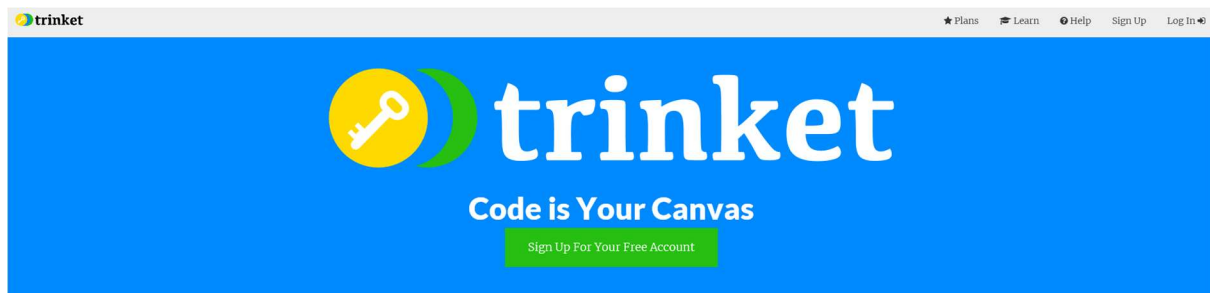


Trinket 시작

<https://trinket.io/> 에 접속합니다.

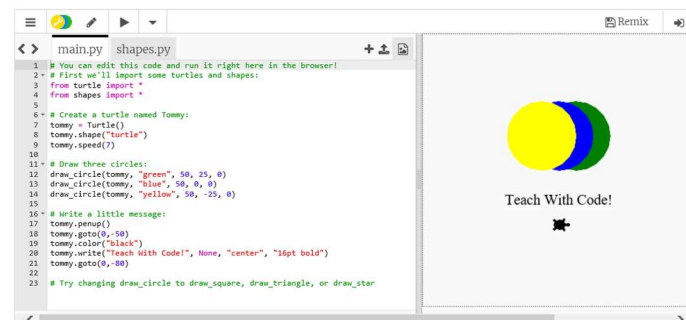


Share Code from any Device

Trinket lets you run and write code in any browser, on any device.

Trinkets work instantly, with no need to log in, download plugins, or install software.

Easily share or embed the code with your changes when you're done.

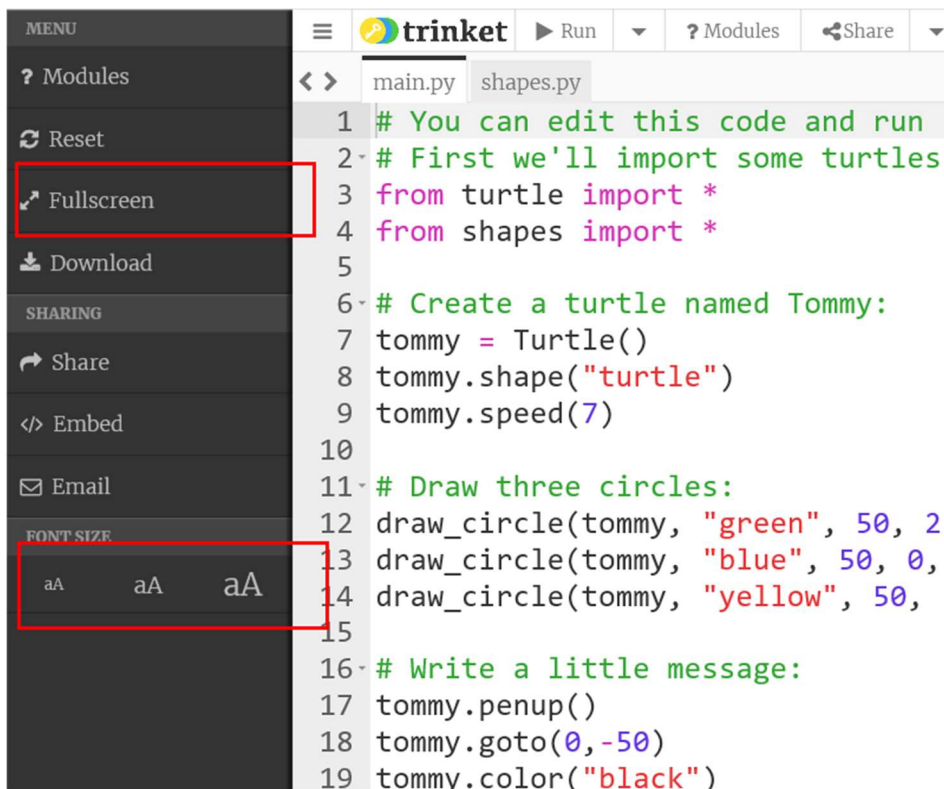


좌측상단의 메뉴버튼을 클릭하면 에디터 설정을 변경할 수 있다.

