6장. 모듈(Module)

모듈(Module)

- **모듈(module)**
  - 변수나 함수 또는 클래스를 모아 놓은 **소스파일**로써, 이를 사용하는 다른 파일에서는 첫 부분에 [**import 모듈이름**]으로 선언한다.

모 듈	설 명
math	수학 계산과 관련된 모듈
datetime	날짜 및 시간과 관련된 모듈
time	시간과 관련된 모듈
random	난수를 발생시키는 모듈
os	운영 체제(OS) 자원 제어 관련 모듈

math 모듈

◎ **math** 모듈

〈 모듈 불러오기 〉

```
import math
```

기능	함수의 사용
올림	**math.ceil(2.54)** # 3
내림	**math.floor(2.54)** # 2
제곱근	**math.sqrt(16)** # 4.0
원주율	**math.pi** # 3.1415

math 모듈

◎ math 모듈

```
import math

# 올림
print(math.ceil(2.54))

# 반올림 - math 모듈이 아님
print(round(2.54))

# 내림(버림)
print(math.floor(2.54))

# 제곱근 - 실수로 반환
print(math.sqrt(2))
print(math.sqrt(25))
```

```
3
3
2
1.4142135623730951
5.0
3.141592653589793
원의 넓이: 50.27
```

math 모듈

◎ **math** 모듈

```
# 원주율
print(math.pi)

# 원의 넓이 = math.pi * 반지름 * 반지름
radius = 4
area = math.pi * radius * radius
print(f"원의 넓이: {area:.2f}")
```

datetime 모듈

◎ **datetime 모듈**

```python
import datetime

# datetime.datetime - 날짜와 시간을 사용
now = datetime.datetime.today() #오늘 날짜
print(now) # 2025-05-18 11:16:33.750807
# 년, 월, 일 출력
print(now.year)
print(now.month)
print(now.day)

# 현재 날짜 표기
print(f"{now.year}. {now.month}. {now.day}.")

# 시, 분, 초 출력
print(now.hour)
print(now.minute)
print(now.second)

# 현재 시간 표기
print(f"{now.hour} : {now.minute} : {now.second}")
```

```
2025-08-01 10:10:24.530677
2025
8
1
2025. 8. 1.
10
10
24
10 : 10 : 24
2025-08-15
2025-08-01
```

datetime 모듈

◎ 지나온 날짜 계산하기

datetime.date(2025, 8, 15) – 특정 날짜 설정

```
# 특정한 날짜 설정
# datetime.date - 날짜만 사용 가능
the_day = datetime.date(2025, 8, 15)
print(the_day)

today = datetime.date.today()
print(today)

# DDay 계산
print("광복절까지 몇일 남았나(DDay)?")

remain_day = the_day - today
print(f"광복절까지 {remain_day.days}일 남았습니다.")
```

```
2025-08-15
2025-08-01
광복절까지 몇일 남았나(DDay)?
광복절까지 14일 남았습니다.
```

calendar 모듈

◎ **calendar** 모듈

- **calendar.prcal(2025)** : 2025년의 달력을 표시
- **calendar.prmonth(2025, 5)** : 2025년 5월의 달력을 표시
- **calendar.weekday(2025, 5, 18)** : 날짜에 해당하는 요일 정보

```
import calendar

calendar.prcal(2025) #전체 출력

calendar.prmonth(2025, 5) #5월 출력

idx = calendar.weekday(2025, 5, 18) # 요일
print(idx)

days = ['월', '화', '수', '목', '금', '토', '일']
print(days[idx])
```

calendar 모듈

◎ calendar 모듈

```
                                  2025

      January                   February                   March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
       1  2  3  4  5                        1  2                      1  2
 6  7  8  9 10 11 12       3  4  5  6  7  8  9       3  4  5  6  7  8  9
13 14 15 16 17 18 19      10 11 12 13 14 15 16      10 11 12 13 14 15 16
20 21 22 23 24 25 26      17 18 19 20 21 22 23      17 18 19 20 21 22 23
27 28 29 30 31            24 25 26 27 28            24 25 26 27 28 29 30
                                                    31

      April                     May                       June
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6                  1  2  3  4                        1
 7  8  9 10 11 12 13       5  6  7  8  9 10 11       2  3  4  5  6  7  8
14 15 16 17 18 19 20      12 13 14 15 16 17 18       9 10 11 12 13 14 15
21 22 23 24 25 26 27      19 20 21 22 23 24 25      16 17 18 19 20 21 22
28 29 30                  26 27 28 29 30 31         23 24 25 26 27 28 29
                                                    30
```

```
6
일
```

