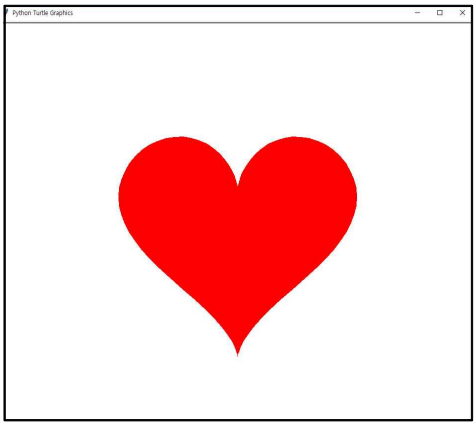
[1. 하트 그리기-heart_draw.py, 교과서-133,148페이지] 하나의 빨간색 하트를 그리는 프로그램을 작성해봅시다.

[실행 결과 예시]



[코드]

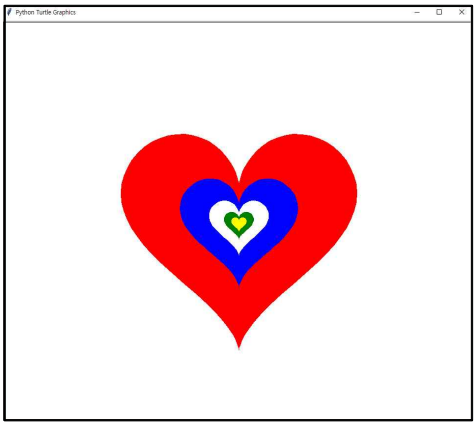
```
import turtle as t              # turtle 모듈을 t로 설정하여 사용
import math as m                # math 모듈을 m으로 설정하여 사용

t.color("red")                  # 펜(거북이)의 색깔을 red로 설정
t.begin_fill()                  # 영역 채우기 시작(펜의 색깔과 동일한 색으로 채워짐)
t.up()                          # 펜을 들어서 그림이 그려지지 않도록 설정
pnum=100                        # 하트를 그리기 위한 점의 개수를 100으로 설정
size=15                         # 하트 그림의 크기를 15로 설정

for i in range(1,pnum+1):       # 하트를 그리기 위한 점의 개수만큼 반복
    h = m.pi*i/(pnum/2)         # 변수 h는 각도(radian)를 의미함( π*1/50 ~ π*100/50)
    x = size*16*m.sin(h)**3     # 변수 x,y에는 '하트방정식' 공식에 의해 하트 모양의 좌표 값이 입력됨
    y = size*13*m.cos(h) - size*5*m.cos(2*h) - size*2*m.cos(3*h) - size*m.cos(4*h)
    t.goto(x,y)                 # 좌표 (x,y)로 이동
    t.down()                    # 펜을 내려서 그림이 그려지도록 설정
t.end_fill()                    # 영역 채우기 끝
```

[참고]

- 다양한 모양의 하트를 그릴 수 있는 방정식이 존재합니다. '하트방정식'으로 검색해보세요.
- t.up()과 t.down()을 주석처리하고 실행시켜보세요. (주석은 문장 앞에 #을 추가)
- 변수 pnum과 size의 값을 달리하여 실행시켜보세요.
- [코드 11번째 줄] 1 radian = 1 degree * (π/180) / 1 degree = 1 radian * (180/π)
- 삼각함수의 인수(각도) 단위는 radian이에요.

[2. 하트 그리기2-heart_draw2.py, 교과서-133,149페이지] 5개의 하트를 색깔과 크기를 달리하여 그리는 프로그램을 작성해봅시다.

[실행 결과 예시]



[코드]

```
import turtle as t
import math as m

c=["red", "blue", "white", "green", "yellow"]    # 펜의 색깔을 5가지로 설정하기 위한 리스트
pnum=100
size=30

for j in range(1,6):      # 총 5개의 하트 그리기
    t.color(c[j-1])       # 리스트를 활용하여 하트의 색깔을 변화
    t.begin_fill()
    t.up()
    size=size/2           # 하트의 크기를 반으로 줄여나감

    for i in range(1,pnum+1):
        h = m.pi*i/(pnum/2)
        x = size*16*m.sin(h)**3
        y = size*13*m.cos(h) - size*5*m.cos(2*h) - size*2*m.cos(3*h) - size*m.cos(4*h)
        t.goto(x,y)
        t.down()

    t.end_fill()
```

[참고]

•11번째 줄 t.up()과 19번째 줄 t.down()을 주석처리하고 실행시켜보세요.