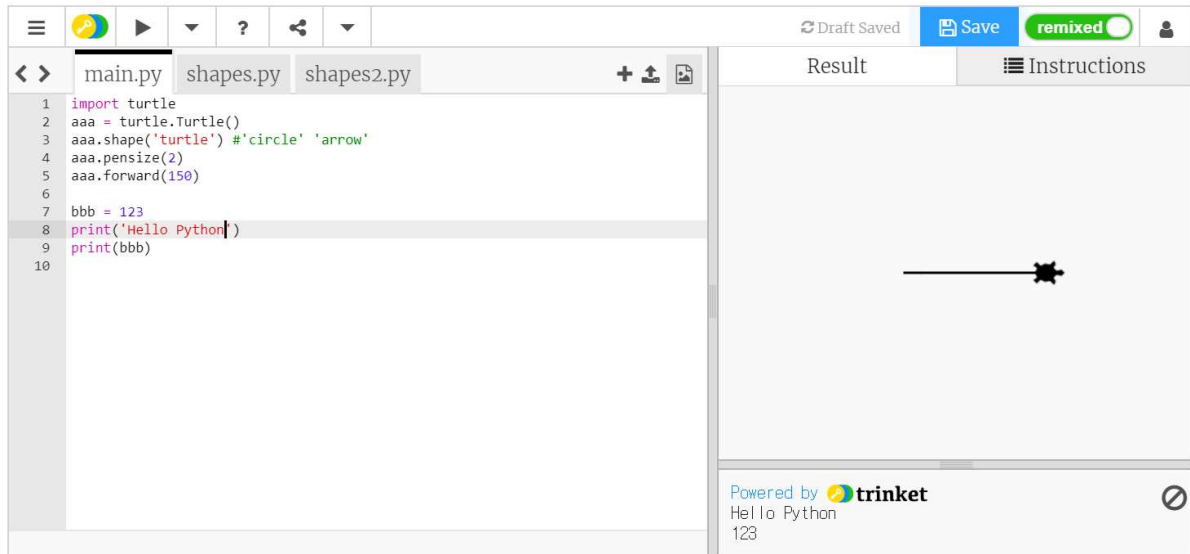
Turtle 시작



전체화면변경과 폰트사이즈 조정은 가장 많이 사용하는 메뉴이다



Turtle 시작



```
import turtle

aaa = turtle.Turtle()

aaa.shape('turtle') # 'circle' 'arrow'

aaa.pensize(2)

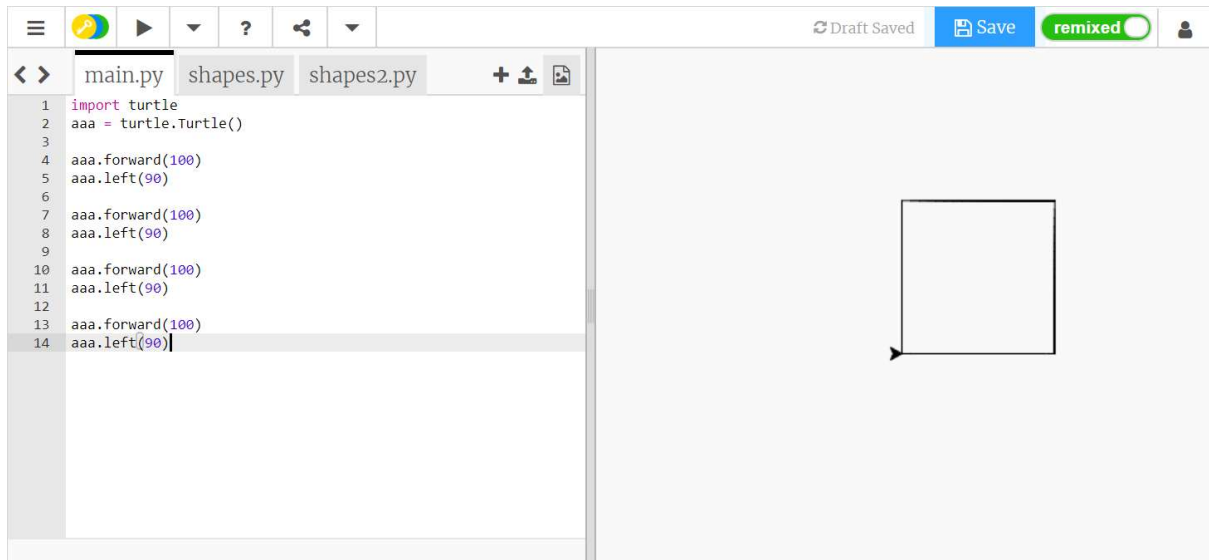
aaa.forward(150)


bbb = 123

print('Hello Python')

print(bbb)
```

다각형 그리기



```
import turtle
```

```
aaa = turtle.Turtle()
```

```
aaa.forward(100)
```

```
aaa.left(90)
```

```
aaa.forward(100)
```

```
aaa.left(90)
```

```
aaa.forward(100)
```

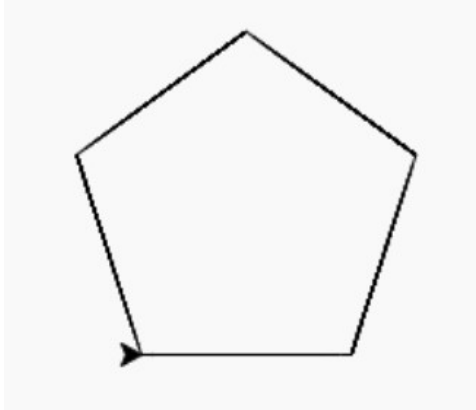
```
aaa.left(90)
```

```
aaa.forward(100)
```

```
aaa.left(90)
```

1. 문제

4각형 이외에 N각형을 그리기 위해서는 코드의 어느 부분이 수정되어야 하는지 확인해보자.

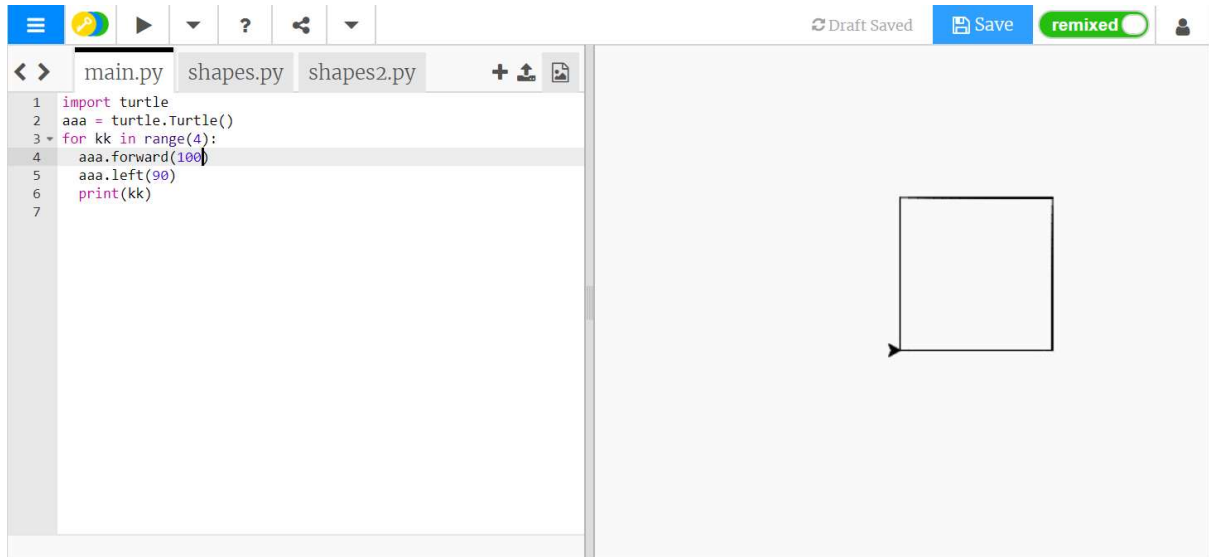


```
import turtle  
aaa= turtle.Turtle()  
aaa.forward(100)  
aaa.left(360/5)  
aaa.forward(100)  
aaa.left(360/5)  
aaa.forward(100)  
aaa.left(360/5)  
aaa.forward(100)  
aaa.left(360/5)  
aaa.forward(100)  
aaa.left(360/5)
```

반복문 조건문 사용

◆ 디버깅 기술의 이해

사각형을 그려보자



변수들의 움직임을 관찰하면서 패턴을 발견하고 알고리즘을 수정할 수 있다.

```
import turtle

aaa = turtle.Turtle()

for kk in range(4):

    aaa.forward(150)

    aaa.left(90)

    print(kk)
```

◆ 선의 색을 바꾸는 방법은?

