

6장. 모듈과 패키지



모듈(Module)

- 모듈(module)

모듈은 Python 파일 하나를 의미하며, 다른 Python 프로그램에서 모듈의 함수, 클래스 등을 재사용할 수 있다.

- 모듈 만들기

```
# my_module.py
# 요리하다
def cook(name):
    return f"{name} 요리를 합니다."

# 청소하다
def clean(place):
    return f"{place}를 청소합니다."
```

모듈(Module)

- 모듈 사용하기

from 모듈 이름 import 함수, 클래스

```
# main.py
# my_module 파일을 불러옵니다.
from my_module import cook, clean

print("요리 시작!")
print(cook("파스타"))

print("청소 시작!")
print(clean("욕실"))
```

패키지(Package)

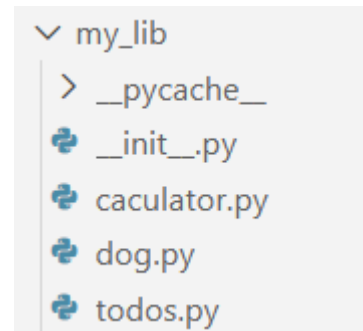
❖ 패키지

여러 모듈을 묶은 디렉터리 구조로 파이썬의 패키지로 인정받기 위해서는 `__init__.py`를 필수로 생성해야 한다.

`__init__.py`는 코드가 없는 빈 파일이다.

▪ 패키지 사용하기

- **import** 패키지 이름
- **from** 패키지 이름 **import** 모듈 이름
- **from** 모듈이름 **import** 함수, 클래스



모듈과 패키지

◆ 패키지 만들기

```
# my_lib/todos.py
# 요리하다
def cook(name):
    return f"{name} 요리를 합니다."

# 청소하다
def clean(place):
    return f"{place}를 청소합니다."
```

모듈과 패키지

◆ 패키지 만들기

```
# my_lib/caculator.py
# 사칙연산 계산 함수
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "0으로 나눌 수 없습니다."
    return a / b
```

모듈과 패키지

◆ 패키지 만들기

```
class Dog:
    kind = '말티즈' # 클래스 변수

    def __init__(self, name):
        self.name = name # 인스턴스 변수

    def bark(self):
        print(f"{self.name}가 짖습니다: 멍멍!")
```

모듈과 패키지

◆ 패키지 사용하기 - main.py

```
# my_lib 패키지에서 모듈 불러오기
from my_lib.todos import cook, clean

print("요리 시작!")
print(cook("파스타"))
print("청소 시작!")
print(clean("욕실"))

# from my_lib.caculator import add, subtract, multiply, divide
from my_lib.caculator import *
print("사칙연산 계산기")
a, b = 10, 5 # 구조 분해 할당

print(f"{a} + {b} = {add(a, b)}")
print(f"{a} - {b} = {subtract(a, b)}")
print(f"{a} * {b} = {multiply(a, b)}")
print(f"{a} / {b} = {divide(a, b)}")
```

모듈과 패키지

◆ 패키지 사용하기

```
# dog 모듈 불러오기
from my_lib.dog import Dog

print("강아지 정보 출력")
my_dog = Dog("뽀빠")
print(f"강아지 이름: {my_dog.name}, 종류: {Dog.kind}")
my_dog.bark()

your_dog = Dog("초코")
print(f"강아지 이름: {your_dog.name}, 종류: {Dog.kind}")
your_dog.bark()
```

모듈과 패키지

- 주요 모듈 및 패키지

모듈	설명
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

기능	함수의 사용
올림	<code>math.ceil(2.54)</code> # 3
내림	<code>math.floor(2.54)</code> # 2
제곱근	<code>math.sqrt(16)</code> # 4.0
원주율	<code>math.pi</code> # 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

```
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(2026)` : 2026년의 달력을 표시
- `calendar.prmonth(2026, 1)` : 2026년 1월의 달력을 표시
- `calendar.weekday(2026, 2, 10)` : 날짜에 해당하는 요일 정보

2026																				
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							1							1
5	6	7	8	9	10	11	2	3	4	5	6	7	8	2	3	4	5	6	7	8
12	13	14	15	16	17	18	9	10	11	12	13	14	15	9	10	11	12	13	14	15
19	20	21	22	23	24	25	16	17	18	19	20	21	22	16	17	18	19	20	21	22
26	27	28	29	30	31		23	24	25	26	27	28		23	24	25	26	27	28	29
																				30 31

calendar 모듈

◎ calendar 모듈