time 모듈

◎ **time** 모듈

- time.time() 현재 시간을 실수 형태로 돌려주는 함수
- time.localtime() 연도, 월, 일, 시, 분, 초.. 형태
- time.sleep(2) 일정한 시간 간격을 두고 루프를 실행할 수 있다.

```
print(time.time())      # 현재시간을 초로 반환
print(time.localtime()) # 년,월,일,시,분,초로 반환
print(time.ctime())     # 날짜와 시간 표기 형태

# 년과 일로 환산(1970. 1.1 이후)
year = round(time.time()/(365*24*60*60))
day = round(time.time()/(24*60*60))
print(year)
print(day)
```

time 모듈

◎ 수행시간 측정하기

```
# 수행시간 측정
start = time.time() # 시작 시간

n = 10
for i in range(1, n + 1):
    print(i)
    time.sleep(0.5)  # 0.5초 간격으로 대기

end = time.time()  # 종료 시간
print(f"수행시간: {(end - start):.3f}초")
```

```
1
2
3
4
5
6
7
8
9
10
수행시간 : 10.010초
```

random 모듈

- **random** 모듈
  - random.random() : 0.0에서 1.0사이의 실수 값 중에서 난수 값 발생
  - random.randint(1, 10) : 1과 10사이의 정수중에서 난수 값 발생
  - random.choice(a) : 리스트에서 무작위로 하나를 선택

```
import random

0.0 <= random.random() < 1
print(random.random())

# random.randint(a, b) : a ~b 까지 정수 범위

# 1 ~ 10중 무작위수(난수) 발생
print(random.randint(1, 10))

# 주사위 눈
dice = random.randint(1, 6) # 1 ~ 6
# print(dice)
```

ramdom 모듈

- **random 모듈**

```
# 주사위 10번 던지기
for i in range(10):
    dice = random.randint(1, 6)
    print(dice)

# 동전 던지기
coin = random.randint(1, 2)
print(coin)
if(coin == 1):
    print("앞면")
else:
    print("뒷면")

# 리스트에서 무작위 추출 - random.choice(리스트)
fruits =["딸기", "참외", "수박", "바나나"]
fruit = random.choice(fruits)
print(fruit)
```

숫자 추측 게임

- **숫자를 추측해서 맞히는 게임**
  - **게임 방법**

    - 게임이 시작되면 컴퓨터가 난수(1 ~ 30 사이의 수)를 생성한다.
    - 사용자의 추측값이 정답과 같으면 '정답!', 추측값이 정답보다 크면 "너무 커요!", 추측값이 정답보다 작으면 '너무 작아요!' 출력
    - 사용자의 추측값이 1 ~ 30의 범위를 벗어나면 "범위를 초과했어요. 다시 입력하세요" 출력함

```
맞혀보세요(입력: 1 ~ 30): 40
범위를 초과했어요. 다시 입력하세요
맞혀보세요(입력: 1 ~ 30): 20
너무 작아요
맞혀보세요(입력: 1 ~ 30): 26
너무 작아요
맞혀보세요(입력: 1 ~ 30): 28
정답!
```

숫자 추측 게임

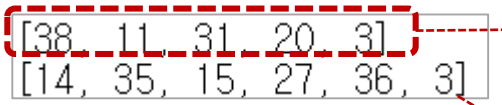
- 숫자를 추측해서 맞히는 게임

```
com = random.randint(1, 30) #컴퓨터 난수
# print(com)

while True:
    x = input("맞혀보세요(입력: 1 ~ 30): ")
    guess = int(x)  # 유저가 추측한 수
    #print(guess + 10)

    if guess < 1 or guess > 30:
        print("범위를 초과했어요. 다시 입력하세요")
    elif guess == com:
        print("정답!")
        break
    elif guess > com:
        print("너무 커요!")
    else:
        print("너무 작아요")

print("게임을 종료합니다.")
```

