

Урок 22. Цикли з лічильником

Вивчення нового матеріалу.

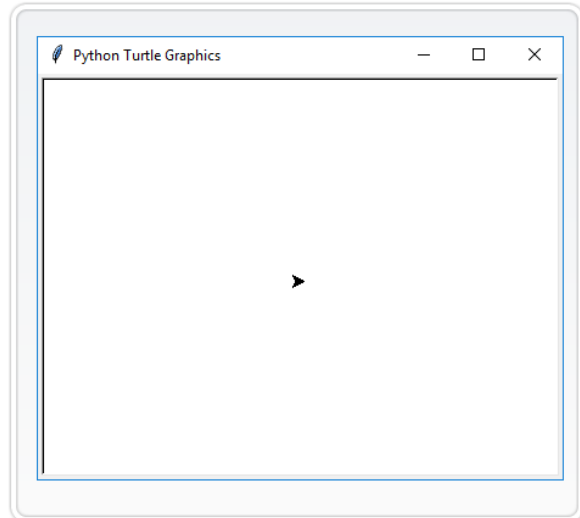
Слайд №1

У Python також можна створювати зображення!

Найпростіші засоби для цього надає модуль **turtle** («черепаха»).

Після виконання двох зазначених нижче команд створюється графічне вікно, у центрі якого розміщується вказівник у вигляді стрілочки. Це і є «черепашка», що малюватиме лінії.

```
import turtle  
turtle.reset()
```



import turtle — команда підключення модуля **turtle**.
turtle.reset — встановлення черепашки в центрі вікна.

Далі

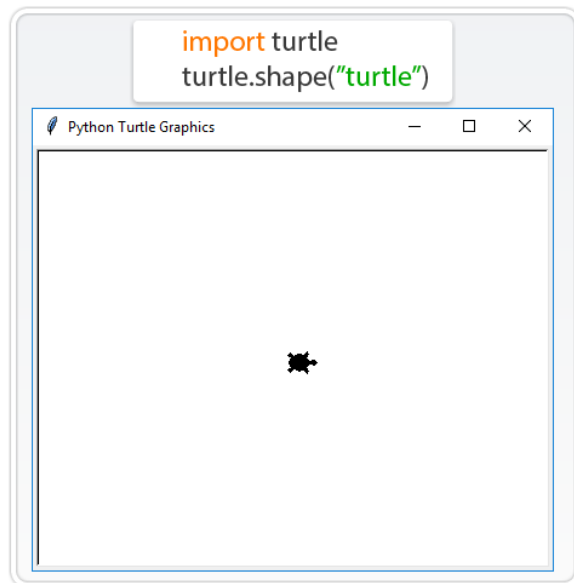
Слайд №2

Якщо хочеш замість звичайної стрілочки бачити саме черепашку, замість команди

```
turtle.reset()
```

введи команду

```
turtle.shape("turtle")
```

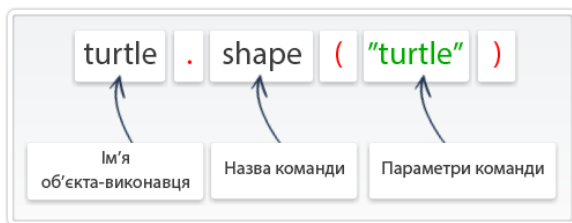


Для вибору вигляду виконавця-художника застосуємо команду **turtle.shape**.

Далі

Слайд №3

Розглянемо детальніше структуру команд для роботи в графічному режимі.



У такий спосіб ми вказуємо об'єкту, що треба виконати команду.

У розглянутій команді об'єктом є "черепашка", вона і є виконавцем алгоритму.

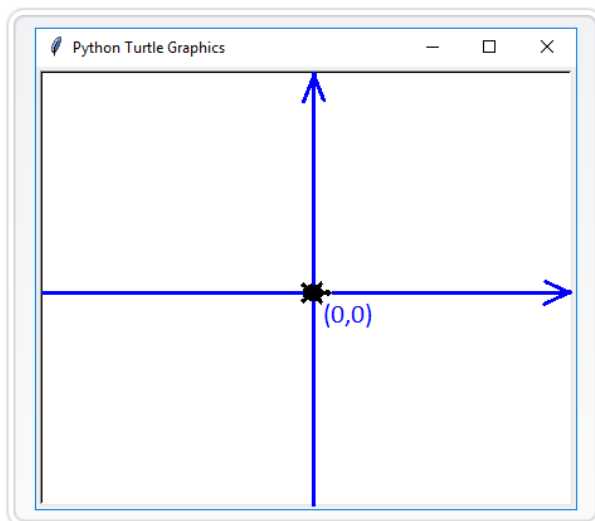


об'єкт.команда(параметри)

Далі

Слайд №4

Графічне вікно являє собою умовну координатну площину, початок відліку якої розташовано в центрі вікна.