•변수 pnum과 size의 값을 달리하여 실행시켜보세요.

[3. 삼각형 판별기-triangle_checker.py, 교과서-142,146페이지] 삼각형 세 변 a, b, c의 길이를 입력 받아서, 삼각형의 종류를 출력하는 프로그램을 작성해봅시다.
(단, a, b, c는 모두 정수이며, a≤b≤c을 보장)

※ 삼각형의 종류: 정삼각형, 이등변삼각형, 직각삼각형, 일반삼각형, 삼각형이 아님

[실행 결과 예시]

세 변의 길이를 차례로 입력하시오. 6 7 8
일반삼각형
세 변의 길이를 차례로 입력하시오. 9 9 9
정삼각형
세 변의 길이를 차례로 입력하시오. 7 7 10
이등변삼각형
세 변의 길이를 차례로 입력하시오. 5 5 20
삼각형이 아님
세 변의 길이를 차례로 입력하시오. 3 4 5
직각삼각형

[코드]

```
a,b,c=map(int, input('세 변의 길이를 차례로 입력하시오.  ').split())
                          # 입력 값을 정수형으로 변환하여 변수 a,b,c에 저장
if a+b>c:                 # 삼각형의 가장 긴 변보다 나머지 두변의 길이의 합이 길어야 한다.
    if a==c:              # 정삼각형의 조건
        print('정삼각형')
    elif a==b or b==c:    # 이등변삼각형의 조건
        print('이등변삼각형')
    elif a*a+b*b==c*c:    # 직각삼각형의 조건
        print('직각삼각형')
    else:                 # 위의 3가지 조건에 부합하지 않을 경우
        print('일반삼각형')
else:
    print('삼각형이 아님') # 삼각형의 조건에 부합하지 않을 경우
```

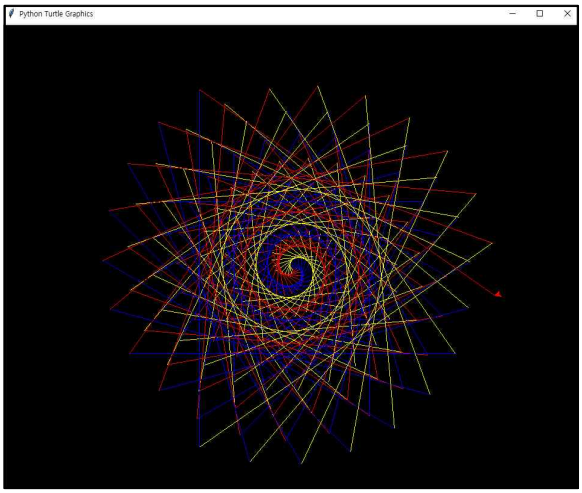
[참고]

•문제의 조건 중에 'a≤b≤c을 보장한다.'라는 부분을 제거하고 문제를 풀어보세요.
- 아래의 코드를 위의 코드 2번째 줄에 삽입하면 됩니다.

```
tmp=max(a,max(b,c))     # 입력된 값 중에서 가장 큰 값을 변수 tmp에 저장
if a==tmp: a,c=c,a      # 가장 큰 값(tmp)이 a이면, a와 c의 값을 서로 교환(swap)
elif b==tmp: b,c=c,b    # 가장 큰 값(tmp)이 b이면, b와 c의 값을 서로 교환(swap)
```

[4. 삼태극 문양 그리기-3taegeuk_draw.py, 교과서-150페이지] 반복문을 활용하여 삼태극 문양의 그림을 그리는 프로그램을 작성해봅시다.