로또(lotto) 복권

- **로또(lotto) 복권 추첨 프로그램**

  로또 번호를 중복되지 않도록 생성하는 프로그램 만들기



중복된 경우 5개만 생성됨(오류)

중복되지 않고 6개 생성됨

로또(lotto) 복권

- **로또(lotto) 복권 추첨 프로그램**
  - for문 사용

```
import random

lotto = []

for i in range(6):
    n = random.randint(1, 45)
    if n not in lotto:
        lotto.append(n)
'''
  중복될 경우 5개만 저장됨
  [44, 7, 10, 8, 31]
'''
print(lotto)
```

로또(lotto) 복권

- **로또(lotto) 복권 추첨 프로그램**
  - while문 사용

```
lotto = []
while len(lotto) < 6:
    num = random.randint(1, 45)
    if num not in lotto:
        lotto.append(num)
print(lotto) #[40, 15, 25, 28, 18, 22]
print(sorted(lotto)) #[15, 18, 22, 25, 28, 40]

# [40, 15, 25, 28, 18, 22]
# 만약 2번 인덱스에서 10이 중복되면 삭제되고 또 추첨
```

영어 타자 연습 게임

- **게임 방법**
  - 게임이 시작되면 영어 단어가 화면에 표시된다.
  - 사용자는 최대한 빠르고 정확하게 입력해야 한다.
  - 바르게 입력했으면 다음 문제로 넘어가고 "통과"를 출력한다.
  - 오타가 있으면 같은 단어가 한 번 더 나온다.
  - 타자 게임 시간을 측정한다.

```
문제 3
earth
earth
통과!

문제 4
strawberry
str
오타! 다시 도전!

문제 4
potato
potato
통과!
```

```
문제 8
flower
flower
통과!

문제 9
sky
sky
통과!

문제 10
earth
earth
통과!
타자 시간: 28.80초
```

타자 연습 게임

- 영어 타자 연습 프로그램 1 – 리스트 활용

```
import random
import time

word = ["sky", "earth", "sun", "moon", "flower",
    "tree", "mountain", "strawberry", "garlic", "potato"]
n = 1 #문제 번호

print("[타자 게임] 준비되면 엔터!")
input()

start = time.time()  #시작 시간
```

타자 연습 게임 1

- 영어 타자 연습 프로그램 1

```
while n < 11:
    print("\n문제", n)
    question = random.choice(word)
    print(question)

    you = input()  # 사용자 입력
    if question == you:
        print("통과!")
        n += 1  #다음 문제
    else:
        print("오타! 다시 도전!")

end = time.time()    #종료 시간
et = end - start
print(f"타자 시간: {et:.2f}초")
```

타자 연습 게임 2

- 영어 타자 연습 프로그램 2 – split() 활용

```
# 문자열 분리로 배열 생성
str = "sky earth sun moon flower tree " \
      "mountain strawberry garlic potato"
word = str.split(' ') #공백문자로 구분
# print(word)
```

os 모듈(Module)

◎ **os** 모듈

- 환경변수나 디렉터리, 파일 등의 OS 자원을 제어할 수 있게 해주는 모듈이다.

```
import os

# pyworks 디렉터리로 이동
os.chdir('c:/pyworks')

# dir 명령 실행
dir = os.popen('dir')
print(dir.read())   # dir 결과 출력

# 파일 목록을 리스트로 얻기
files = os.listdir('c:/pyworks')
print(files)
print(files[1])
# 전체 출력
for file in files:
    print(file)
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: 4A81-5207

 c:\pyworks 디렉터리

2025-04-24  오전 07:48    <DIR>          .
2025-04-24  오전 08:03    <DIR>          basic
2025-04-24  오전 08:03    <DIR>          choice
               0개 파일                   0 바이트
               3개 디렉터리  31,223,566,336 바이트 남음

['basic', 'choice']
choice
basic
choice
```

모듈의 사용과 패키지

◆ **모듈 사용방법**

- **import** 모듈 이름
- **from** 패키지 이름 **import** 파일(모듈)이름
- **from** 패키지이름.파일이름 **import** 함수, 클래스

```
import datetime

# 현재 날짜와 시간 출력
now = datetime.datetime.now()
print(now)

# 특정한 날짜 설정
today = datetime.date.today()
print(today)
```