```
import calendar

# 2026 Calendar
calendar.prcal(2026)

# January 2026 Calendar
calendar.prmonth(2026, 1)

# 요일 이름 출력
print("요일 이름:", calendar.day_name[:])

# 특정 날짜의 요일 반환 (0: 월요일, 6: 일요일)
day_of_week = calendar.weekday(2026, 2, 10)
print(f"2026년 2월 10일의 요일: {calendar.day_name[day_of_week]}")
```

calendar 모듈

◎ calendar 모듈

```
# 2024년 1월 27일의 요일 확인(한글이름으로)
idx = calendar.weekday(2026, 1, 27)
days = ['월요일', '화요일', '수요일', '목요일', '금요일', '토요일', '일요일']
print(f"2024년 1월 27일은 {days[idx]}입니다.")

# 윤년 여부 확인
is_leap = calendar.isleap(2026)
print(f"2026년은 윤년인가요? {is_leap}")

# 특정 연도의 윤년 개수 계산
leap_days = calendar.leapdays(2000, 2027) # 2000년부터 2026년까지의 윤년 개수
print(f"2000년부터 2026년까지의 윤년 개수: {leap_days}")
```

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.

```
import time
```

```
print(time.time()) # 현재 시간의 타임스탬프 출력  
print(time.localtime()) # 현재 시간을 지역 시간으로 출력  
print(time.ctime()) # 현재 시간을 읽기 쉬운 문자열로 출력
```

```
1769313182.573292  
time.struct_time(tm_year=2026, tm_mon=1, tm_mday=25, tm_hour=12,  
Sun Jan 25 12:53:02 2026  
3초 대기 시작...  
3초 대기 끝.  
포맷팅된 현재 시간: 2026-01-25 12:53:05  
현재까지 경과한 일수: 20478.16일  
현재까지 경과한 연수: 56.1년
```

time 모듈

◎ time 모듈

```
# 특정 시간으로 대기
print("3초 대기 시작...")
time.sleep(3)
print("3초 대기 끝.")

# 시간 포매팅
formatted_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
print("포매팅된 현재 시간:", formatted_time)

# 년과 일로 환산(1970-01-01부터 경과한 시간)
days = round(time.time() / (24 * 60 * 60), 2)
year = round(time.time() / (365 * 24 * 60 * 60), 2)
print(f"현재까지 경과한 일수: {days}일")
print(f"현재까지 경과한 연수: {year}년")
```

time 모듈

◎ 수행시간 측정하기

```
start = time.time()

n = 100
for i in range(1, n + 1):
    print(i)
    time.sleep(0.1) # 0.1초 대기

end = time.time()
print(f"\n1부터 {n}까지 출력하는데 걸린 시간: {end - start}초")
```

random 모듈

■ 챗GPT 프롬프트 입력

+ 파이썬의 random 모듈을 사용하는 방법에 대해서 알려줘



1 random 모듈 불러오기

python

코드 복사

```
import random
```

2 기본 난수 생성

◆ 0 이상 1 미만의 실수

python

코드 복사

```
random.random()
```

예: 0.3748293

◆ 특정 범위의 정수

python

코드 복사

```
random.randint(1, 10)
```

예 1 이상 10 이하 정수 하나

random 모듈

● random 모듈

```
# 난수 시드 설정 - 동일한 시드를 사용하면 동일한 난수 시퀀스 생성
random.seed(42)
print(f"시드 설정 후 난수: {random.randint(1, 10)}")
random.seed(42)
print(f"시드 설정 후 난수: {random.randint(1, 10)}")
random.seed() # 시드 초기화
print(f"시드 초기화 후 난수: {random.randint(1, 10)}")

# 동전 던지기 시뮬레이션
# '앞면' 또는 '뒷면' 중 하나를 무작위로 선택
# 방법1
coin = random.randint(0, 1)
if coin == 0:
    print("동전 던지기 결과: 앞면")
else:
    print("동전 던지기 결과: 뒷면")