선의 색을 바꿔보는 작업을 진행해보자..

선 색을 바꾸기 위해서는 어떤 함수(기능)이 필요할까? 파이썬에서는 수많은 함수를 제공한다. 이

Turtle 시작

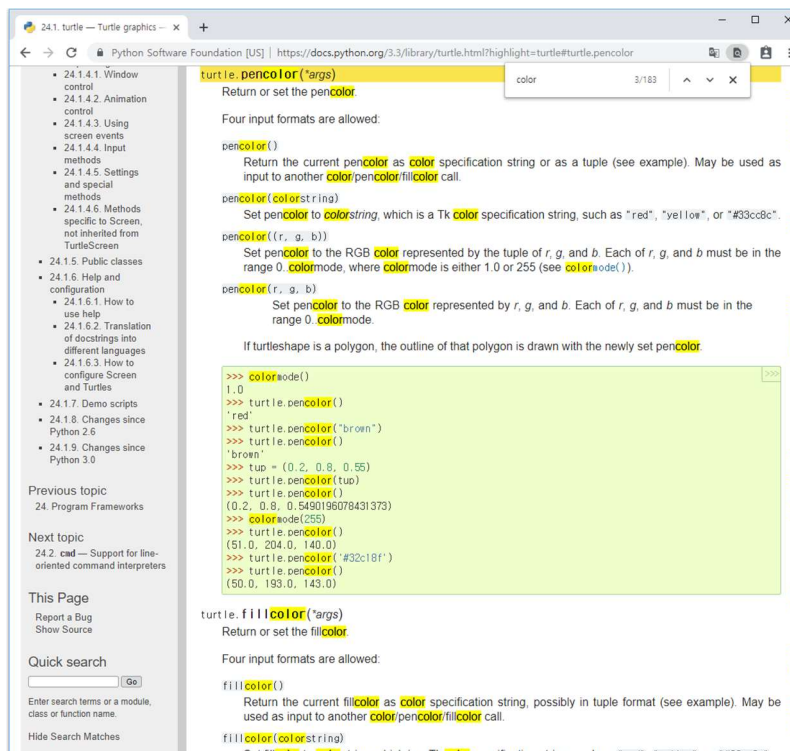
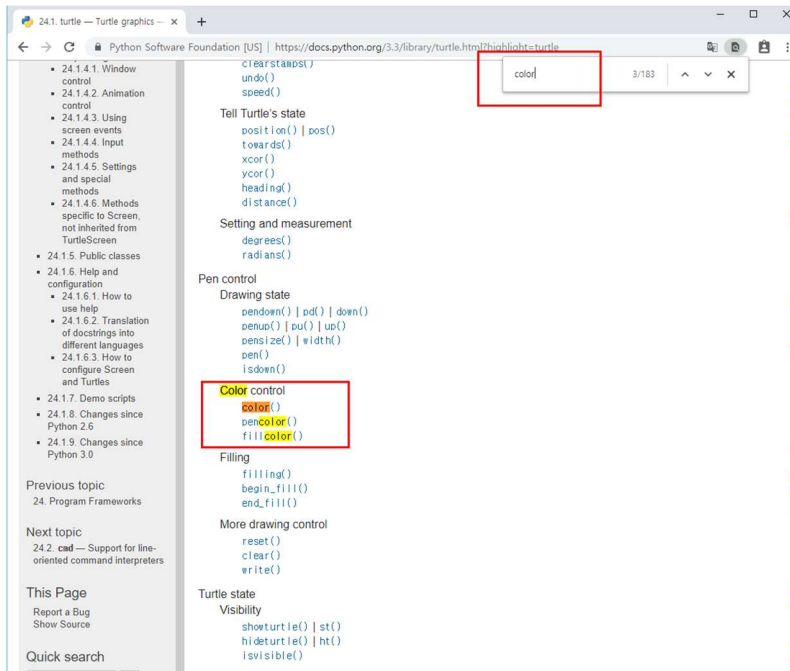
것을 모두 암기할 수는 없다.

따라서 가장 사용하기 적합한 함수를 찾는 것이 프로그래밍 실력에 매우 중요한 능력이 된다.

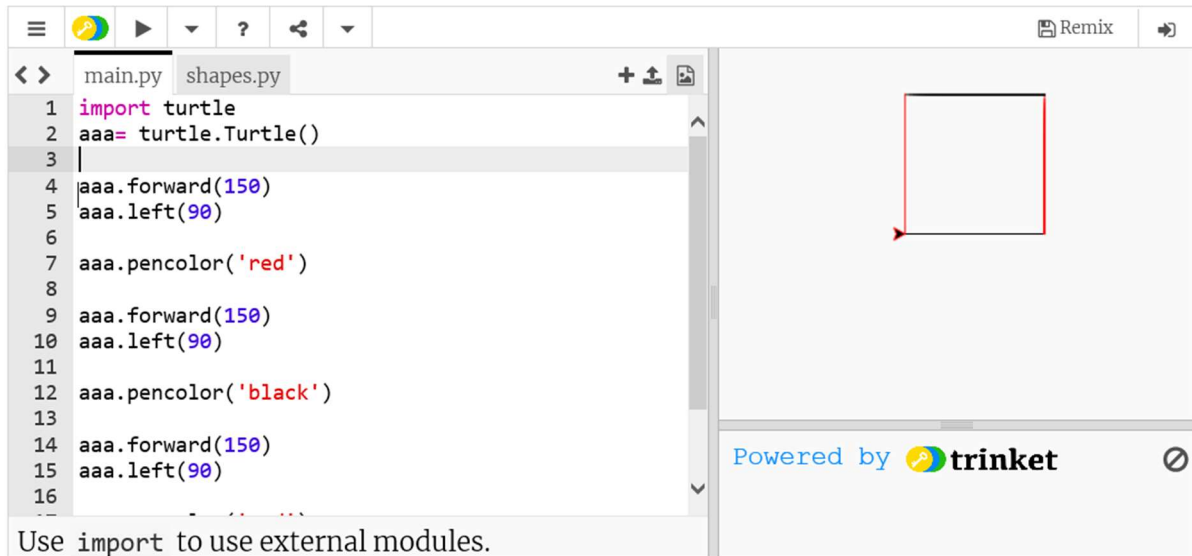
구글검색의 도움을 받아보자.