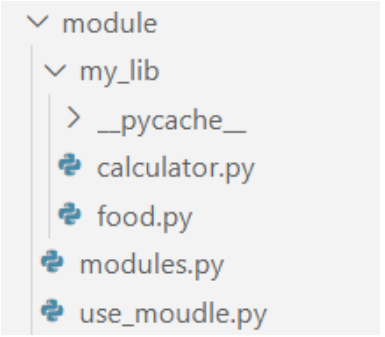
```
from datetime import datetime, date

# 현재 날짜와 시간
now = datetime.now()
print(now)

# 오늘 날짜 출력
today = date.today()
print(today)
```

사용자 정의 모듈

◆ 모듈을 만들고 사용하기

```python
# 모듈 만들기 - calculator.py
def add(x, y):
    return x + y

def sub(x, y):
    return x - y

def mul(x, y):
    return x * y

def div(x, y):
    if y != 0:
        return x / y
    else:
        print("0으로 나눌 수 없습니다.")
```

```python
# 모듈 만들기 - food.py
name = "장금이"

def cook():
    print("요리하다")

def eat():
    print("먹는다")
```



사용자 정의 모듈

◆ 모듈을 만들고 사용하기

```
from my_lib.calculator import add, sub, mul, div
from my_lib.food import name, cook, eat

# calculator 모듈 사용하기
# add() 호출
addition = add(10, 4)
print(addition)

# sub() 호출
subtraction = sub(10, 4)
print(subtraction)

# mul() 호출
multiple = mul(10, 4)
print(multiple)
```

사용자 정의 모듈

◆ 모듈을 만들고 사용하기

```
# mul() 호출
multiple = mul(10, 4)
print(multiple)

# div() 호출
division = div(10, 4)

# 분모가 0인 경우
division = div(10, 0)
print(division)

# food 모듈 사용하기
print(f'이름: {name}')

cook()

eat()
```

```
14
6
40
2.5
이름: 장금이
요리하다
먹는다
```

```
0으로 나눌 수 없습니다.
None
```

사용자 정의 모듈

◆ 모듈 사용 방법 2

```
# from 패키지(디렉터리) 이름 import 모듈이름
from my_lib import food
from my_lib import calculator

# calculator
print(calculator.add(10, 4))
print(calculator.sub(10, 4))
print(calculator.mul(10, 4))
print(calculator.div(10, 4))

# food
print(food.name)
food.cook()
```

실행 파일(.exe) 만들기

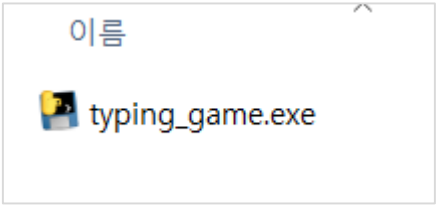
- **pyinstaller 모듈 설치**

  pip install pyinstaller

- **파일(스크립트)이 1개인 경우 터미널에 명령어 입력**

  C:₩pyworks/module>pyinstaller --onefile typing_game.py

  같은 경로에 dist 폴더 생성됨



[dist] 폴더

실행 파일(.exe) 만들기

✓ **콘솔창 꺼짐 문제 발생**

os 모듈을 impor하고 system("pause") 명시한다.