# 방법2
coin = random.choice(['앞면', '뒷면'])
print(f"동전 던지기 결과: {coin}")
```

random 모듈

```
# 주사위 굴리기 시뮬레이션
dice_roll = random.randint(1, 6)
print(f"주사위 굴리기 결과: {dice_roll}")

# 로또 번호 생성 (1부터 45 사이의 숫자 중 6개 선택)
lotto_numbers = random.sample(range(1, 46), k=6)
print(f"생성된 로또 번호: {sorted(lotto_numbers)}")

# 리스트에서 무작위로 요소 선택
fruits = ['사과', '바나나', '딸기', '감', '사과']
selected_fruit = random.choice(fruits)
print(f"선택된 과일: {selected_fruit}")

# 리스트에서 무작위로 여러 요소 선택 (중복 허용)
selected_fruits = random.choices(fruits, k=2)
print(f"선택된 과일들 (중복 허용): {selected_fruits}")

# 리스트에서 무작위로 여러 요소 선택 (중복 비허용)
selected_fruits = random.sample(fruits, k=2)
print(f"선택된 과일들 (중복 비허용): {selected_fruits}")

# 리스트 섞기
random.shuffle(fruits)
print(f"섞인 과일 리스트: {fruits}")
```

숫자 맞추기 게임

- 숫자 맞추기 게임

- 챗GPT 프롬프트 입력

파이썬으로 숫자 맞추기 게임을 만들 거야.
1~100까지 무작위 숫자를 생성하고,
내가 숫자를 입력하면 무작위 숫자와 비교해서 숫자가 큰지 작은지 알려줘
숫자를 맞추면 몇 회 만에 맞췄는지 알려주고 게임을 종료해

+



숫자 추측 게임

- 숫자를 추측해서 맞히는 게임

```
import random

# 1~100 사이의 무작위 숫자 생성
answer = random.randint(1, 100)
# print(answer) # 디버깅용 출력, 실제 게임에서는 제거할 것

count = 0 # 시도 횟수

print("숫자 맞추기 게임!")
print("1부터 100 사이의 숫자를 맞춰보세요.")

while True:
    guess = int(input("숫자를 입력하세요: "))
    count += 1

    if guess > answer:
        print("더 작은 숫자입니다.")
    elif guess < answer:
        print("더 큰 숫자입니다.")
    else:
        print(f"정답입니다! {count}회 만에 맞췄어요.")
        break
```

로또(lotto) 복권

- 로또(lotto) 복권 추첨 프로그램

로또 번호를 중복되지 않도록 생성하는 프로그램 만들기

```
[38, 11, 31, 20, 31]  
[14, 35, 15, 27, 36, 3]
```

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

# 현재 작업 디렉터리 확인
current_dir = os.getcwd()
print("현재 작업 디렉터리:", current_dir)

# 디렉터리 목록 출력
dir_list = os.listdir()
print("디렉터리 목록:", dir_list)
dir = os.popen('dir')
print(dir.read())

# pyworks 디렉터리로 이동
os.chdir('c:/pyworks')
print("디렉터리 변경 후 현재 작업 디렉터리:", os.getcwd())

# 경로 관련 정보 확인
print("경로 정보 확인:")
print("경로 존재 여부:", os.path.exists('c:/pyworks')) # 경로 존재 여부
print("디렉토리 여부:", os.path.isdir('c:/pyworks')) # 디렉토리 여부
```

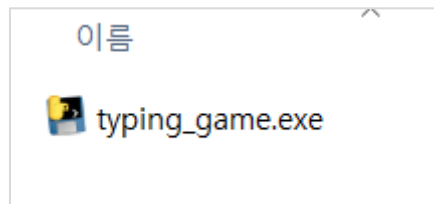
실행 파일(.exe) 만들기

- pyinstaller 모듈 설치
`pip install pyinstaller`

- 파일(스크립트)이 1개인 경우 터미널에 명령어 입력

`C:\wpyworks/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
```

텍스트를 음성으로 변환


■ 챗GPT 프롬프트 입력


+ 파이썬을 이용해서 한글 텍스트를 음성으로 변환하는 코드를 작성해줘



■ gtts 패키지 설치하기

pip install gtts

 텍스트를 음성으로 변환.py

 output.mp3

텍스트를 음성으로 변환