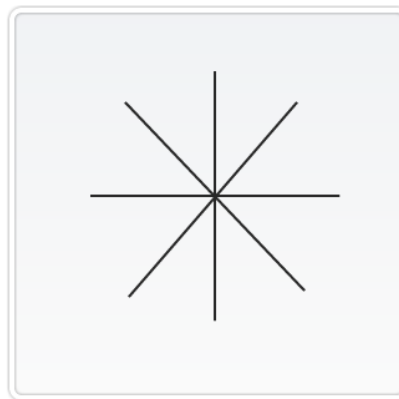
Зауваж, що початкове положення «черепашки» задано координатами (0,0), а початковий напрям її руху — вправо.

Далі

Слайд
№5

Виконавець «черепаха» може виконувати команди переміщення вперед, назад, повороту ліворуч, праворуч, команди переміщення у задану точку тощо.

Розглянемо, як побудувати ось таку зірку.



Побудуємо восьмипроменеву зірку.

Далі

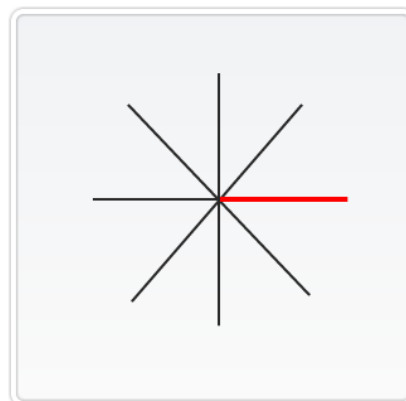
Слайд
№6

Побудову зірки розпочнемо з першого променя, позначеного на рисунку червоним.

Для побудови цього променя необхідно до попереднього коду дописати команду

```
turtle.forward(100)
```

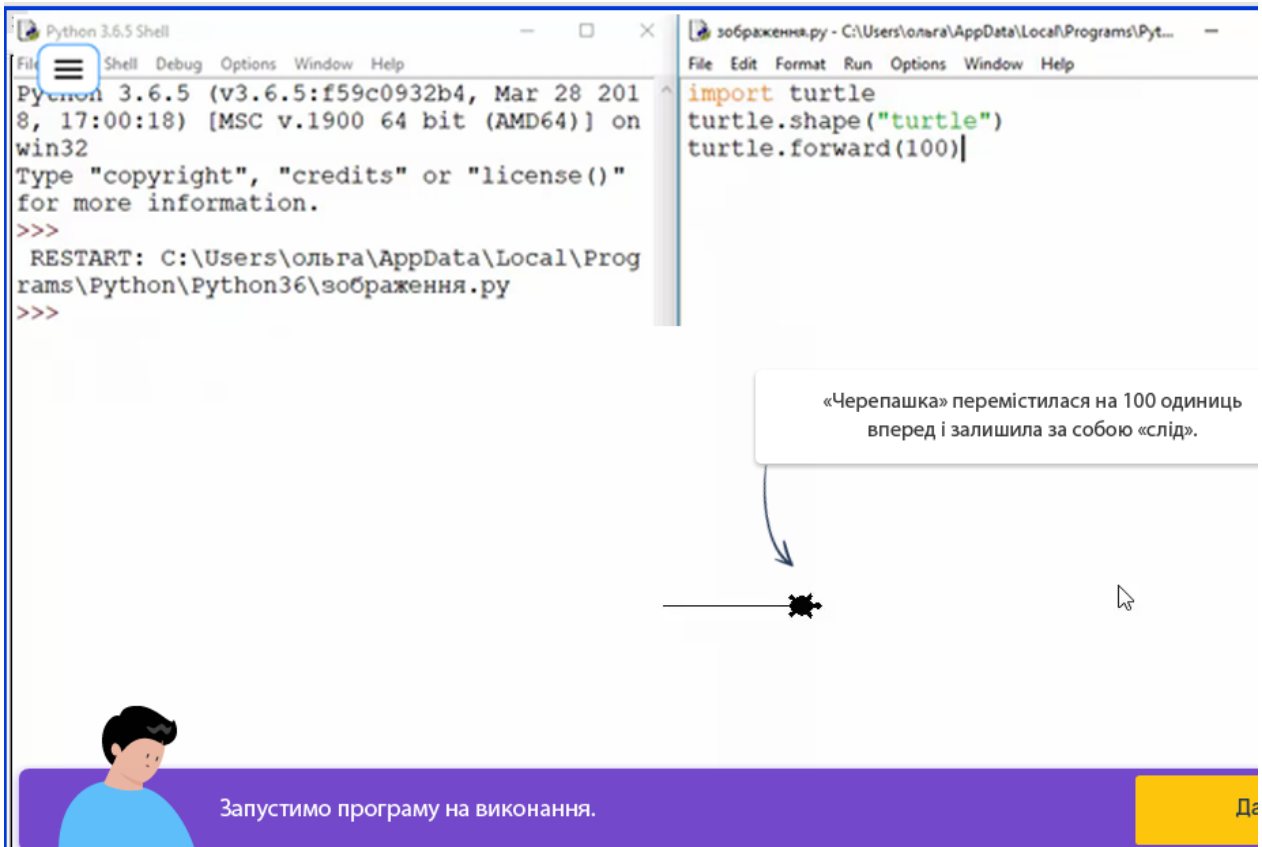
Ця команда перемістить “черепашку” на 100 одиниць вперед від поточного положення.