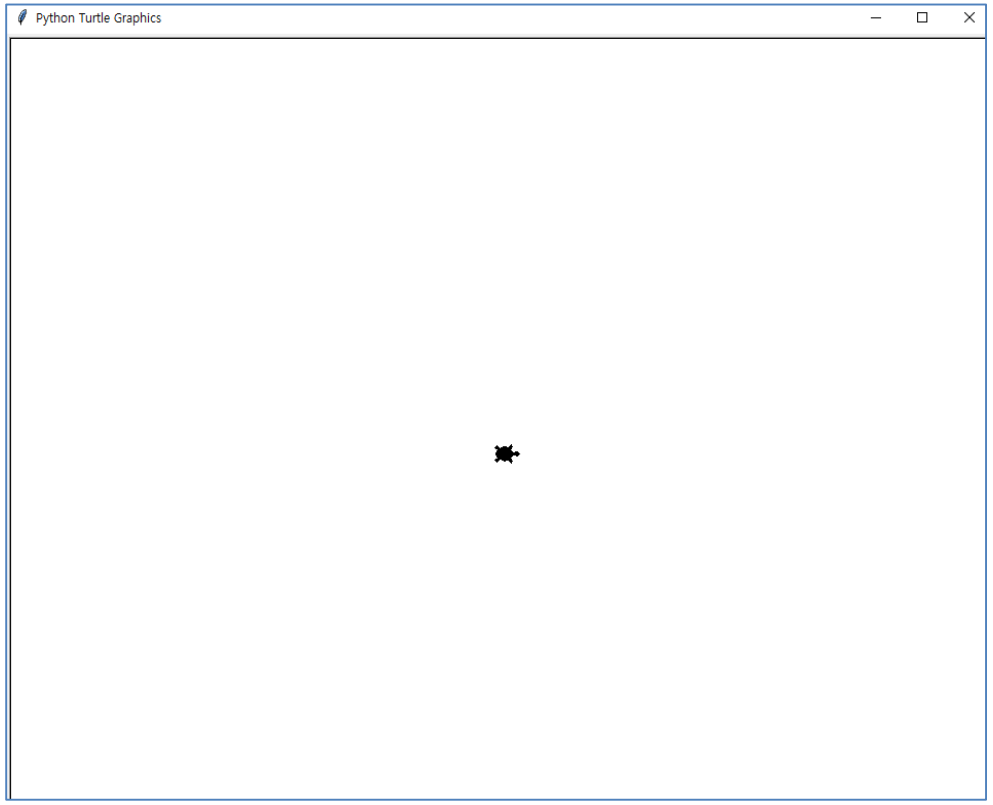
```
import random
import time
import os

word = ["sky", "earth", "sun", "moon", "flower",
    "tree", "mountain", "strawberry", "garlic", "potato"]
n = 1 #문제 번호
```

```
end = time.time()    #종료 시간
et = end - start
print(f"타자 시간: {et:.2f}초")

os.system("pause") # exe파일 - 콘솔창 유지
```

거북이 그래픽 모듈

➢ 거북이 그래픽 모듈

모듈(Module)이란 만들어져 사용 가능한 프로그램의 단위를 말한다.



거북이 그래픽 모듈

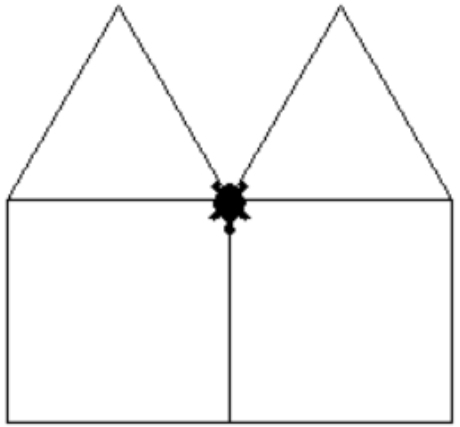
➢ 이동 및 방향 바꾸기

import turtle as t # t는 별칭

t.shape("turtle") – 거북이 모양

t.forward(거리) – 거리만큼 직진함

t.right(각도) – 오른쪽으로 각도만큼 방향을 바꿈



거북이 그래픽 모듈

➢ 이동 및 방향 바꾸기

```
# 사각형
t.forward(100) #100픽셀 직진
t.right(90)    #오른쪽으로 90도 회전
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)

# 삼각형
t.forward(100)
t.left(120)     #왼쪽으로 120도 회전
t.forward(100)
t.left(120)
t.forward(100)
t.left(120)
```

```
# 방향 전환
t.right(180)

# 삼각형
t.forward(100)
t.right(120)
t.forward(100)
t.right(120)
t.forward(100)
t.right(120)

# 방향 전환
t.left(90)

# 사각형
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
```

