

7. 사례 연구: 함수 인터페이스

개요

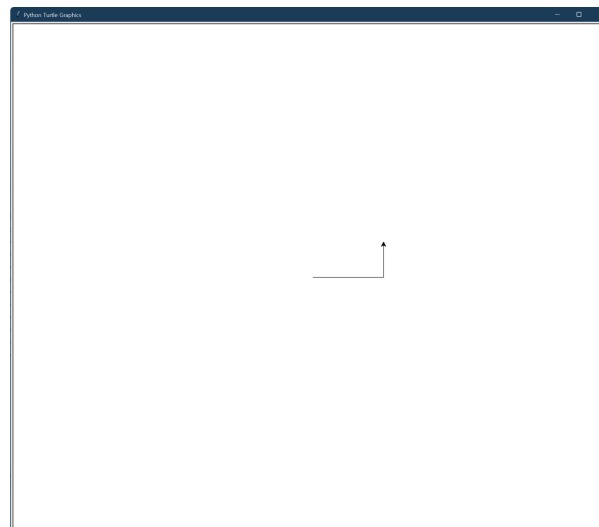
- `turtle` 모듈 활용: 2차원 컴퓨터 그래픽스 지원
- 반복문과 함수의 활용
- 객체 지향 프로그래밍(OOP) 맛보기

7.1. turtle 모듈

```
import turtle

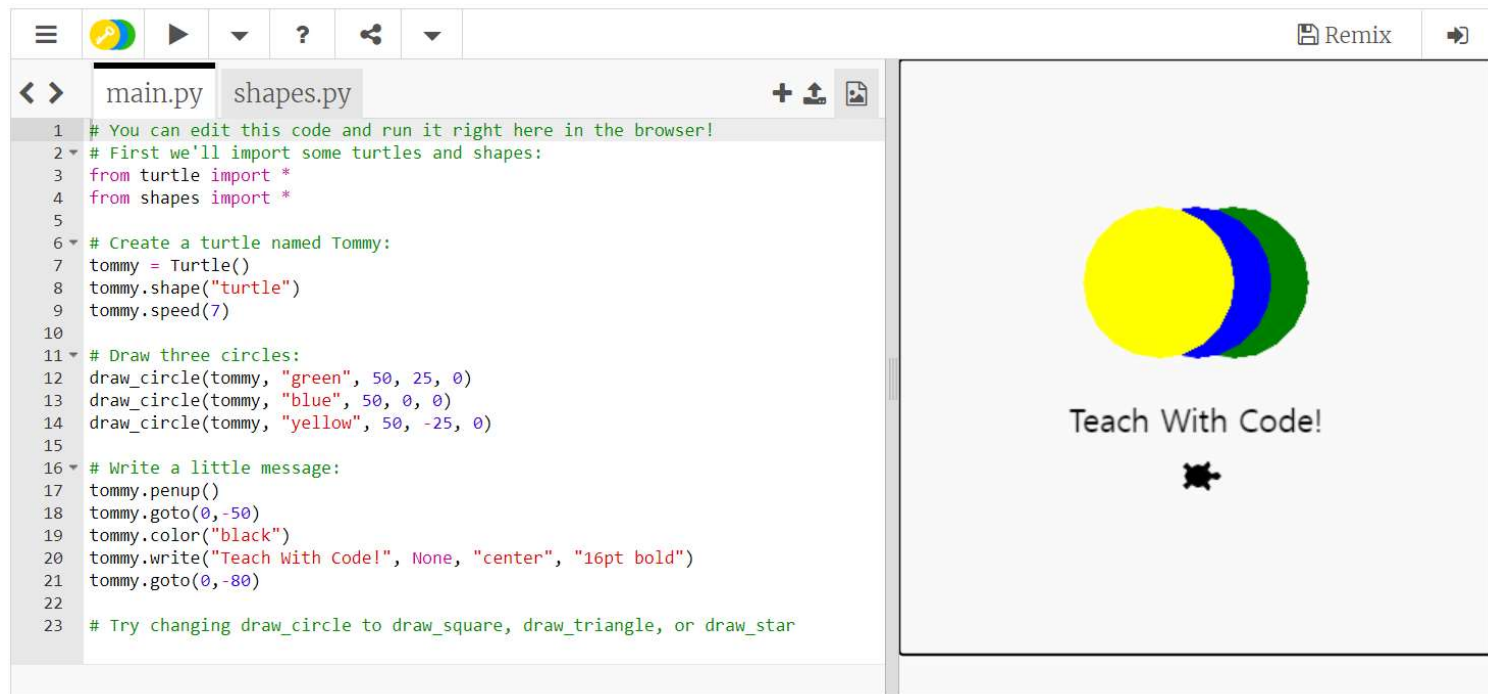
wn = turtle.Screen()
bob = turtle.Turtle()

bob.forward(150)
bob.left(90)
bob.forward(75)
```



Trinket 사이트 활용

- 구글 코랩 사용 불가
- [트링킷](#) Trinket이 거북이 그래픽스 지원



The screenshot displays the Trinket website interface, which is used for learning and practicing Python programming with the Turtle graphics library. The interface is divided into two main sections: a code editor on the left and a live preview window on the right.