```
from gtts import gTTS

# 변환할 한글 텍스트
text = "안녕하세요. 파이썬으로 한글 텍스트를 음성으로 변환합니다."

# TTS 객체 생성 (lang='ko' 중요!)
tts = gTTS(text=text, lang='ko')

# mp3 파일로 저장
tts.save("output.mp3")

print("음성 파일이 생성되었습니다: output.mp3")

# 생성된 mp3 파일 재생 (선택 사항)
import os

os.system("start output.mp3") # Windows에서는 'start', macOS에서는 'open'
```

거북이 그래픽 모듈

■ 챗GPT 프롬프트 입력

+ 파이썬 turtle 모듈 사용하는 방법에 대해 알려줘



`turtle` 모듈은 파이썬으로 그림 그리듯이 코딩할 수 있게 해주는 초보자 친화적인 그래픽 모듈이에요. 하나씩 차근차근 설명해볼게요.

1 turtle 모듈 시작하기

python

📄 코드 복사

```
import turtle
```

이 한 줄로 `turtle` 을 불러옵니다.

기본 세팅:

python

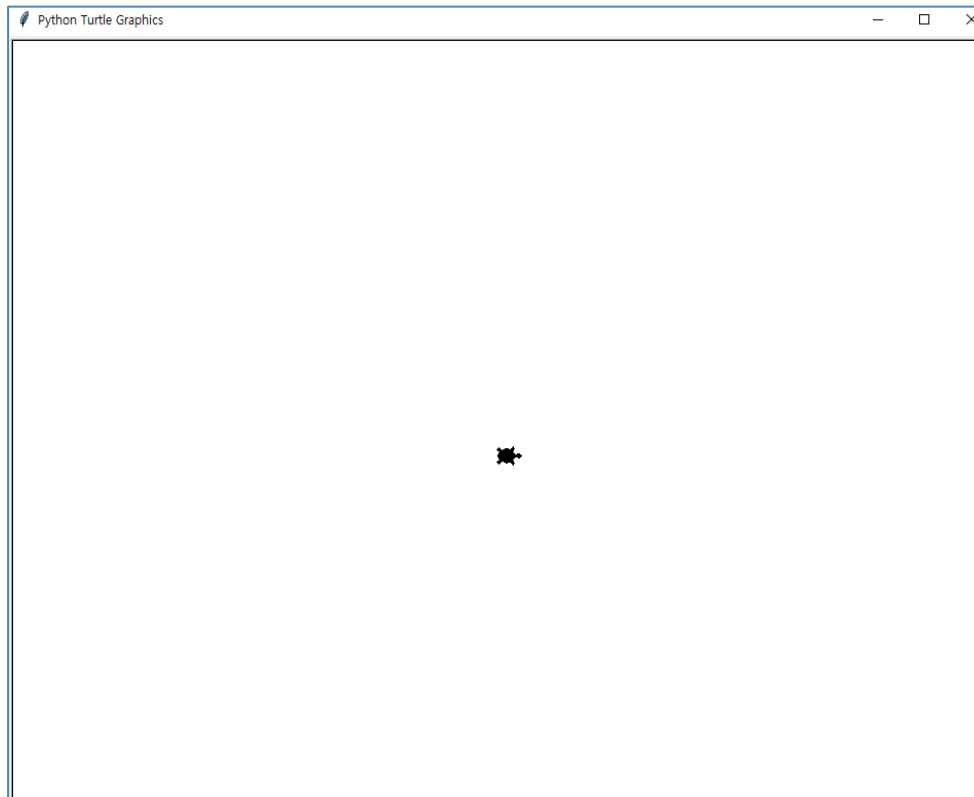
📄 코드 복사

```
t = turtle.Turtle() # 거북이 생성
turtle.Screen()     # 화면 생성 (보통은 생략 가능)
```

거북이 그래픽 모듈

➤ 거북이 그래픽 모듈

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



거북이 그래픽 모듈

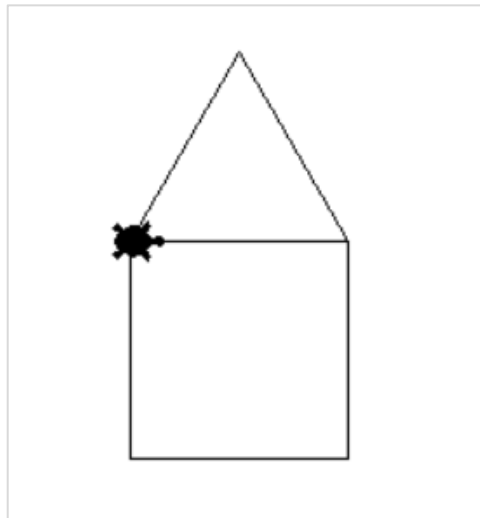
➤ 이동 및 방향 바꾸기

`import turtle as t` # `t`는 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