거북이 그래픽 모듈

➢ **반복문 사용하기**

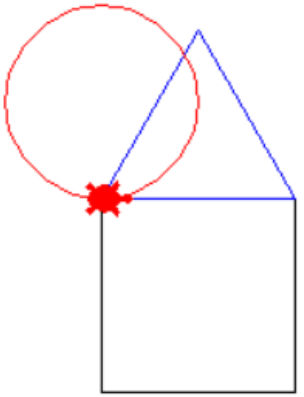
```
# 사각형
for i in range(4):
    t.forward(100)
    t.right(90)

# 삼각형
for i in range(3):
    t.forward(100)
    t.left(120)

# 방향 전환
t.right(180)

# 삼각형
for i in range(3):
    t.forward(100)
    t.right(120)
```

```
# 삼각형
for i in range(3):
    t.forward(100)
    t.right(120)

# 방향 전환
t.left(90)

# 사각형
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
```

거북이 그래픽 모듈

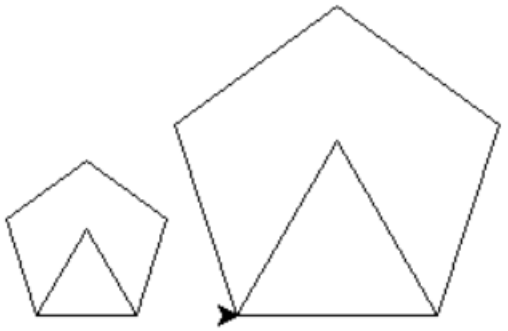
➢ 변수 사용하기

```
# 사각형
d = 100
n = 4
for i in range(n):
    t.forward(d)
    t.right(360/n)

# 삼각형
t.color("blue")
n = 3
for i in range(n):
    t.forward(d)
    t.left(360/n)

# 원
t.color('red')
t.circle(50)  #반지름 50픽셀
```



다각형 그리기

- 함수 사용하기

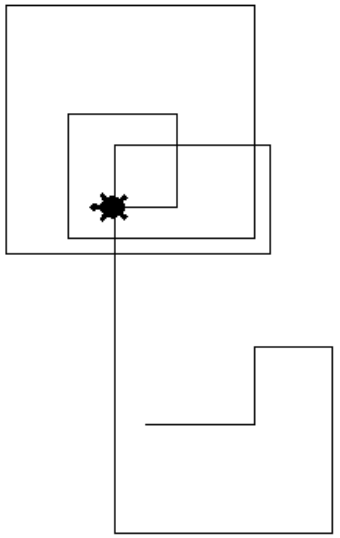


```
def polygon(n):
    for x in range(n):
        t.forward(50)
        t.left(360/n)  #내각

def polygon2(n, d):
    for x in range(n):
        t.forward(d)
        t.left(360/n)
```

```
polygon(3)  #삼각형
polygon(5)  #오각형

t.up()   #펜 올리기
t.forward(100)
t.down() #펜 내리기

polygon2(3, 100)
polygon2(5  , 100)
```

키보드로 조종하기

❖ 키보드로 거북이 조종하기



```
t.shape("turtle")
#상수 - Right(오른쪽), 첫글자 대문자
t.onkeypress(turn_right, "Right")
t.onkeypress(turn_up, "Up")
t.onkeypress(turn_left, "Left")
t.onkeypress(turn_down, "Down")
t.listen() #동작 실행
```

```
# 오른쪽으로 회전후 이동
def turn_right():
    t.setheading(0)
    t.forward(10)

# 위쪽으로 회전후 이동
def turn_up():
    t.setheading(90)
    t.forward(10)

# 왼쪽으로 회전후 이동
def turn_left():
    t.setheading(180)
    t.forward(10)

# 아래쪽으로 회전후 이동
def turn_down():
    t.setheading(270)
    t.forward(10)
```