Code Editor: The editor shows a Python script named `main.py` with the following code:

```
1 # You can edit this code and run it right here in the browser!
2 # First we'll import some turtles and shapes:
3 from turtle import *
4 from shapes import *
5
6 # Create a turtle named Tommy:
7 tommy = Turtle()
8 tommy.shape("turtle")
9 tommy.speed(7)
10
11 # Draw three circles:
12 draw_circle(tommy, "green", 50, 25, 0)
13 draw_circle(tommy, "blue", 50, 0, 0)
14 draw_circle(tommy, "yellow", 50, -25, 0)
15
16 # Write a little message:
17 tommy.penup()
18 tommy.goto(0, -50)
19 tommy.color("black")
20 tommy.write("Teach With Code!", None, "center", "16pt bold")
21 tommy.goto(0, -80)
22
23 # Try changing draw_circle to draw_square, draw_triangle, or draw_star
```

Live Preview: The right-hand window shows the output of the code. It features three overlapping circles: a yellow circle on the left, a blue circle in the middle, and a green circle on the right. Below the circles, the text "Teach With Code!" is displayed in a bold, black font. A small black turtle icon is positioned at the bottom center of the preview area.

캔버스와 거북이 객체

- `turtle.Screen()`: 캔버스 객체를 하나 생성. 하나만 생성 가능.
- `Turtle` 클래스: 거북이 객체(인스턴스) 생성. 여러 개 생성 가능.

거북이 객체 주요 메서드

메서드	단축어	의미
forward()	fd()	전진
left()	lt()	반시계방향 회전
backward()	bd()	후진
right()	rt()	시계방향 회전
pendown()	pd()/down()	펜 내리기
penup()	pu()/up()	펜 들기
showturtle()	st()	거북이 보이기
hideturtle()	ht()	거북이 숨기기

7.2. 반복문 활용

복불 보단 반복문!

- 복불

```
bob.fd(100)
bob.lt(90)
bob.fd(100)
bob.lt(90)
bob.fd(100)
bob.lt(90)
bob.fd(100)
bob.lt(90)
```

- for 반복문

```
for i in
range(4):
..... bob.fd(100)
..... bob.lt(90)
```

예제: 여러 개의 거북이 객체 활용

- 두 개의 Turtle 클래스의 인스턴스 사용
- 두 객체의 속성 구분 가능

```
import turtle

wn = turtle.Screen()
wn.bgcolor("lightyellow")
# wn.title("Hello, Bob and Alice!")

# bob 생성
bob = turtle.Turtle()

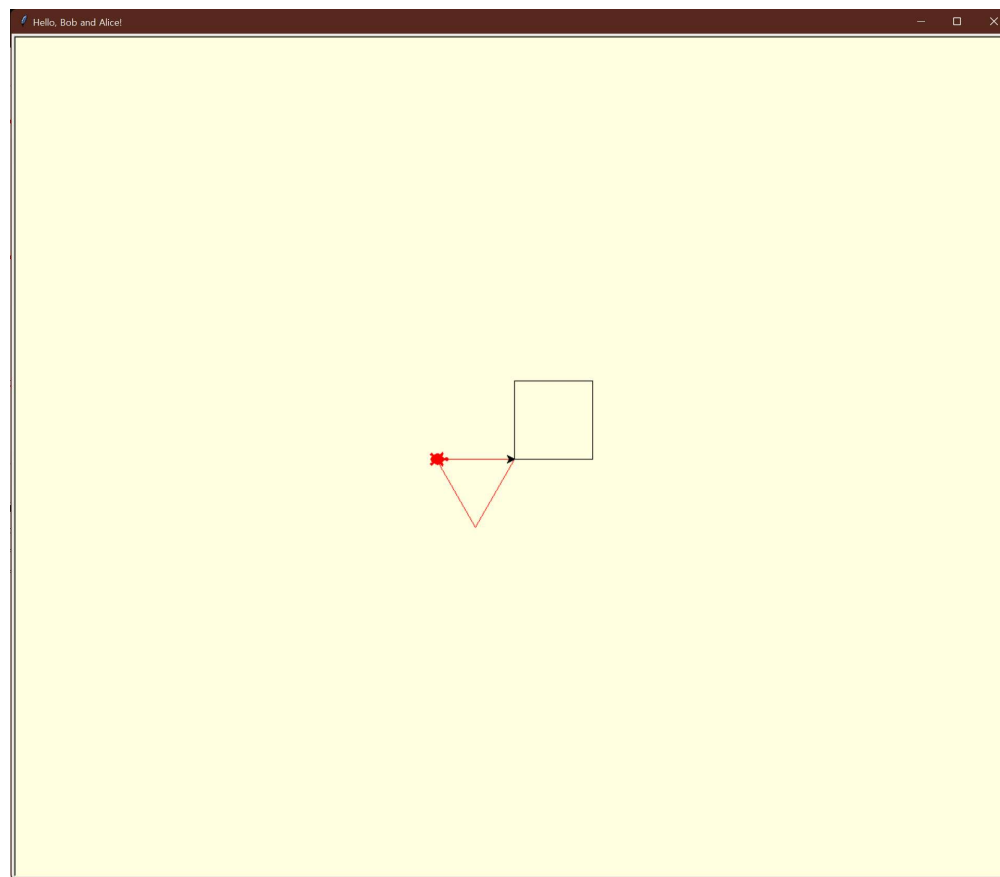
# alice 생성
alice = turtle.Turtle()
alice.shape("turtle")
alice.color("red")

alice.penup()
alice.backward(100)
alice.pendown()

# bob 으로 사각형 그리기
for i in range(4):
    bob.forward(100)
    bob.left(90)

# alice로 삼각형 그리기
for i in range(3):
    alice.forward(100)
    alice.right(120)

# wn.mainloop()
```