[출력 결과]



[코드]

```
import turtle as t              # turtle 모듈을 t로 설정하여 사용

t.bgcolor("black")              # 배경색을 검은색으로 설정
t.speed(0)                      # 펜 속도를 가장 빠르게 설정(cf. 0- 가장 빠름, 1-가장 느림, 10-빠름)

for x in range(200):            # for문을 200번 실행
    if x % 3 == 0:              # 3으로 나눈 나머지가 0일 경우
        t.color("blue")         # 펜의 색깔을 파랑으로 설정
    elif x % 3 == 1:            # 3으로 나눈 나머지가 1일 경우
        t.color("red")          # 펜의 색깔을 빨강으로 설정
    else:                       # 3으로 나눈 나머지가 2일 경우
        t.color("yellow")       # 펜의 색깔을 노랑으로 설정
    t.forward(x * 3)            # 길이가 x*3인 선을 그음
    t.right(125)                # 거북이를 125도 오른쪽으로 회전합니다.
```

[참고]

•for문의 반복 횟수를 변경해보세요.

•펜의 색깔을 변경해보세요.

•선의 길이를 변경해보세요.

•회전 각도를 변경해보세요.

[5. 영문 타자 연습기-typing_training.py, 교과서-135,152페이지] 영문 타자 연습 프로그램을 작성해봅시다.
(단, 시간제한은 5초이며, 5초를 초과한 경우에는 경고를 출력)

[실행 결과 예시]

```
u -> u
time limit exceed!
ws -> ws
pass!
eegh -> eegg
fail!
bi -> bi
pass!
v -> vv
fail!
cdz ->
```

[코드]

```
import random as rn                     # random 모듈을 rn으로 설정하여 사용
import time as tm                       # time 모듈을 tm으로 설정하여 사용

a=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v
','w','x','y','z']                      # 리스트 a에 알파벳 소문자를 저장

while True:                             # break 명령어가 실행될 때까지 무한 반복
    s=rn.randint(1,5)                   # 1부터 5사이의 임의의 정수를 s에 저장
    key=''                              # 변수 key의 초기화
    for i in range(s):                  # s만큼 반복
        tmp=rn.choice(a)                # 리스트 a에서 무작위로 선택된 알파벳을 tmp에 저장
        key=key+tmp                     # 변수 key에 tmp의 알파벳을 누적
    print(key,'->',end=' ')             # key의 값을 출력하고, 출력된 값의 마지막을 공백으로 설정
    ts=tm.time()                        # 현재의 시간을 기록
    in_key=input()                      # 입력 값을 변수 in_key에 저장
    te=tm.time()                        # 현재의 시간을 기록
    if in_key=='':                      # 공백이 입력되었을 경우, break 명령어를 실행
        break
    if te-ts<=5:                        # 기록한 시간들의 차이가 5초 이하일 경우,
        if key==in_key: print('pass!')  # 타자 연습기가 제시한 영문과 입력한 영문이 같으면 pass!
        else: print('fail!')            # 다르면 fail!을 출력
    else:                               # 기록한 시간들의 차이가 5초 초과일 경우,
        print('time limit exceed!')     # time limit exceed!를 출력
```

[6. 통계값 계산기-statistics_calculator.py, 교과서-140,141,151페이지] 학생들의 수학 점수를 입력 받고, 그것에 대한 통계값(평균, 분산, 표준편차)을 구하는 프로그램을 작성해봅시다.

[실행 결과 예시]

```
학생들의 수학 점수를 하나씩 입력하고 엔터를 누르세요.
입력이 끝나면, 엔터를 한 번 더 눌러주세요.
50
60
70
45
83
37
97
59
66
48

평균 =  61.5
분산 =  301.05
표준편차 =  17.350792489105505
```

[코드]