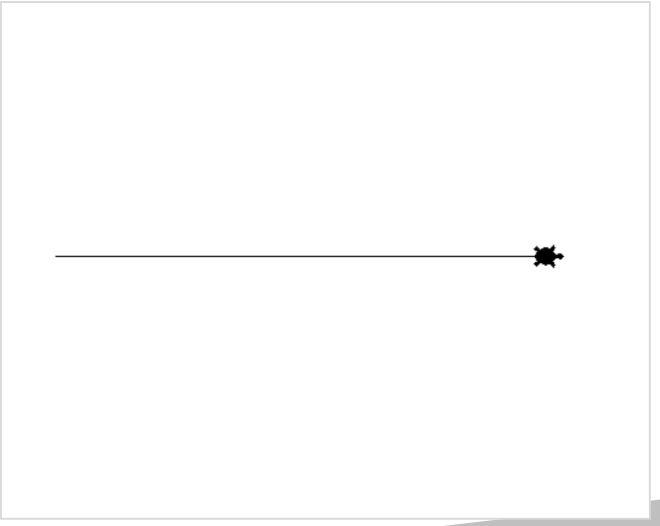
좌표 이동

- **turtle.goto(x, y) : 좌표 이동**
  - **특정 위치에서 나타나기**

```
t.up()  #펜 올리기
t.speed(0) #가장 빠른 속도(0 ~ 10)
t.goto(200, 0) #x좌표: 200, y좌표 0
t.goto(0, 200)
```

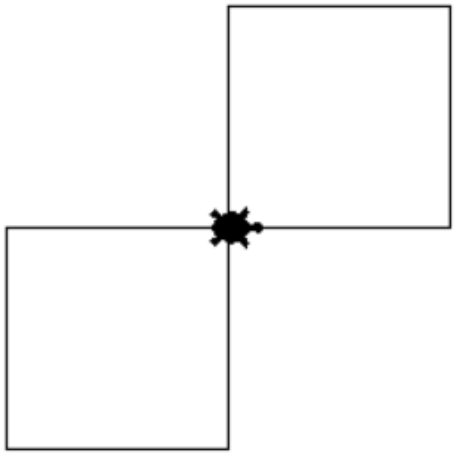
  - **직선 그리기**

```
t.down() #펜 내리기
t.goto(-200, 0) #200픽셀
t.goto(200, 0)  #200픽셀
```



좌표 이동

- **turtle.goto(x, y) : 좌표 이동**
  - 좌표(0, 0)에서 사각형 그리기
  - 1초 간격으로 그리기 : time.sleep(1)



좌표 이동

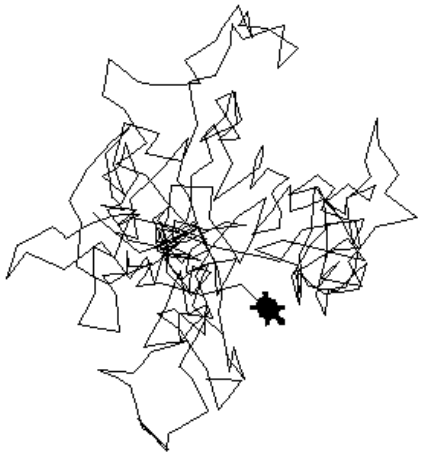
- **turtle.goto(x, y) : 좌표 이동**

```
t.goto(0, 0) #시작 위치

time.sleep(1)  #대기 시간 1초
t.goto(-100, 0)

time.sleep(1)
t.goto(-100, -100)

time.sleep(1)
t.goto(0, -100)

time.sleep(1)
t.goto(0, 0) #시작 위치
```

```
time.sleep(1)
t.goto(0, 100)

time.sleep(1)
t.goto(100, 100)

time.sleep(1)
t.goto(100, 0)

time.sleep(1)
t.goto(0, 0)
```

거북이 그래픽 모듈

- ***마음대로 걷는 거북이***



```
# 랜덤 위치 지정
# t.speed(0)  #가장 빠름
x = random.randint(-250, 250)
y = random.randint(-250, 250)
# t.up()
t.goto(x, y)
```

```
# 마음대로 걷는 거북이
# 거북이의 머리 방향(각도) - 랜덤
t.speed(0)

n = 300  #반복 횟수
for i in range(n):
    ang = random.randint(1, 360)
    t.setheading(ang) #머리 방향
    t.forward(20)

t.mainloop()
```