```
import turtle as t

t.shape("turtle")

# 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)
```

```
# 삼각형
t.forward(100)
t.left(120)
t.forward(100)
t.left(120)
t.forward(100)
t.left(120)

t.mainloop()
```

거북이 그래픽 모듈

➤ 반복문 사용하기

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

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

t.mainloop()
```

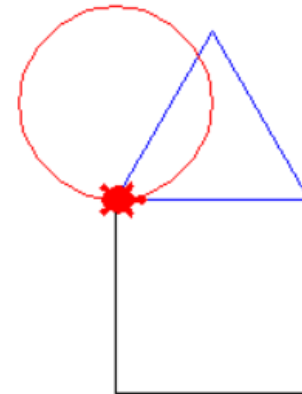
거북이 그래픽 모듈

➤ 변수 사용하기

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

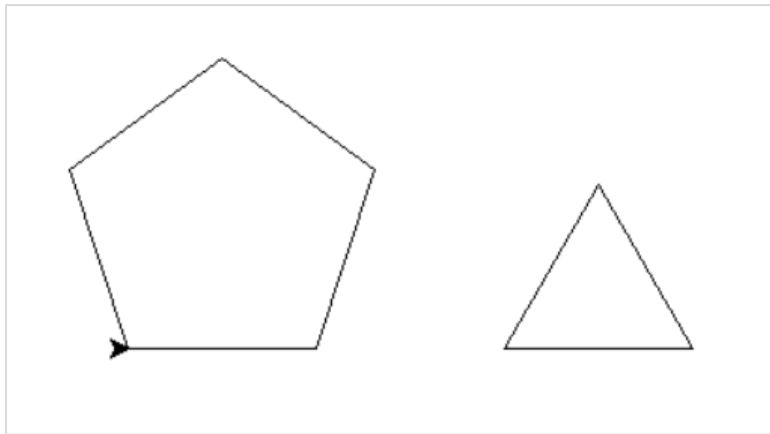
# 삼각형
t.color("blue")
n = 3
for i in range(n):
    t.forward(d)
    t.left(360/n)

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



다각형 그리기

- 함수 사용하기



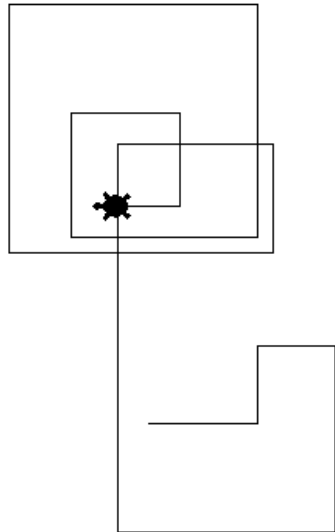
```
def polygon(n):  
    for x in range(n):  
        t.forward(100)  
        t.left(360/n)
```

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

```
polygon(3)  
t.penup() # 펜 올리기  
  
t.backward(200)  
t.pendown() # 펜 내리기  
polygon2(5, 100)  
  
t.mainloop()
```

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 회전 후 이동
def turn_right():
    t.setheading(0)
    t.forward(10)
```