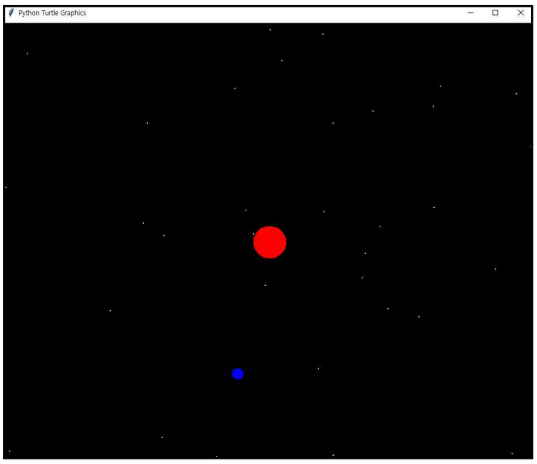
```
import math as m                # math 모듈을 m으로 설정하여 사용
a=[]                            # 리스트 선언
print('학생들의 수학 점수를 하나씩 입력하고 엔터를 누르세요.')
print('입력이 끝나면, 엔터를 한 번 더 눌러주세요.')
while True:                     # break 명령어가 실행될 때까지 무한 반복
    tmp=input()                 # 입력 값을 tmp에 저장
    if tmp=='': break           # 공백이 입력되었을 경우, break 명령어 실행
    else: a.append(int(tmp))    # tmp값이 공백이 아닐 경우, 정수로 변환하여 리스트 a에 추가
avg=sum(a)/len(a)               # 평균 = 리스트 a의 총합 / 리스트 a의 길이
tsum=0                          # 변수 tsum의 초기화
for i in a:                     # 리스트의 데이터 개수만큼 반복
    tsum=tsum+(i-avg)**2        # 각 데이터 값과 평균값의 차이를 제곱한 값을 누적
var=tsum/len(a)                 # 위에서 누적한 값을 데이터 개수로 나누면 분산
std=m.sqrt(var)                 # 분산의 제곱근을 구하는 함수(sqrt)를 사용
print('평균 = ',avg)            # 통계값 출력
print('분산 = ',var)
print('표준편차 = ',std)
```

[참고]

•데이터의 입력 방식을 좀 더 편리하게 변경해보면 어떨까요?

[7. 태양 주위를 공전하는 지구-sun_earth.py, 교과서-153,156,164페이지] 태양 주위를 공전하는 지구를 표현할 수 있는 프로그램을 작성해봅시다.

[실행 결과 예시]



[참고]

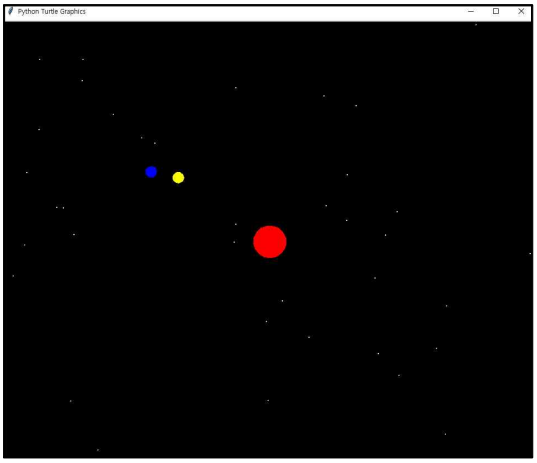
- 수학시간에 학습한 삼각함수(sin(),cos())에 대해 간단하게 언급을 하는 것이 좋을 것 같아요.
- 지구(파란색 원)의 공전 궤도를 확인하고 싶다면, 아래의 코드를 23번째 줄 아래에 삽입해보세요.
  - t.pendown() # 펜 내리기
- t.goto(x,y) 대신에 t.setpos(x,y)를 사용할 수도 있어요.
- 지구(파란색 원)가 태양 주위를 1회 공전하는 동안 지구 주위를 12회 공전하는 달도 함께 표현하고 싶으면 어떻게 하면 될까요?

[코드]

```
1  import turtle as t                  # turtle 모듈을 t로 설정하여 사용
2  import math as m                    # math 모듈을 m으로 설정하여 사용
3  import random as rn                 # random 모듈을 rn으로 설정하여 사용
4  def star(r,c,x,y):                  # star함수(반지름 r, 색깔이 c, 위치가 x, y인 원 그리기)
5      t.penup()                       # 펜(거북이) 들기
6      t.goto(x,y)                     # (x,y) 위치로 이동
7      t.color(c)                      # 펜 색깔을 c로 설정
8      t.begin_fill()                  # 영역 채우기 시작
9      t.circle(r)                     # 반지름이 r인 원 그리기
10     t.end_fill()                    # 영역 채우기 끝
11 t.bgcolor("black")                  # 배경색을 검은색으로 설정
12 t.speed(0)                          # 펜 속도를 가장 빠르게 설정(cf. 0- 가장 빠름, 1-가장 느림, 10-빠름)
13 for i in range(50):                 # 임의의 위치에 반지름이 1인 하얀색 원 그리기 50번 반복(별)
14     star(1,'white',rn.randint(-500,500),rn.randint(-500,500))
15 star(30,'red',0,-30)                # 화면의 가운데 지점에 반지름이 30인 빨간색 원 그리기(태양)
16 t.shape('circle')                   # 펜의 모양을 원으로 설정(지구)
17 t.color('blue')                     # 펜의 색깔을 파랑으로 설정
18 r=250                               # 지구의 공전 궤도 반지름을 250으로 설정
19 i=0                                 # 변수 i 초기화
20 while True:                         # 지구의 공전을 표현하기 위해 무한 반복
21     x=m.cos(i)*r                    # 삼각함수를 적용하여 x,y의 위치를 결정
22     y=m.sin(i)*r
23     t.goto(x,y)                     # x,y의 위치로 이동
24     i=i+0.01                        # 변수 i 0.01씩 증가(회전각도 증가)
```

