

## Day 32. 특정 위치 색구하기

우리는 특정 부분에 색을 칠하는 방법에 대해 살펴보았습니다. 그렇다면 특정 위치에 어떤 색이 칠해져 있는지 알 수 있는 방법은 없을까요?

예제를 작성한 후 실행해 보도록 하겠습니다.

getcolor.py

```
import turtle

turtle.setup(500, 500)
turtle.shape("turtle")

turtle.pencolor("black")
turtle.pensize(5)

turtle.fillcolor("red")
turtle.begin_fill()

for x in range(2):
    turtle.forward(60)
    turtle.left(90)
turtle.end_fill()

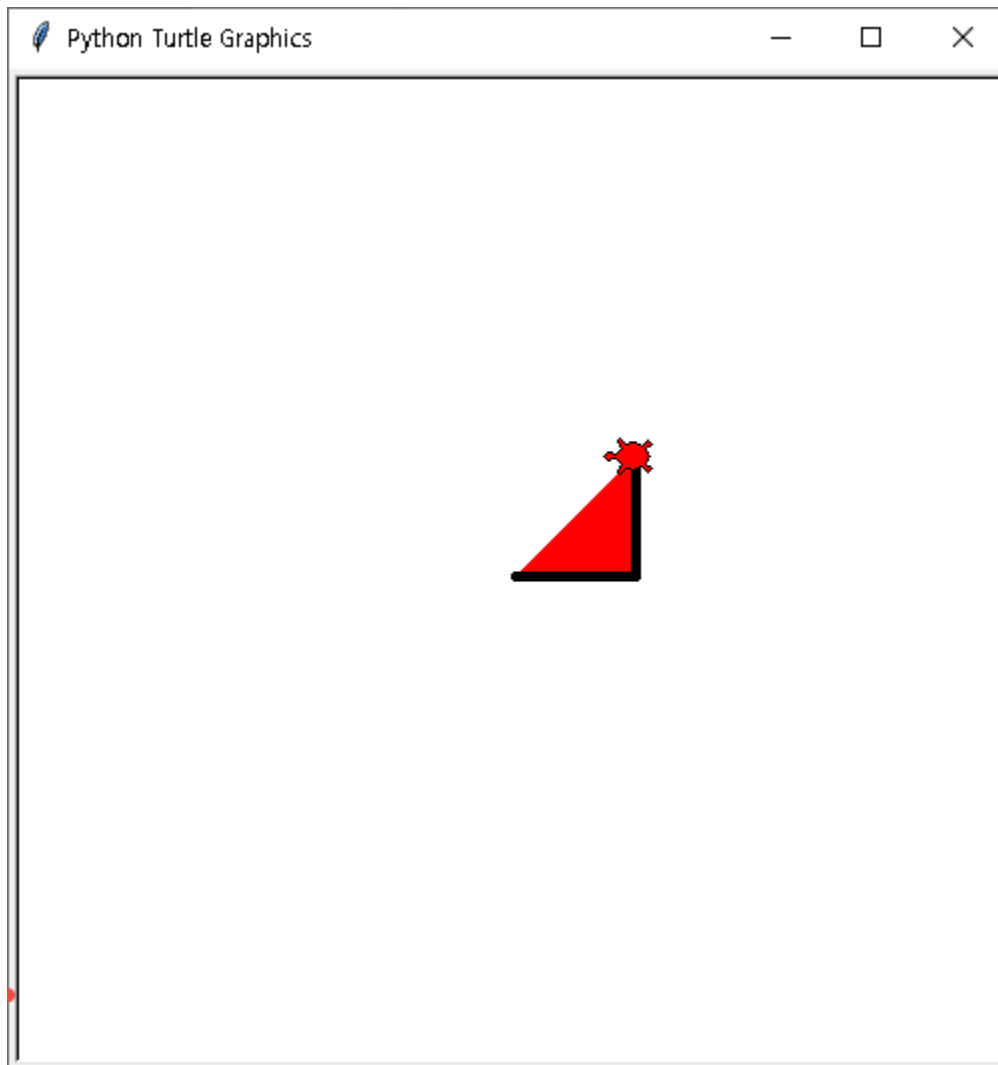
def print_color(x, y):
    color = get_pixel_color(x, y)
    print(color)

def get_pixel_color(x, y):
    y = -y
    canvas = turtle.getcanvas()
    ids = canvas.find_overlapping(x, y, x, y)
    if ids:
        index = ids[-1]
        color = canvas.itemcget(index, "fill")
        if color:
            return color
    return "white"

turtle.onscreenclick(print_color)

turtle.done()
```

실행하면 아래와 같이 그림이 그려집니다..



색깔을 마우스로 클릭해보세요. 검은 선을 한번 클릭하고, 빨강색 부분을 클릭하고, 배경을 클릭해보세요.