Turtle 시작



Turtle 시작



```
import turtle
```

```
aaa= turtle.Turtle()
```

```
aaa.forward(150)
```

```
aaa.left(90)
```

```
aaa.pencolor('red')
```

```
aaa.forward(150)
```

```
aaa.left(90)
```

```
aaa.pencolor('black')
```

```
aaa.forward(150)
```

```
aaa.left(90)
```

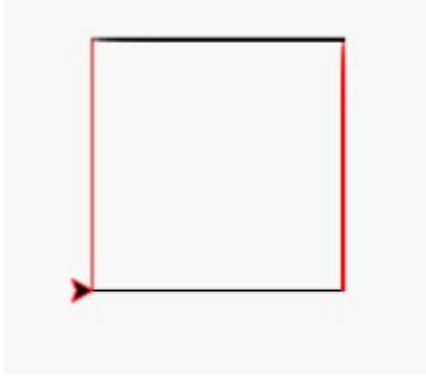
```
aaa.pencolor('red')
```

```
aaa.forward(150)
```

```
aaa.left(90)
```

2. 문제

다음 그림과 같이 4각형의 세로 방향을 빨강색을 이용해서 표현해보자.



- 1) 조건문에 논리 연산자를 이용해 보시오
- 2) 드모르강법칙을 이용해보시오
- 3) 단항연산자 not 을 이용해보시오

```
import turtle

aaa = turtle.Turtle()

#1)if kk == 1 or kk == 3:

#2)if not(kk != 1 and kk != 3): #DeMorgan (논리식)

#3-1)if kk%2 == 1:

#3-2)if kk%2:

#4-1)if kk%2 == 0:

#4-2)if not kk%2 == 0:

for kk in range(4):

    if _____

        aaa.pencolor('red')

    else:

        aaa.pencolor('black')

    aaa.forward(150)
```

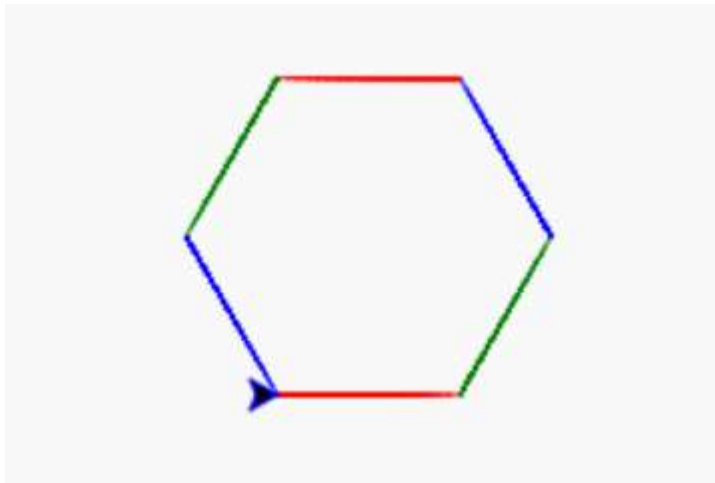
aaa.left(90)

print(kk)

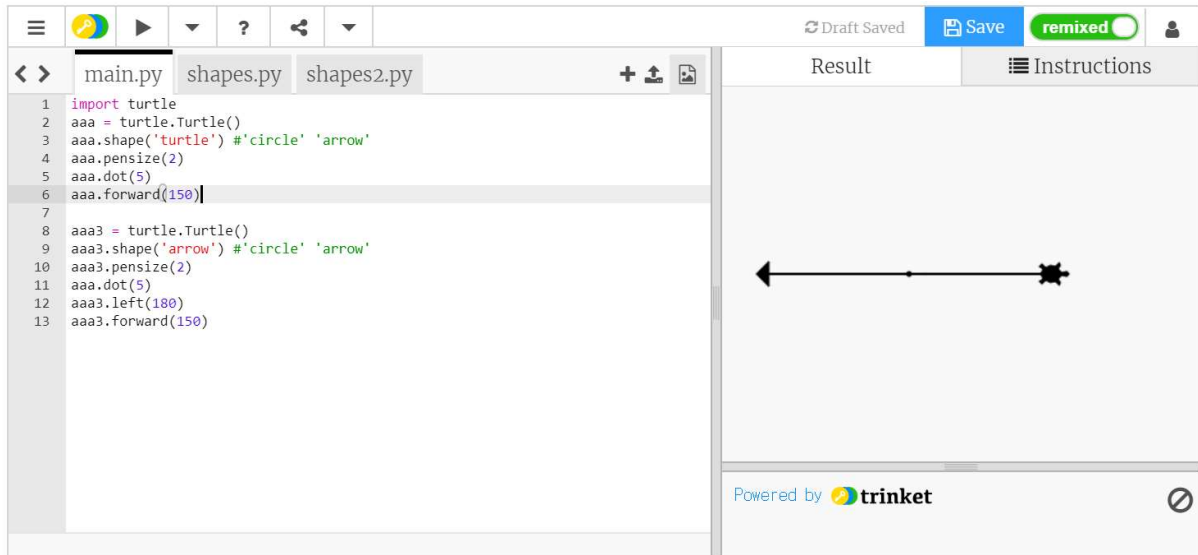
방법	코드
열거식	if kk == 1 or kk == 3:
열거식 (DeMorgan)	if not(kk != 1 and kk != 3):
산술식	if kk%2 == 1:
산술식 (조건식 축약)	if kk%2:
산술식	if kk%2 == 0:
산술식 (단항연산자)	if not kk%2 == 0:

◆ HW

다음 그림과 같이 6각형의 마주보는 변의 색을 같은 색으로 표현해보자. (반복문, 조건문 이용)



새로운 객체 변수 추가



```
import turtle
```

```
aaa = turtle.Turtle()
```

```
aaa.shape('turtle') #'circle' 'arrow'
```

```
aaa.pensize(2)
```

```
aaa.dot(5)
```

```
aaa.forward(150)
```

```
aaa3 = turtle.Turtle()
```

```
aaa3.shape('arrow') #'circle' 'arrow'
```

```
aaa3.pensize(2)
```