Олесь продемонструє, як виконується ця програма.

Далі

Слайд
№7



Python 3.6.5 Shell

```
File Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ольга\AppData\Local\Programs\Python\Python36\зображення.py
>>>
```

зображення.py - C:\Users\ольга\AppData\Local\Programs\Python\Python36\зображення.py

```
File Edit Format Run Options Window Help
import turtle
turtle.shape("turtle")
turtle.forward(100)
```

«Черепашка» перемістилася на 100 одиниць вперед і залишила за собою «слід».

Запустимо програму на виконання.

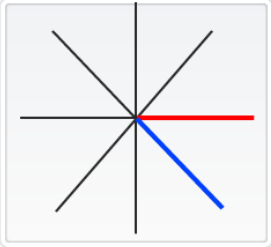
Далі

Слайд
№8

Дійсно, перед тим як побудувати наступний промінь, необхідно повернутися у початкове положення та повернути на 45° за годинниковою стрілкою.

Нижче наведено команди переходу в задану точку площини та повороту виконавця.

<code>turtle.goto(x,y)</code>	переміщення "черепашки" в точку з координатами (x,y)
<code>turtle.right(n)</code>	поворот "черепашки" на n градусів за годинниковою стрілкою
<code>turtle.left(n)</code>	поворот "черепашки" на n градусів проти годинникової стрілки



Запам'ятай основні команди виконавця "черепашка".

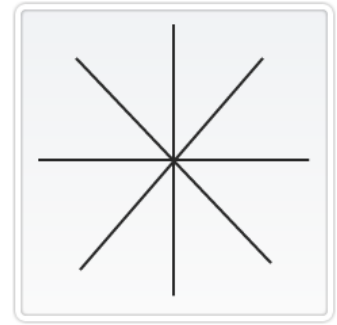
Далі

Слайд
№9

Дійсно, для побудови усієї зірки можна скопіювати останні три команди 7 разів. У результаті ми отримаємо доволі довгий код:

```
import turtle
turtle.shape("turtle")
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
```

```
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
```



Код дуже довгий, спробуємо його скоротити!

Далі

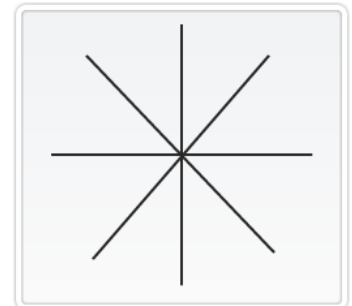
Слайд
№10

```
import turtle
turtle.shape("turtle")
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
turtle.forward(100)
turtle.goto(0,0)
turtle.right(45)
```

У Python, як і в інших мовах програмування, також можна користуватися циклами.

Створену нами раніше громіздку програму за допомогою циклу можна записати ось так:

```
import turtle
turtle.shape("turtle")
for i in range(8):
    turtle.forward(100)
    turtle.goto(0,0)
    turtle.right(45)
```



З використанням циклів програма стає набагато коротшою!

Далі

Слайд
№11

Python 3.6.5 Shell

File Shell Debug Options Window Help

Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ольга\AppData\Local\Programs\Python\Python36\зображення.py
>>>
RESTART: C:\Users\ольга\AppData\Local\Programs\Python\Python36\зображення.py
>>>
RESTART: C:\Users\ольга\AppData\Local\Programs\Python\Python36\зображення.py
>>>
RESTART: C:\Users\ольга\AppData\Local\Programs\Python\Python36\зображення.py
>>>

зображення.py - C:\Users\ольга\AppData\Local\Programs\Pyt...