[8. 태양 주위를 공전하는 지구와 달-sun_earth_moon.py, 교과서-153,156,164페이지] 태양 주위를 공전하는 지구와 지구 주위를 공전하는 달을 표현할 수 있는 프로그램을 작성해봅시다.

[실행 결과 예시]



[참고]

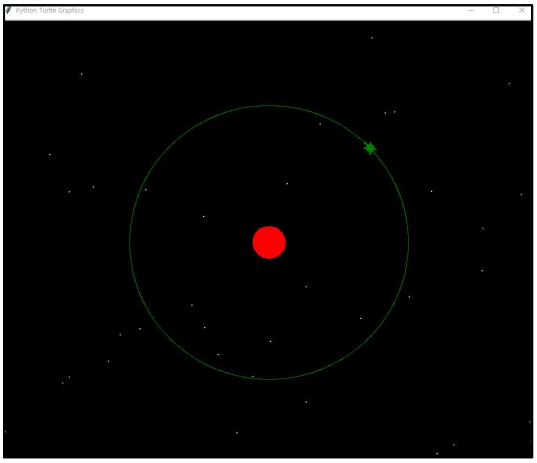
- 태양 주위를 공전하는 지구를 실습한 이후에 이어서 심화된 내용으로 실습하시면 좋을 것 같아요.
- 달의 위치(x1,y1)에 대한 설명을 자세하게 할 필요가 있어요. (코드 22번째 줄)
- 지구의 1회 공전(1년)시에 달은 12회 공전(12개월)하도록 설정되어있어요.
- 지구와 달의 공전 궤도를 확인하고 싶다면, 아래의 코드를 23번째 줄 아래에 삽입해보세요.
  - `t.pendown(); t1.pendown()`
- `t.setpos(x,y)` 대신에 `t.goto(x,y)`를 사용할 수도 있어요.

[코드]

```
1  import turtle as t
2  import math as m
3  import random as rn
4  def star(r,c,x,y):
5      t.penup(); t.setpos(x,y); t.color(c); t.begin_fill(); t.circle(r); t.end_fill()
6  t.bgcolor("black"); t.speed(0)
7  for i in range(50):
8      star(1,'white',rn.randint(-500,500),rn.randint(-500,500))
9  star(30,'red',0,-30)
10 t.shape('circle')
11 t.color('blue')
12 r=250
13 i=0
14 t1=t.Turtle()                                  # 달을 표현하기 위해 펜(거북이) 추가
15 t1.speed(0); t1.shape('circle'); t1.color('yellow'); t1.penup()
16 r1=50
17 i1=0
18 while True:                                    # 지구와 달의 공전을 표현하기 위해 무한 반복
19     x=m.cos(i)*r; y=m.sin(i)*r                 # 지구의 위치
20     t.setpos(x,y)
21     i=i+0.01
22     x1=x + m.cos(i1)*r1; y1=y + m.sin(i1)*r1   # 태양을 중심으로 한 달의 위치 =
23     t1.setpos(x1,y1)                           # (지구의 위치 + 지구를 중심으로 한 달의 위치)
24     i1=i1+0.12                                 # 지구보다 12배 빨리 공전하도록 설정(12개월을 표현)
```

[9. 태양 주위를 공전하는 거북이-sun_turtle.py, 교과서-153,156,164페이지] 태양 주위를 공전하는 거북이를 표현해봅시다.
(단, 거북이의 머리 방향이 공전 궤도와 일치하도록 설정)

[실행 결과 예시]



[참고]