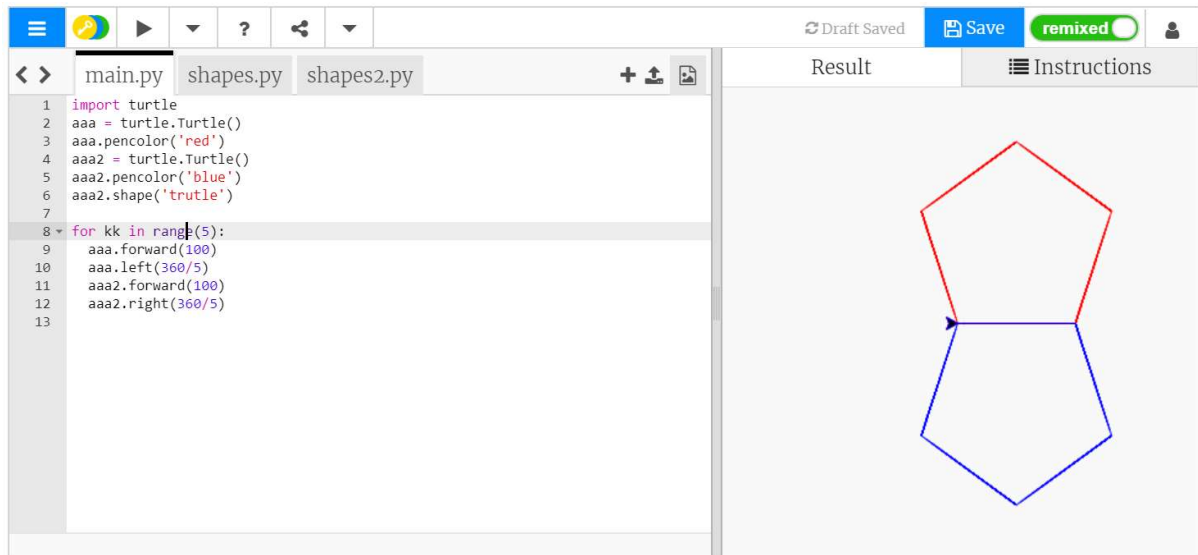
```
aaa3.dot(5)
```

```
aaa3.left(180)
```

```
aaa3.forward(150)
```

3. 문제

그림과 같이 5각형 별집 모양을 두개의 터틀을 이용해서 그려보자. (반복문사용)



```
import turtle
```

```
aaa = turtle.Turtle()
```

```
aaa.pencolor('red')
```

```
aaa2 = turtle.Turtle()
```

```
aaa2.pencolor('blue')
```

```
aaa2.shape('turtle')
```

```
for kk in range(5):
```

```
    aaa.forward(100)
```

```
    aaa.left(360/5)
```

```
    aaa2.forward(100)
```

```
    aaa2.right(360/5)
```

함수 만들기

4. 문제

다음 그림과 같이 다각형을 만드는 함수 (MyPolygon) 를 작성하시오.



```
import turtle  
  
aaa = turtle.Turtle()  
  
def MyPolygon(cnt,length):  
    for kk in range(cnt):  
        aaa.forward(length)  
        aaa.left(360/cnt)  
  
MyPolygon (7,80)  
MyPolygon (5,80)
```

5. 문제

아래 그림과 같이 5각별을 만들고자 한다. 프로그램을 완성하시오.

- 각도 144도 유도과정을 설명하시오.
- 한 붓 그리기가 가능한 경우는 어떤 경우인지 설명하시오.



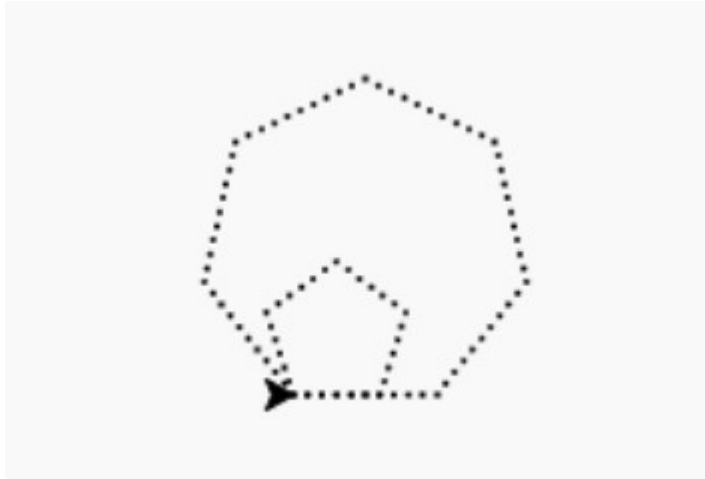
```
import turtle
import math
aaa = turtle.Turtle()

def MyStar(length):
    for kk in range(5):
        aaa.forward(length)
        aaa.left(144)

MyStar(88)
```

◆ HW

다음 그림과 같이 다각형 선을 점으로 표현하는 함수 (DotPolygon)를 만드시오.



호출 예)

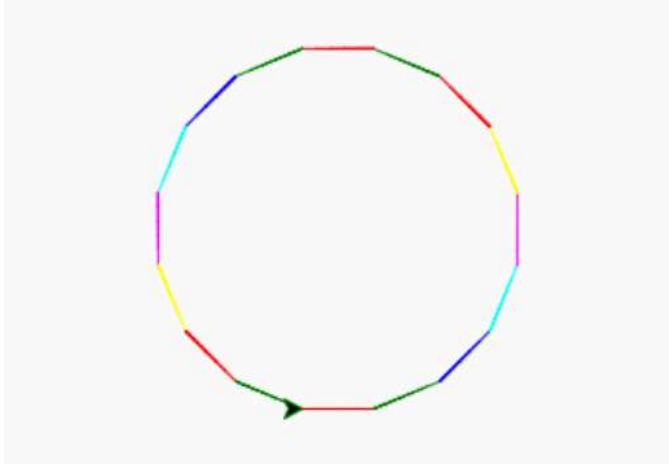
```
DotPolygon(7,80)
```

```
DotPolygon(5,80)
```

◆ HW

다음 그림과 같이 N각형의 마주보는 변의 색을 같은 색으로 표현해보자.

- 색정보를 리스트를 담아서 사용할 것
- 인접한 색이 연속해서 출력되지 않도록 할 것
- 마주보는 면이 없는 경우는 흑백으로 출력 하도록 할 것

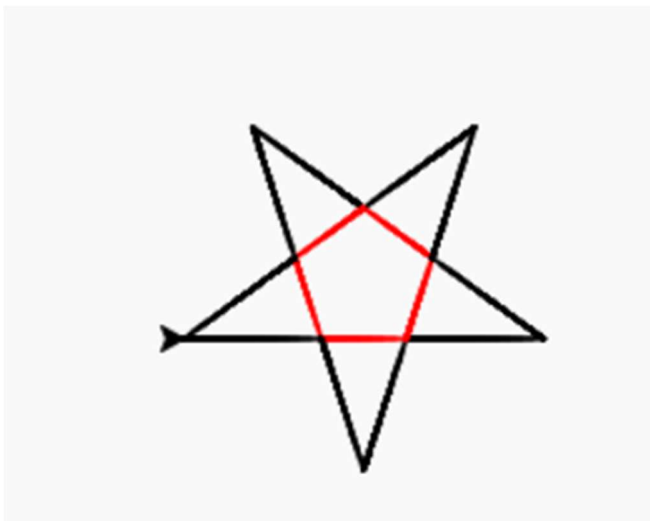


◆ HW

아래 그림과 같이 정오각형을 품는 별모양을 한 붓 그리기 하시오.