거북이 대포 게임

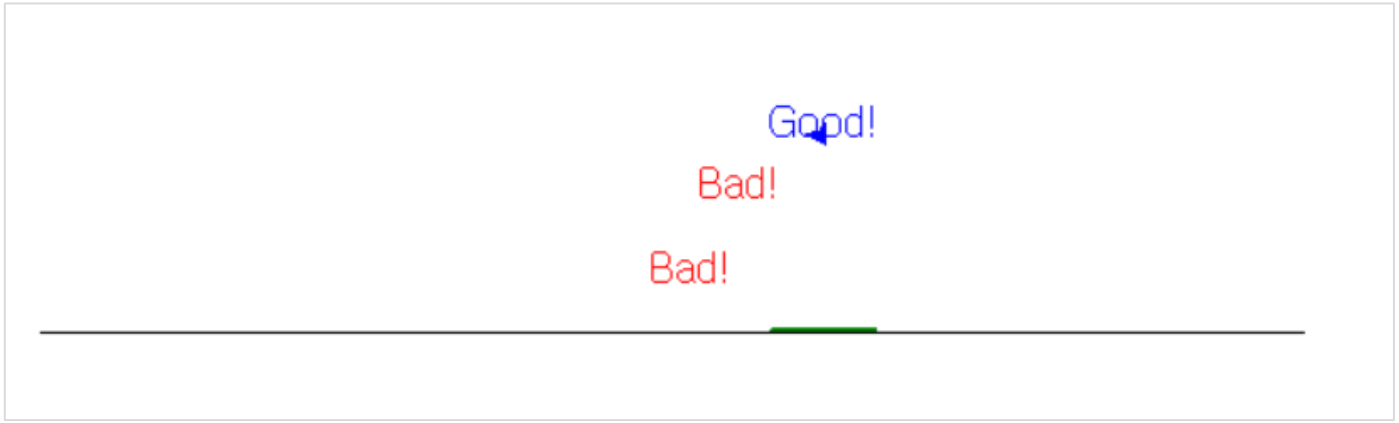
- 거북이 대포 게임

  각도를 맞춰 대포를 발사해 목표 지점을 맞히는 게임

  게임 방법

  ① 키보드 방향키로 발사 각도를 조절하고, 스페이스 바로 대포를 발사하면 화살촉 모양의 포탄이 하늘로 날아간다.

  ① 포탄이 땅에 닿을때 초록색 목표 지점을 맞히면 'Good!'이라는 메시지를 보여주고, 빗나가면 'Bad!'라는 메시지를 보여줌.



거북이 대포 게임

- **cannon(대포)**

```
# 땅 그리기
t.goto(-300, 0)
t.goto(300, 0)

# 목표 지점 설정
target = random.randint(50, 150)
t.color('green')
t.pensize(2)
t.up()  #펜 올리기

# 목표 지점의 길이 - 50px
t.goto(target-25, 1)
t.down()
t.goto(target+25, 1)
```

```
# 포탄의 처음 위치
t.color('black')
t.up()
t.goto(-200, 10)
t.setheading(20)

# 거북이 대포 동작 설정
t.onkeypress(turn_up, "Up")
t.onkeypress(turn_down, "Down")
#스페이스 키를 누르면 발사됨
t.onkeypress(fire, "space")
t.listen() #동작 실행

t.mainloop()
```

거북이 대포 게임

- **cannon(대포)**

```
def turn_up():
    t.left(2)

def turn_down():
    t.right(2)

def fire():
    angle = t.heading() #거북이가 바라보는 현재 각도
    while t.ycor() > 0: #포탄이 땅위에 있는동안
        t.forward(15)
        t.right(5)
```

거북이 대포 게임

- **cannon(대포)**

```
d = t.distance(target, 0) #포탄과 목표지점과의 거리
# t.write(d)
t.sety(random.randint(10, 100))  #y좌표 설정-성공, 실패를 표시할 위치
if d < 25: #목표 지점에 닿으면
  t.color('blue')
  # 문자열 쓰기 - 글꼴 크기 15, False-포탄의 위치를 옮기지 않음
  t.write("Good!", False, 'center', ('', 17))
else: #목표 지점에 닿지 않으면
  t.color('red')
  t.write("Bad!", False, 'center', ('', 15))
  t.color('black')
  t.goto(-200, 10)  #거북이를 처음 발사 위치로 보냄
  t.setheading(ang) #처음 기억한 각도로 되돌림
```

거북이 대표 게임

- **cannon(대포)**

글자쓰기 함수

t.write(“문자열”, False, “center”, (“”, 15)

False – 거북이는 위치를 옮기지 않음

center – 문자열을 가운데 정렬

(“”, 15) – 전달된 문자열의 글자크기 15