- 거북이의 공전과 머리 방향의 회전 비율(0.573)을 알려주지 마시고, 학생들이 직접 찾을 수 있도록 하는 것이 좋을 것 같아요.
- 거북이가 태양 주변을 1회 공전하는 동안에 거북이의 자전을 표현하고 싶다면, 22번째 줄의 코드를 아래의 코드로 수정해보세요.
  - h=h+(0.573*(원하는 자전횟수+1))

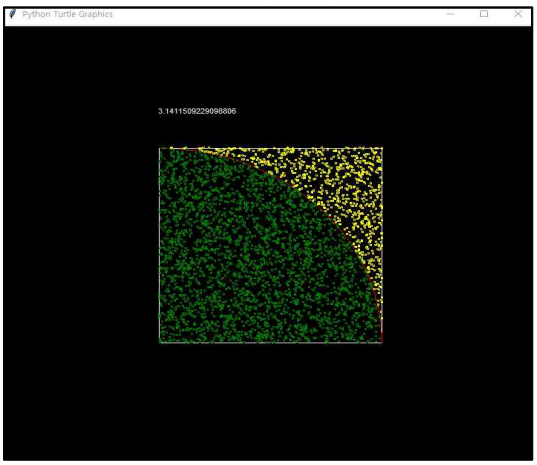
원하는 자전횟수에 1을 더하는 이유는 거북이가 공전을 하면서 기본적으로 자전을 1번하기 때문입니다.

[코드]

```
1  import turtle as t
2  import math as m
3  import random as rn
4  def star(r,c,x,y):
5      t.penup(); t.setpos(x,y); t.color(c); t.begin_fill(); t.circle(r); t.end_fill()
6  t.bgcolor("black")
7  t.speed(0)
8  for i in range(50):
9      star(1,'white',rn.randint(-500,500),rn.randint(-500,500))
10 star(30,'red',0,-30)
11 t.shape('turtle')              # 펜의 모양을 거북이로 설정
12 t.color('green')
13 r=250
14 i=0
15 h=90                           # 거북이의 머리 방향을 지정하는 변수(오른쪽-0,위쪽-90)
16 while True:
17     x=m.cos(i)*r
18     y=m.sin(i)*r
19     t.setpos(x,y)
20     t.setheading(h)            # 거북이의 머리 방향을 h로 설정
21     i=i+0.01
22     h=h+0.573                  # 거북이의 공전과 머리 방향의 회전 비율 = 약 0.573
23     t.pendown()                # 펜 내림(거북이의 공전 궤도를 확인 할 수 있음)
```

[10. 몬테카를로 방법을 이용하여 원주율 구하기-monte_carlo_pi.py, 교과서-131,152페이지] 몬테카를로 방법을 이용하여 원주율을 구하는 프로그램을 작성해봅시다.

[실행 결과 예시]



[참고]

- 먼저 몬테카를로 방법에 대해 검색을 해보시면 좋을 것 같아요.
- π 값 계산 식
  - 사각형 넓이 : 부채꼴의 넓이 = cnt_total : cnt_in
  - r^2 : π*r^2*1/4 = cnt_total : cnt_in
  - π*r^2*1/4 * cnt_total = r^2 * cnt_in
  - π = 4 * cnt_in / cnt_total

[코드]

```
import turtle as t; import random as rn; import math as m   # 각종 모듈 설정
t.bgcolor('black'); t.speed(0); t.color('white'); t.penup(); t.hideturtle()
t.goto(150,-150); t.pendown()
t.goto(150,150);t.goto(-150,150);t.goto(-150,-150);t.goto(150,-150)  # 사각형 그리기
tw=t.Turtle()                # 문자를 표현하기 위한 객체
tw.penup();tw.hideturtle();tw.goto(-150,200);tw.color('white')
tw.write('hello!',font=(11))  # 문자 출력
t.color('red');i=0
while True:                  # 부채꼴 모양 그리기
    x=300*m.cos(i)-150; y=300*m.sin(i)-150
    t.goto(x,y)
    i=i+0.01
    if -150<=x<=-149 and 149<=y<=150: break
cnt_in=0;cnt_total=0;pi=0
while True:
    t.penup()
    tw.write(pi,font=(11))
    x=rn.randint(-150,150); y=rn.randint(-150,150) # 무작위로 x,y의 좌표를 설정
    cnt_total+=1
    if m.sqrt((x+150)**2 + (y+150)**2)<=300:  # 부채꼴 안의 점이면,
        cnt_in+=1                             # 초록색 점(작은 원)으로 표현
        t.goto(x,y);t.pendown();t.color('green');t.circle(1)
    else:                                     # 부채꼴 밖의 점이면, 노란색 점으로 표현
        t.goto(x,y);t.pendown();t.color('yellow');t.circle(1)
    pi=4*(cnt_in/cnt_total)   # 화면에 찍힌 점들의 개수를 파악하여 π 값을 유추
    tw.clear()
```