7.3. 함수화와 일반화

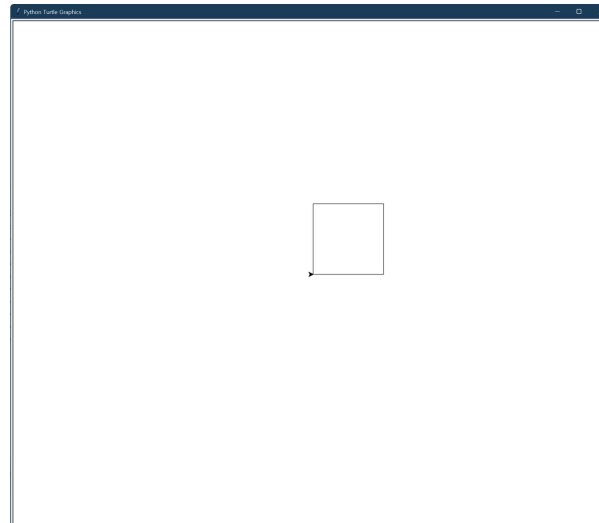
함수 구현 과정

- 함수를 사용하지 않는 간단한 프로그램 작성
- 함수화: 작성한 프로그램이 잘 작동하면 하나의 함수로 선언
- 일반화: 적절한 매개변수 `parameters`를 추가하여 함수의 기능 확장
- 원하는 수준의 함수를 얻을 때까지 위 과정 반복

함수화 예제

- 사각형을 그리는 거북이를 지정하는 함수

```
def square(t):  
    for i in range(4):  
        t.fd(100)  
        t.lt(90)  
  
square(bob)
```



일반화 예제: 길이 매개변수 추가

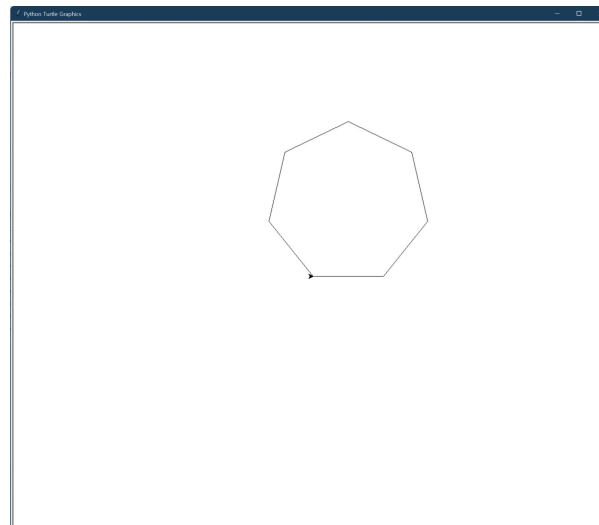
```
def square(t, length):  
    for i in range(4):  
        t.fd(length)  
        t.lt(90)
```

```
square(bob, 100)
```

```
square(bob, 50)
```


함수화 예제: 다각형 그리기

```
def polygon(t, n, length):  
    angle = 360 / n  
    for i in range(n):  
        t.fd(length)  
        t.lt(angle)  
  
polygon(bob, 7, 70)
```