- 오각형은 빨간색으로 표시되어야한다.
- 선의 두께를 2로 조정하시오
- forward 길이는 임의 값을 적용 가능 하도록 하시오.

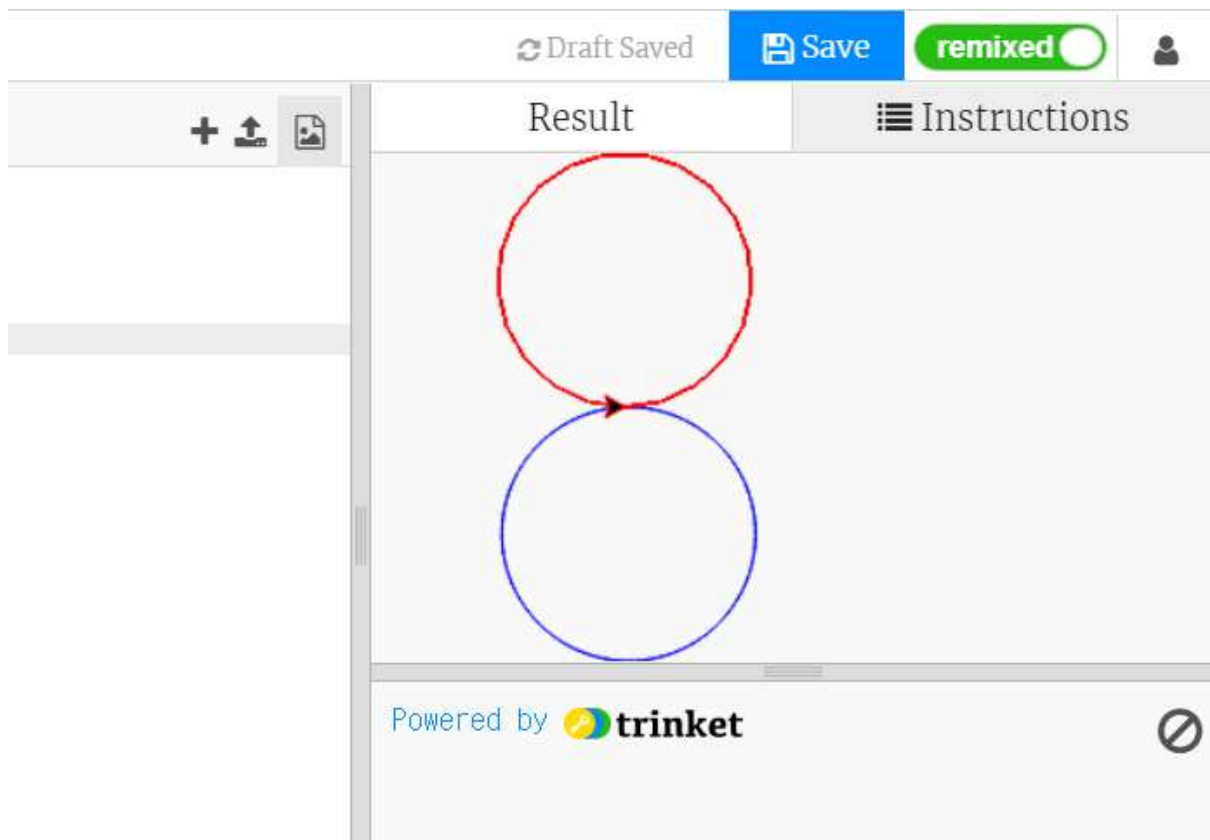
- 이중 for 문을 사용할 수 있는가?
- 관계식을 유도할 수 있는가?



◆ HW

아래 그림은 반지름 100이 주어졌을 때 원을 그리는 함수의 비교이다. Turtle 의 circle 과 같은 함수 MyCircle() 을 완성하시오

- 펜의 색은 파란색을 사용하시오.
- (1) 과 같이 반지름값을 입력값으로 사용할 것



```
import turtle
import math
aaa = turtle.Turtle()
# MyCircle() 함수를 구현하시오.
MyCircle(100)
aaa.pencolor('red')
aaa.circle(100,360)  #(1) 반지름, 각도
```

기본 데이터 관리 방법

6. 문제

아래 <변경전> 은 `math.radians` 함수를 이용해서 degree 값을 radian 값으로 치환하고, 이를 이용해서 sin 값을 degree, radian 과 함께 출력하는 프로그램이다. <변경후> 와 같이 출력품을 바꿔보자.

The screenshot shows the Trinket IDE interface. On the left, the code in `MyAxis.py` is as follows:

```
1 import math
2
3 for kk in range(0,361,30):
4     rad = math.radians(kk)
5     sinval = math.sin(rad)
6
7     print(kk, rad, sinval)
8
```

On the right, the 'Result' panel shows the output of the program:

```
Powered by trinket
(0, 0.0, 0.0)
(30, 0.523598775598, 0.5)
(60, 1.0471975512, 0.866025403784)
(90, 1.57079632679, 1.0)
(120, 2.09439510239, 0.866025403784)
(150, 2.61799387799, 0.5)
(180, 3.14159265359, 1.22464679915e-16)
(210, 3.66519142919, -0.5)
(240, 4.18879020479, -0.866025403784)
(270, 4.71238898038, -1.0)
(300, 5.23598775598, -0.866025403784)
(330, 5.75958653158, -0.5)
(360, 6.28318530718, -2.44929359829e-16)
```

<변경후>

The screenshot shows the Trinket IDE interface with the modified code. On the left, the code in `MyAxis.py` is as follows:

```
1 import math
2
3 for kk in range(0,361,30):
4     rad = math.radians(kk)
5     sinval = math.sin(rad)
6
7     #print(kk, rad, sinval)
8     print('[deg]%4d [rad]%7.4f [sin()%7.4f'%(kk,rad,sinval))
9
10
11
12
```