[11. nCr(조합) 계산기-combination_calculator.py, 교과서-166페이지] nCr(조합)을 계산하는 프로그램을 작성해봅시다.
(단, n과 r은 1이상 20이하의 정수이고, n>=r)

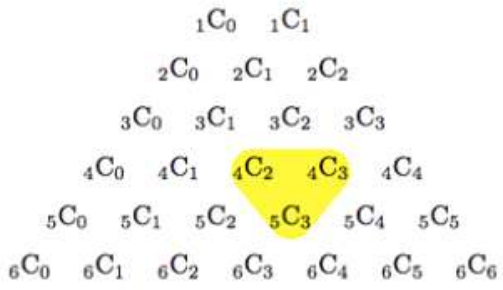
[실행 결과 예시]
n값을 입력해주세요.20
r값을 입력해주세요.5
15504

[코드]

```
print('nCr(조합) 계산기입니다.')
n=int(input('n값을 입력해주세요.'))
r=int(input('r값을 입력해주세요.'))

def combi(n,r):                          # 조합 계산을 위한 함수 선언
    if n==r: return 1                    # n과 r의 값이 같으면 1을 반환
    elif r==1: return n                  # r이 1이면 n을 반환
    return combi(n-1,r-1)+combi(n-1,r)   # 그 이외에는 재귀함수 호출

print(combi(n,r))
```

[참고]

•파스칼의 삼각형을 조합 계산에 활용



$$_nC_r = {}_{n-1}C_{r-1} + {}_{n-1}C_r$$

예) $_5C_3 = {}_4C_2 + {}_4C_3$

- 위와 같이 nCr(조합)을 계산하기 위한 방법으로 파스칼의 삼각형을 활용할 수 있어요.
- 파스칼의 삼각형을 활용하여 표현된 점화식은 재귀함수를 활용하여 코드로 작성될 수 있지요.
- 점화식에 대한 참조 : https://terms.naver.com/entry.nhn?docId=3338015&cid=47324&categoryId=4732
- n과 r의 값을 20보다 더 높은 값으로 입력을 해보세요. 예) n=50, r=20
- n=50, r=20을 입력하면, 계산하는데 굉장히 오랜 시간이 걸리죠.
- 이러한 문제를 해결하기 위해서 '메모이제이션(memoization)'을 활용할 수 있어요.

[12. nCr(조합) 계산기2-combination_calculator(memoization).py, 교과서-166,173페이지]
nCr(조합)을 계산하는 프로그램을 작성해봅시다.
(단, n과 r은 1이상 100이하의 정수이고, n>=r)

[실행 결과 예시]
n값을 입력해주세요.100
r값을 입력해주세요.30
293723398216109448239637 60

[코드]

```
print('nCr(조합) 계산기입니다.')
n=int(input('n값을 입력해주세요.'))
r=int(input('r값을 입력해주세요.'))

memo=[[0]*101 for i in range(101)]          # 101*101인 2차원 배열의 선언(0으로 초기화)
                                            # 메모이제이션을 활용하기 위한 배열임
def combi(n,r):                             # 조합 계산을 위한 함수 선언
    if memo[n][r]!=0: return memo[n][r]     # 배열에 저장되어 있는 값이 있다면, 그 값을 반환
    elif n==r: return 1                     # n과 r의 값이 같으면 1을 반환
    elif r==1: return n                     # r이 1이면 n을 반환
    memo[n][r]=combi(n-1,r-1)+combi(n-1,r) # 그 이외에는 재귀함수 호출 및 계산된 값을 배열에 저장
    return memo[n][r]

print(combi(n,r))
```

[참고]

