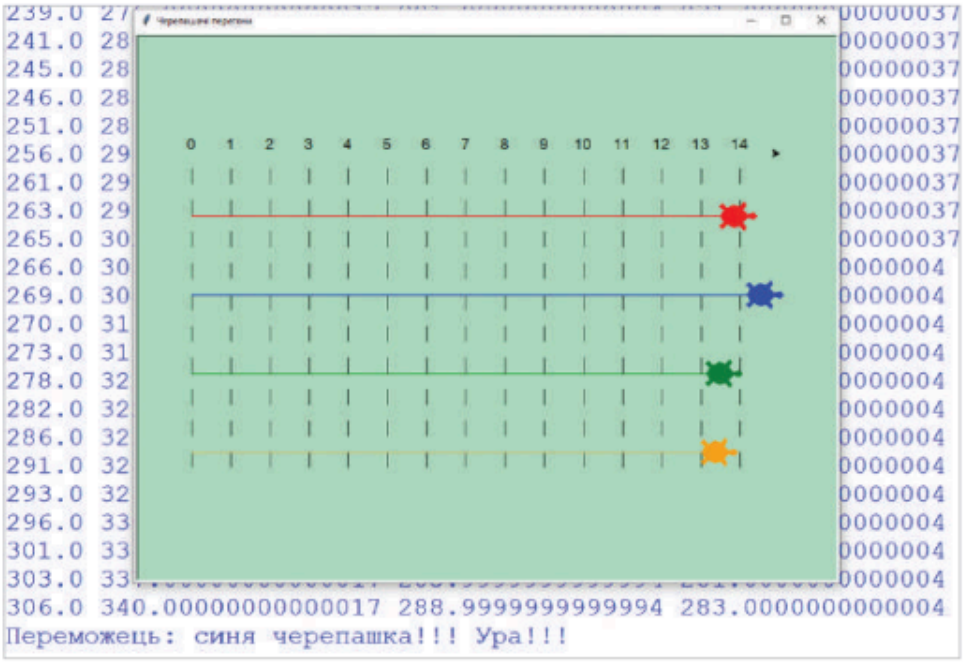
Як тільки побачили, що все працює належним чином — команду **print()** можна вилучити з коду.

Отже, ми використали змінні і всередині циклу, і попереду нього. Попереду — щоб ініціалізувати змінні, які використовуємо в умові. Всередині — переприсвоєння значень змінних.

Визначаємо переможця

Додаємо умови, які визначають переможця. Тобто, який із учасників матиме координату більшу або що дорівнюватиме 340 — то та Черепашка буде переможцем.

```
if xp1 >= 340:
    print('Переможець: червона черепашка!!! Ура!!!')
elif xp2 >= 340:
    print('Переможець: синя черепашка!!! Ура!!!')
elif xp3 >= 340:
    print('Переможець: зелена черепашка!!! Ура!!!')
elif xp4 >= 340:
    print('Переможець: оранжева черепашка!!! Ура!!!')
```



Отримали повідомлення, що перемогла синя Черепашка. Наша програма працює.

Етап 4. Тестувальний

Запустити програму кілька разів, щоб протестувати, чи працює програма так, як потрібно, при цьому змінюючи різні значення, наприклад, швидкостей Черепашок. Якщо буде віднайдено баг — виправити його.

Додатково:

1. Вивести повідомлення про перемогу не в консоль, а у вікно програми за допомогою функції **`write()`**.

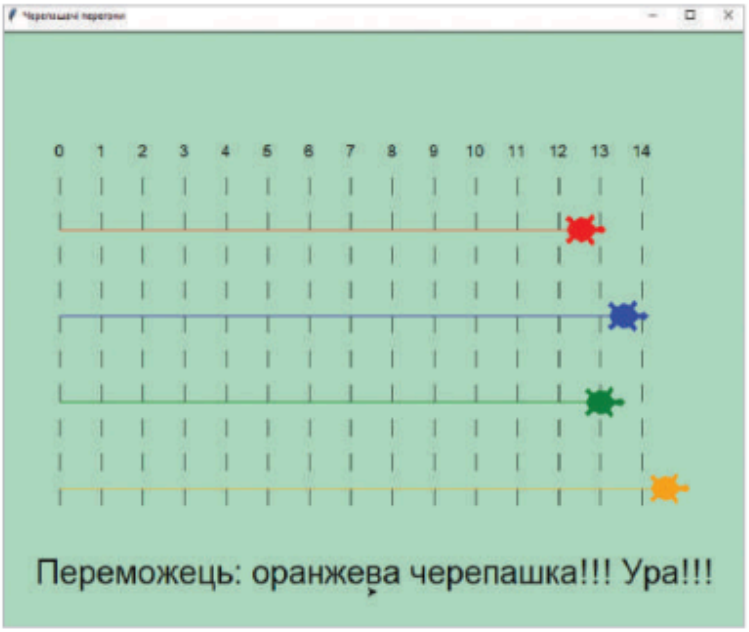
   Примітка:

   В умовах не використовувати функцію **`print()`**, а присвоїти значення змінній.

```
if xp1 >= 340:
    text = 'Переможець: червона черепашка!!! Ура!!!'
```

- Перемістити Черепашку **t** в місце, де хочете розмістити текст, і по аналогії до уже використаної функції **write()** вище в коді — вивести інформацію у вікні.

Наприклад:



2. Справа від ліній траси перегонів створити ще одну лінію — червоного кольору, яка відповідатиме за переможну стрічку. При цьому поміняти значення координат перемоги, бо логічно, що вони змістяться вправо.
3. Додати п'ятого учасника.
4. Запитувати в користувача, яка Черепашка переможе. І в кінці видати повідомлення, чи відгадав користувач переможця.
5. (*) Гра для кількох користувачів:
   - запитує, скільки користувачів братиме участь;
   - кожен вводить своє ім'я та номер Черепашки, яку вважає майбутнім переможцем;
   - у кінці видати повідомлення, яка Черепашка перемогла;
   - звіритися з консоллю, де вводили ім'я, хто з користувачів правильно вгадав.