On the right, the 'Result' panel shows the output of the program:

```
Powered by trinket
[deg] 0 [rad] 0.0000 [sin()] 0.0000
[deg] 30 [rad] 0.5236 [sin()] 0.5000
[deg] 60 [rad] 1.0472 [sin()] 0.8660
[deg] 90 [rad] 1.5708 [sin()] 1.0000
[deg] 120 [rad] 2.0944 [sin()] 0.8660
[deg] 150 [rad] 2.6180 [sin()] 0.5000
[deg] 180 [rad] 3.1416 [sin()] 0.0000
[deg] 210 [rad] 3.6652 [sin()] -0.5000
[deg] 240 [rad] 4.1888 [sin()] -0.8660
[deg] 270 [rad] 4.7124 [sin()] -1.0000
[deg] 300 [rad] 5.2360 [sin()] -0.8660
[deg] 330 [rad] 5.7596 [sin()] -0.5000
[deg] 360 [rad] 6.2832 [sin()] -0.0000
```

`import math`

```
for kk in range(0,361,30):
```

```
    rad = math.radians(kk)
```

```
    sinval = math.sin(rad)
```

```
    #print(kk, rad, sinval)
```

```
    #print('[deg]%4d [rad]%7.4f [sin()]%7.4f'%(kk,rad,sinval))
```

```
    print('[deg]{0:>4} [rad]{1:>7.4} [sin()]{2:>7.4}'.format(kk,rad,sinval))
```

◆ HW

아래 프로그램은 `math.radians` 함수를 이용해서 degree 값을 radian 값으로 치환하고 `math.degrees` 함수를 이용해서 다시 역으로 치환하는 프로그램이다.

`radians` 과 `degrees` 함수와 같은 기능을 하는 `MyRadians` 과 `MyDegrees` 함수를 만들어보자.

```
1 import math
2
3 for kk in range(0,361,30):
4     rad = math.radians(kk)
5     deg = math.degrees(rad)
6
7     print('[deg]{0:>4} [rad]{1:>7.4} [deg\']{2:>9.4}'.format(kk,rad,deg))
8
```

Result

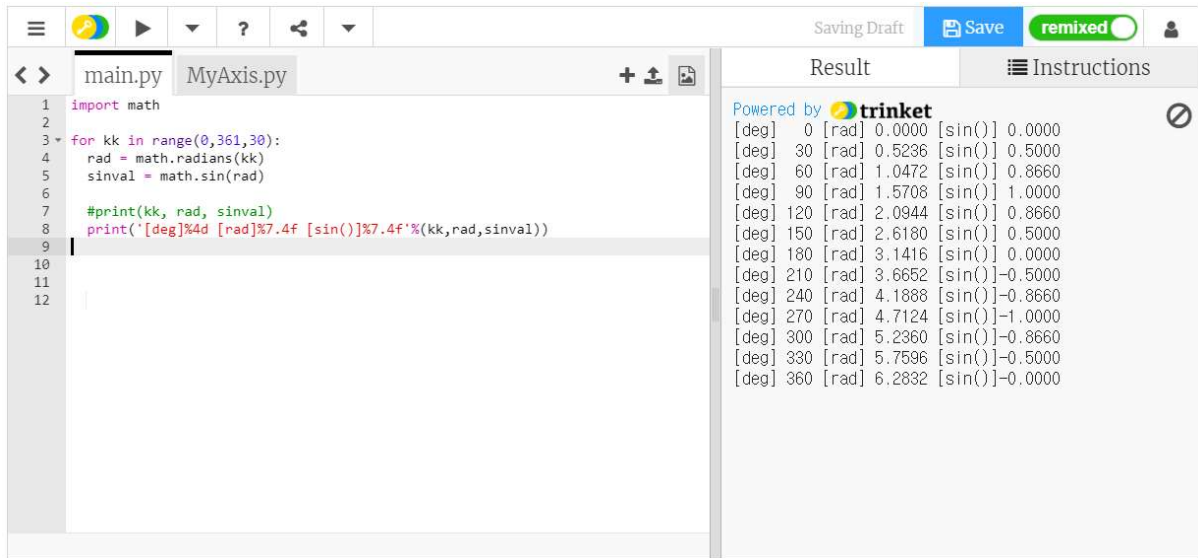
Powered by trinket

[deg]	0	[rad]	0.0000	[deg']	0.0000
[deg]	30	[rad]	0.5236	[deg']	30.0000
[deg]	60	[rad]	1.0472	[deg']	60.0000
[deg]	90	[rad]	1.5708	[deg']	90.0000
[deg]	120	[rad]	2.0944	[deg']	120.0000
[deg]	150	[rad]	2.6180	[deg']	150.0000
[deg]	180	[rad]	3.1416	[deg']	180.0000
[deg]	210	[rad]	3.6652	[deg']	210.0000
[deg]	240	[rad]	4.1888	[deg']	240.0000
[deg]	270	[rad]	4.7124	[deg']	270.0000
[deg]	300	[rad]	5.2360	[deg']	300.0000
[deg]	330	[rad]	5.7596	[deg']	330.0000
[deg]	360	[rad]	6.2832	[deg']	360.0000

7. 문제

아래 프로그램과 같이 `deg`, `radian`, `sin` 값을 3개의 리스트를 이용해서 각각 저장하고 저장된 리스트에서 값을 출력해보는 프로그램을 작성해보자.

Turtle 시작



The screenshot shows a Trinket.io code editor with a file named 'MyAxis.py'. The code is as follows:

```
1 import math
2
3 for kk in range(0,361,30):
4     rad = math.radians(kk)
5     sinval = math.sin(rad)
6
7     #print(kk, rad, sinval)
8     print('[deg]%4d [rad]%7.4f [sin()]%7.4f'%(kk,rad,sinval))
9
10
11
12
```

The output window on the right shows the results of the script, which are the same as the code's print statements:

[deg]	[rad]	[sin()]
0	0.0000	0.0000
30	0.5236	0.5000
60	1.0472	0.8660
90	1.5708	1.0000
120	2.0944	0.8660
150	2.6180	0.5000
180	3.1416	0.0000
210	3.6652	-0.5000
240	4.1888	-0.8660
270	4.7124	-1.0000
300	5.2360	-0.8660
330	5.7596	-0.5000
360	6.2832	-0.0000

```
import math

lt_deg = []
lt_rad = []
lt_sinval = []

for kk in range(0,361,30):

    rad = math.radians(kk)

    sinval = math.sin(rad)

    # print('[deg]{0:>4} [rad]{1:>7.4} [sin()]{2:>7.4}'.format(kk,rad,sinval))

    lt_deg.append(kk)
    lt_rad.append(rad)
    lt_sinval.append(sinval)

for kk in range(len(lt_deg)):
```

```
print('{0:>4} {1:>7.4} {sin()}{2:>7.4}'.format(lt_deg[kk],lt_rad[kk],lt_sinval[kk]))
```

8. 문제

아래 프로그램과 같이 deg, radian, sin 값을 1개의 리스트를 이용해서 함께 저장하고 이를 출력해 보는 프로그램을 작성해보자.