아래와 같이 결과가 나오는 것을 알 수 있습니다.

```
black
```

```
red
```

```
white
```

어떻게 이런 결과가 나왔는지 확인해 보도록 하겠습니다.

```
turtle.onscreenclick(print_color)
```

마우스를 클릭하면 `print_color` 함수가 호출됩니다. 해당 함수에는 마우스로 클릭한 `(x, y)`좌표가 전달됩니다.

```
def print_color(x, y):  
    color = get_pixel_color(x, y)  
    print(color)
```

`print_color()`함수는 `get_pixel_color()`함수를 이용하여 색을 구하고 그 색을 `print()`함수로 출력하고 있습니다. 실제로 색을 구하는 것은 `get_pixel_color()`함수라는 것을 알 수 있습니다.

```
def get_pixel_color(x, y):  
    y = -y  
    canvas = turtle.getcanvas()  
    ids = canvas.find_overlapping(x, y, x, y)  
    if ids:  
        index = ids[-1]  
        color = canvas.itemcget(index, "fill")  
        if color:  
            return color  
    return "white"
```

`turtle.getcanvas()`는 거북이가 그림을 그리는 공간인 `canvas`에 대한 객체를 리턴합니다. 이 `canvas`객체를 이용하면 해당 `canvas`에 어떤 색이 칠해져 있는지 알 수 있습니다.

```
ids = canvas.find_overlapping(x, y, x, y)  
if ids:  
    index = ids[-1]  
    color = canvas.itemcget(index, "fill")  
    if color:  
        return color  
  
return "white"
```

위의 부분은 **canvas**에서 클릭한 위치에 해당하는 색을 찾아 그 결과를 리턴합니다. 만약 색을 찾지 못했다면 **"white"**를 리턴합니다. 이 부분을 이해하려면 **Canvas**객체에 대한 많은 이해가 필요합니다. 인터넷에서 해당 객체를 사용하는 방법에 대해 찾아보세요.

## 속제

우리는 몇일 동안 거북이 즉 **Turtle** 객체를 이용해 그림 그리는 방법에 대해 학습을 하고 있습니다. **Turtle** 객체에 대해 사실 가장 자세히 아는 방법은 공식 문서를 읽는 것입니다.

구글에서 "python turtle document" 라고 검색해보세요. 가장 첫번째 나오는 링크를 클릭하시면 다음과 같은 사이트가 보여질 것입니다.

( <https://docs.python.org/3.3/library/turtle.html?highlight=turtle> )

Python » 3.3.7 Documentation » The Python Standard Library » 24. Program Frameworks »

previous | next | modules | index

### Table Of Contents

- 24.1. **turtle** — Turtle graphics
  - 24.1.1. Introduction
  - 24.1.2. Overview of available Turtle and Screen methods
    - 24.1.2.1. Turtle methods
    - 24.1.2.2. Methods of TurtleScreen/Screen
  - 24.1.3. Methods of RawTurtle/Turtle and corresponding functions
    - 24.1.3.1. Turtle motion
    - 24.1.3.2. Tell Turtle's state
    - 24.1.3.3. Settings for measurement
    - 24.1.3.4. Pen control
      - 24.1.3.4.1. Drawing state
      - 24.1.3.4.2. Color control
      - 24.1.3.4.3. Filling
      - 24.1.3.4.4. More drawing control
    - 24.1.3.5. Turtle state
      - 24.1.3.5.1. Visibility
      - 24.1.3.5.2. Appearance
    - 24.1.3.6. Using events
    - 24.1.3.7. Special Turtle methods
    - 24.1.3.8. Compound shapes
  - 24.1.4. Methods of TurtleScreen/Screen and corresponding functions
    - 24.1.4.1. Window

## 24.1. **turtle** — Turtle graphics

### 24.1.1. Introduction

Turtle graphics is a popular way for introducing programming to kids. It was part of the original Logo programming language developed by Wally Feurzig and Seymour Papert in 1966.

Imagine a robotic turtle starting at (0, 0) in the x-y plane. After an `import turtle`, give it the command `turtle.forward(15)`, and it moves (on-screen) 15 pixels in the direction it is facing, drawing a line as it moves. Give it the command `turtle.right(25)`, and it rotates in-place 25 degrees clockwise.

By combining together these and similar commands, intricate shapes and pictures can easily be drawn.

The **turtle** module is an extended reimplementaion of the same-named module from the Python standard distribution up to version Python 2.5.

It tries to keep the merits of the old turtle module and to be (nearly) 100% compatible with it. This means in the first place to enable the learning programmer to use all the commands, classes and methods interactively when using the module from within IDLE run with the `-n` switch.

The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways. Because it uses **tkinter** for the underlying graphics, it needs a version of Python installed with Tk support.

The object-oriented interface uses essentially two+two classes:

1. The **TurtleScreen** class defines graphics windows as a playground for the drawing turtles. Its constructor needs a **tkinter.Canvas** or a **ScrolledCanvas** as argument. It should be used when **turtle** is used as part of some application.

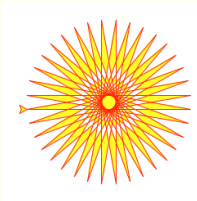
The function **Screen()** returns a singleton object of a **TurtleScreen** subclass. This function should be used when **turtle** is used as a standalone tool for doing graphics. As a singleton object, inheriting from its class is not possible.

All methods of TurtleScreen/Screen also exist as functions, i.e. as part of the procedure-oriented interface.

2. **RawTurtle** (alias: **RawPen**) defines Turtle objects which draw on a **TurtleScreen**. Its constructor

#### Turtle star

Turtle can draw intricate shapes using programs that repeat simple moves.



```
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill()
done()
```

이 사이트의 내용을 읽고 공부하는 것이 **Turtle**에 대해 가장 자세히 아는 방법입니다. 그런데 영어네요. 가끔 프로그래머들에게 초중고생들이 처음 프로그래밍을 배우려고 하는데 어떤 언어를 배워야

하느냐? 라고 질문하는 경우가 있습니다. 그때 많이 나오는 답중에 하나가 "영어" 입니다.

프로그래밍을 잘하려면 영어공부도 잘해야 합니다. 그렇다면 수학, 국어는 어떨까요? 여러분의 생각을 말씀해보세요.