File Edit Format Run Options Window Help

import turtle
turtle.shape("turtle")
for i in range(8):
 turtle.forward(100)
 turtle.goto(0,0)
 turtle.right(45)|

Запустимо програму на виконання.

Слайд
№12

Ось загальний вигляд оператора циклу з лічильником у мові Python:

for змінна-лічильник in діапазон:
 тіло циклу

У нашому коді:

змінна-лічильник

діапазон, у якому лічильник змінюється

for i in range(8):
 turtle.forward(100)
 turtle.goto(0,0)
 turtle.right(45)

тіло циклу

Зверни увагу на відступи в тілі циклу. Якщо команди тіла записати без відступів, вони циклічно виконуватися вже не будуть.

Далі

Як сформувати діапазон значень, що їх набуватиме змінна?

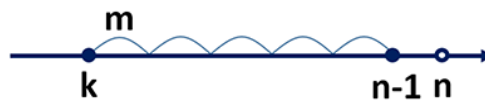
діапазон значень змінної

```
for i in range(8):  
    turtle.forward(100)  
    turtle.goto(0,0)  
    turtle.right(45)
```

У мові Python для цього використовується спеціальна функція **range**, що має такий вигляд:

```
range(k,n,m)
```

k — початкове значення діапазону.
n — наступне число після кінцевого значення діапазону, тобто діапазон буде завершено значенням $n-1$.
m — крок, на який збільшуватиметься змінна.

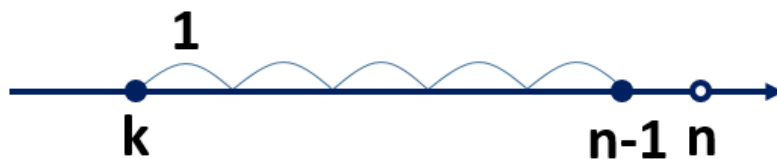


Крок — це на скільки збільшуватиметься змінна кожної ітерації циклу.

Далі

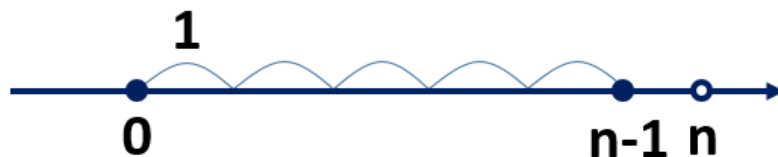
Функцію **range** можна записати без встановлення значення кроку, тоді крок дорівнюватиме 1:

```
range(k,n)
```



Також функцію **range** можна записати без початкового значення, тоді воно дорівнюватиме 0:

```
range(n)
```



У функції **range** не обов'язково вказувати всі параметри.

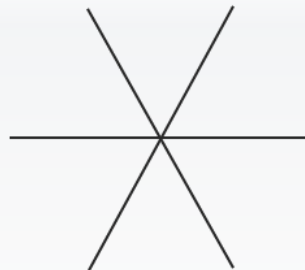
Вправа
№1



Вправа 1. Створи програму побудови зірки в середовищі Python.

Побудова зірки

Створи програму побудови
шестипроменевої зірки.



Створи програму побудови зірки в
середовищі Python і перевір правильність її виконання.

Готово

Вправа
№2



Вправа 2. Створи програму побудови зірки в середовищі Python.

Побудова зірки

Скопіюй файл щойно створеної програми і
зміни її код так, щоб будувалася 7-променева
зірка.



Перетвори попередню програму та перевір правильність її виконання.

Готово

Вправа 3.

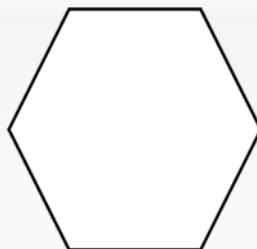
Вправа №3



Вправа 3. Створи програму розв'язання задачі в Python.

Задача "Побудова n-кутника"

Користувач вводить число n , а «черепашка» має малювати правильний n -кутник з довжиною сторони 100.



Створи програму розв'язання задачі «Побудова n-кутника» в середовищі Python та перевір правильність її виконання.

Готово

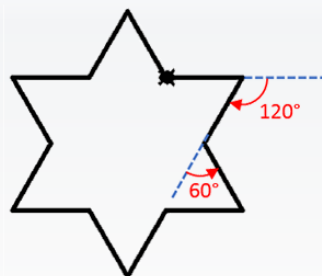
Домашнє завдання



Домашнє завдання.

Задача

Створи програму побудови зірки згідно зразка.



Створи у Python програму, що малює зірку.

Готово