```
# 위쪽으로 회전 후 이동
def turn_up():
    t.setheading(90)
    t.forward(10)
```

```
# 왼쪽으로 회전 후 이동
def turn_left():
    t.setheading(180)
    t.forward(10)
```

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

좌표 이동

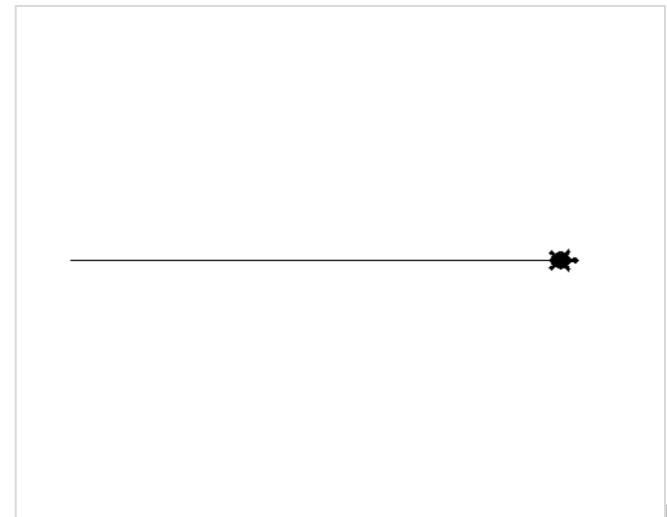
➤ turtle.goto(x, y) : 좌표 이동

- 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)
```

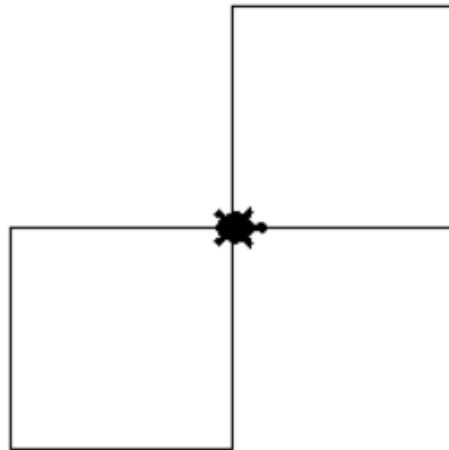
- 직선 그리기

```
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)

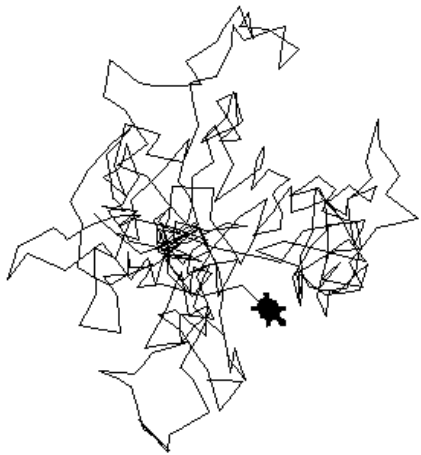
time.sleep(1)
t.goto(100, 100)

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()
```

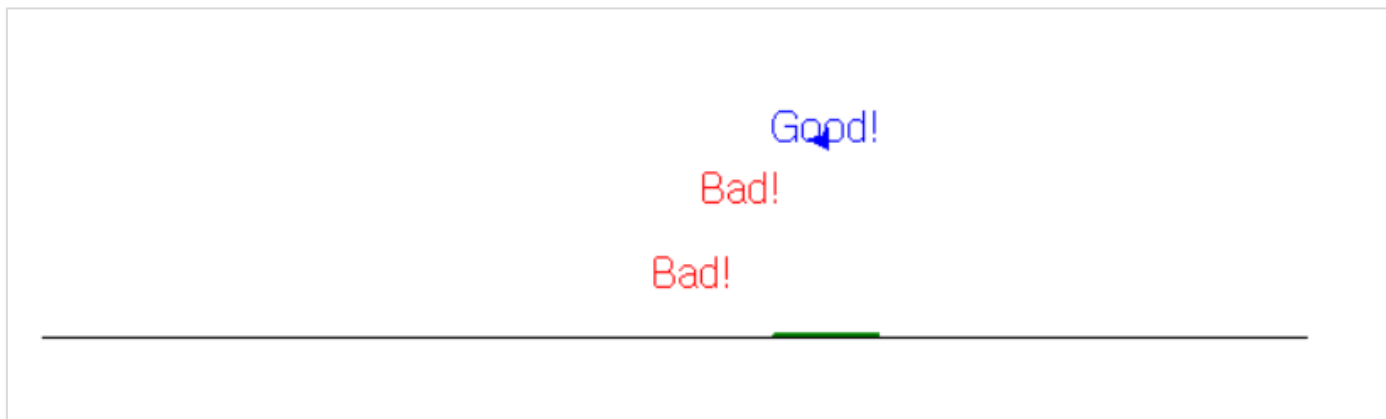
거북이 대포 게임

● 거북이 대포 게임

각도를 맞춰 대포를 발사해 목표 지점을 맞추는 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 대포 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