함수화 예제: 키워드 인자 활용

```
def polygon(t, n=7, length=70):  
    ... angle = 360 / n  
  
    ... for i in range(n):  
        ... t.fd(length)  
        ... t.lt(angle)
```

```
polygon(bob)
```

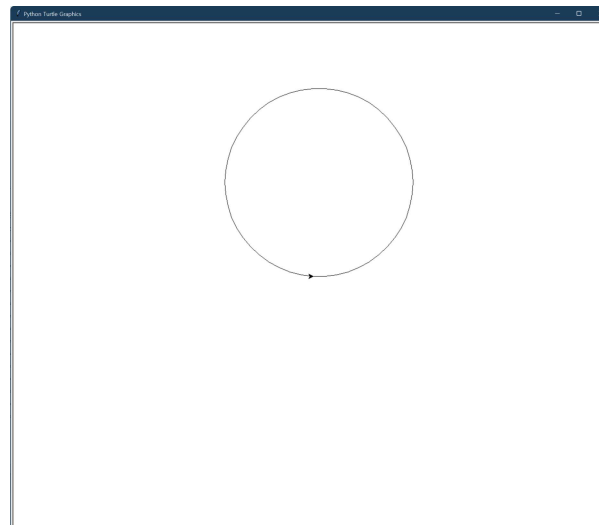
7.4. 함수 인터페이스

함수 사용법 요약문

- 어떤 매개변수 사용?
- 함수의 기능, 즉 함수가 하는 일은?
- 함수의 반환값은?

예제: 원의 크기가 반지름에 의존

```
def circle(t, r):  
    ... circumference = 2 * math.pi * r  
    ... n = 50  
    ... length = circumference / n  
    ... polygon(t, n, length)  
  
circle(bob, 200)
```

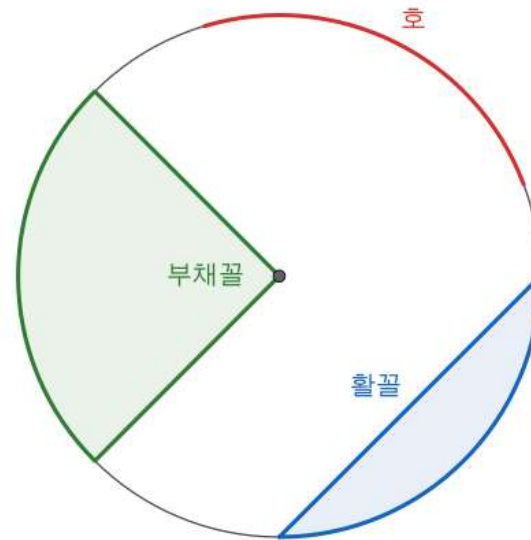


원의 크기 일정하게 유지하기

```
def circle(t, r):  
    ... circumference = 2 * math.pi * r  
  
    ... n = int(circumference / 3) + 3 # 원둘레의 1/3 정도에 해당하는 선분 수  
    ... length = circumference / n ... # 약 3 정도의 값  
  
    ... polygon(t, n, length)  
  
    ... return n
```

7.5. 리팩토링: 코드 재구성

원과 호



- 호를 새로 구현

```
def arc(t, r, angle):  
    arc_length = 2 * math.pi * r * angle / 360  
    n = int(arc_length / 3) + 1  
    step_length = arc_length / n  
    step_angle = angle / n  
  
    for i in range(n):  
        t.fd(step_length)  
        t.lt(step_angle)
```


보조 함수 활용

```
def polyline(t, n, length, angle):
    for i in range(n):
        t.fd(length)
        t.lt(angle)

def polygon(t, n, length):
    angle = 360.0 / n

    polyline(t, n, length, angle)

def arc(t, r, angle):
    arc_length = 2 * math.pi * r * angle / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n

    polyline(t, n, step_length, step_angle)

def circle(t, r):
    arc(t, r, 360)
```

7.6. 독스트링: 문서화 문자열

```
def polyline(t, n, length, angle):  
    """t: 거북이 객체  
    n: 선분 그리기 반복 횟수  
    length: 선분의 길이  
    angle: 회전 각도. 즉, 두 선분 사이의 각도  
  
    지정된 크기와 각도를 이용하여 거북이 t 가 n 개의 선분을 그린다.  
    """  
    for i in range(n):  
        t.fd(length)  
        t.lt(angle)
```

```
help(polyline)
```

```
Help on fuction polyline in module __main__:
```

```
polyline(t, n, length, angle)  
t: 거북이 객체  
n: 선분 그리기 반복 횟수  
length: 선분의 길이  
angle: 회전 각도. 즉, 두 선분 사이의 각도
```

```
지정된 크기와 각도를 이용하여 거북이 t 가 n 개의 선분을 그린다.
```