The screenshot shows a Trinket.io code editor with a Python script in a file named 'main.py'. The script imports the 'math' module and uses a 'for' loop to iterate over angles from 0 to 360 in increments of 30. For each angle 'kk', it calculates the radian value 'rad' using 'math.radians(kk)' and the sine value 'sinval' using 'math.sin(rad)'. It then prints the values in a formatted string. The 'Result' panel on the right shows the output of the program, displaying the angle in degrees, radians, and the sine value for each iteration.

deg	rad	sin()
0	0.0000	0.0000
30	0.5236	0.5000
60	1.0472	0.8660
90	1.5708	1.0000
120	2.0944	0.8660
150	2.6180	0.5000
180	3.1416	0.0000
210	3.6652	-0.5000
240	4.1888	-0.8660
270	4.7124	-1.0000
300	5.2360	-0.8660
330	5.7596	-0.5000
360	6.2832	0.0000

```
import math

lt_pack = []

for kk in range(0,361,30):

    rad = math.radians(kk)

    sinval = math.sin(rad)

    # print('{0:>4} {1:>7.4} {sin()}{2:>7.4}'.format(kk,rad,sinval))

    lt_pack.append([kk,rad,sinval])
```

```
for kk in range(len(lt_pack)):
    print('[deg]{0:>4} [rad]{1:>7.4} [sin()]{2:>7.4}'W
        .format(lt_pack[kk][0],lt_pack[kk][1],lt_pack[kk][2]))
```

◆ HW (검색기능 추가)

아래 프로그램과 같이 deg, radian, sin 값을 함께 저장한 리스트에서 특정 degree 값을 input() 을 이용해서 입력을 받고, radian, sin 정보를 호출하는 프로그램을 추가 하시오

```
import math

lt_pack = []

for kk in range(0,361,30):
    rad = math.radians(kk)
    sinval = math.sin(rad)

    print('[deg]{0:>4} [rad]{1:>7.4} [sin()]{2:>7.4}'.format(kk,rad,sinval))

    lt_pack.append([kk,rad,sinval])
```

9. 문제 (Dictionary 컬렉션 타입을 이용)

아래 프로그램과 같이 deg, radian, sin 값을 1개의 딕셔너리를 이용해서 함께 저장하고 이를 출력 해보는 프로그램을 작성해보자.

```
import math
```

```
dic = {}

for kk in range(0,361,30):
    rad = math.radians(kk)
    sinval = math.sin(rad)

    dic[kk] =[rad,sinval]

for kk,vv in dic.items():
    print('[deg]{0:>4} [rad]{1:>7.4} [sin()]{2:>7.4}'W
        .format(kk,vv[0],vv[1]))
```

◆ HW (검색기능 추가)

아래 프로그램과 같이 deg, radian, sin 값을 함께 저장한 딕셔너리에서 특정 degree 값을 input()을 이용해서 입력을 받고, radian, sin 정보를 호출하는 프로그램을 추가 하시오

- For 문을 이용한 검색
- 딕셔너리의 get() 함수를 이용한 검색

Turtle 시작



The screenshot shows a Trinket Python IDE interface. The left pane displays a Python script named 'main.py' with the following code:

```
1 import math
2
3 dic = {}
4
5 for kk in range(0,361,30):
6     rad = math.radians(kk)
7     sinval = math.sin(rad)
8
9     dic[kk] =[rad,sinval]
10
11 for kk,vv in dic.items():
12     print('[deg]{0:>4} [rad]{1:>7.4} [sin()]{2:>7.4}'\
13           .format(kk,vv[0],vv[1]))
14
15
16 deg = int(input('degree?'))
17
18 for kk,vv in dic.items():
19     if deg == kk:
20         print('[deg]{0:>4} [rad]{1:>7.4} [sin()]{2:>7.4}'\
21               .format(kk,vv[0],vv[1]))
22
```

The right pane shows the 'Result' tab with the following output:

```
Powered by trinket
[deg] 0 [rad] 0.0000 [sin()] 0.0000
[deg] 30 [rad] 0.5236 [sin()] 0.5000
[deg] 60 [rad] 1.0472 [sin()] 0.8660
[deg] 90 [rad] 1.5708 [sin()] 1.0000
[deg] 120 [rad] 2.0944 [sin()] 0.8660
[deg] 150 [rad] 2.6180 [sin()] 0.5000
[deg] 180 [rad] 3.1416 [sin()] 0.0000
[deg] 210 [rad] 3.6652 [sin()] -0.5000
[deg] 240 [rad] 4.1888 [sin()] -0.8660
[deg] 270 [rad] 4.7124 [sin()] -1.0000
[deg] 300 [rad] 5.2360 [sin()] -0.8660
[deg] 330 [rad] 5.7596 [sin()] -0.5000
[deg] 360 [rad] 6.2832 [sin()] -0.0000
degree? 90
[deg] 90 [rad] 1.5708 [sin()] 1.0000
```

```
import math
```

```
dic = {}
```

```
for kk in range(0,361,30):
```

```
    rad = math.radians(kk)
```

```
    sinval = math.sin(rad)
```

```
    dic[kk] =[rad,sinval]
```

수치해석 방법 (Numerical Method) VS. Numpy 방법

10.문제

x,y 2차원 좌표평면에서, 점 (-2,1) 과 점 (2,9) 이 이루는 직선과 x 축 사이의 면적을 구하시오.

- 1) 수치해석 방법 (Numerical Method)
- 2) Numpy 이용방법 (Broadcasting, Vectorization)

[정답] 약 20

◆ HW

0~2PI 사이의 $\sin()$ 함수와 x 축 사이의 면적을 구하시오. (음의면적 또한 양의 면적으로 계산하시오)

- math.radians 함수를 사용하지 말 것
- 1) 수치해석 방법 (Numerical Method) -- Stop Condition
 - 2) Numpy 이용방법 (Broadcasting, Vectorization)
 - ~~3) List Comprehension 방법~~

[정답] 약 4

◆ HW

길이 15m 의 끈이 있다. 한쪽 벽면을 한변으로, 직사각형 울타리를 만들고자 한다.

울타리 내부 면적이 최대가 될 때, 벽면과 평행인 변의 길이와 넓이는 얼마인가?

[정답] 약 7.5