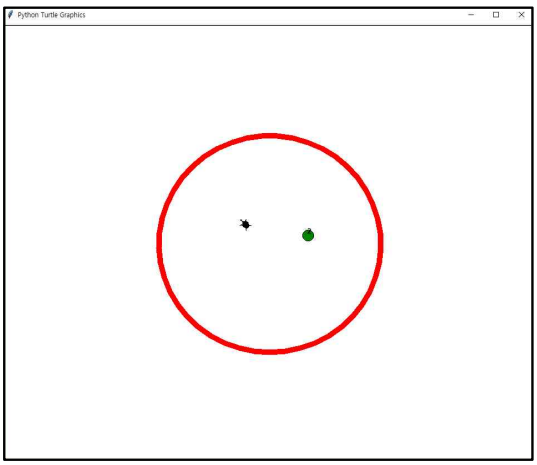
• 동적 프로그래밍(dynamic programming)
- 큰 문제의 해답에 작은 문제의 해답이 포함되어 있고 이를 재귀호출 알고리즘으로 구현하면 지나친 중복이 발생하는 경우, 이 재귀적 중복을 해결하는 데 사용되는 방법
- 중복된 재귀호출을 피하는 동적 프로그래밍을 특별히 메모이제이션(memoization)이라고 함

• 메모이제이션(memoization)
- 컴퓨터 프로그램이 동일한 계산을 반복해야 할 때, 이전에 계산한 값을 메모리에 저장함으로써 동일한 계산의 반복 수행을 제거하여 프로그램 실행 속도를 빠르게 하는 기술

- 쉽게 배우는 알고리즘(문병로 저)

[13. 먹이를 사냥하는 거북이-turtles_hunting.py, 교과서-156페이지] 먹이를 사냥하는 거북이를 표현해봅시다.
(단, 먹이는 총 5개이며 각각의 먹이는 순차적으로 임의의 장소에서 등장)

[실행 결과 예시]



[참고]

- 감각(시각)이 없는 거북이가 임의의 방향으로 직진하면서 우연히 먹이를 사냥하는 것을 표현한 프로그램이에요.
- 만약, 거북이에게 감각(시각)을 선물 한다면 먹이를 좀 더 쉽게 사냥할 수 있겠죠.
- 거북이에게 감각(시각)을 아래와 같이 선물해봅시다.
- 아래의 코드를 12번째 줄 아래에 삽입.

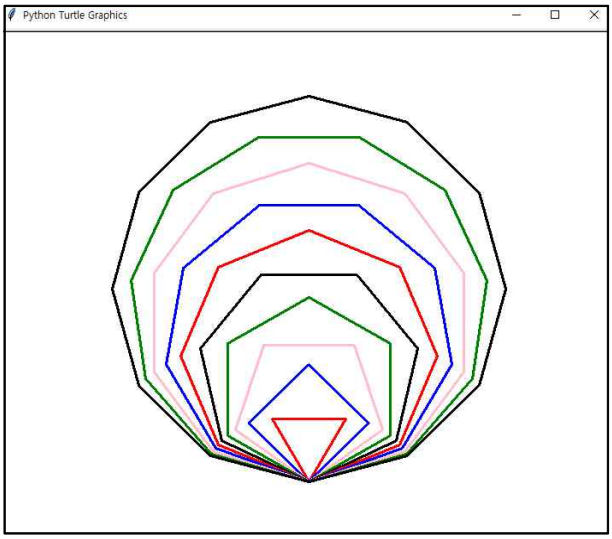
```
1  a=t.towards(fd.pos())  # 거북이가 먹이를 바라보는 각도를 변수 a에 저장
2  t.setheading(a)  # 거북이가 바라보는 방향 설정
```

[코드]

```
1  import turtle as t; import random as rn  # turtle 모듈과 random 모듈을 각각 t, rn으로 설정
2  t.shape('turtle'); t.ht(); t.speed(0); t.penup(); t.goto(0,-200); t.pendown();
3  t.pensize(10); t.pencolor('red'); t.circle(200)
4                                          # 거북이의 활동 영역(울타리)을 지정하기 위한 원 그리기
5                                          # ht()함수는 hideturtle()함수와 동일
6  cnt=1; fd=t.Turtle(); fd.shape('circle'); fd.penup(); fd.speed(0); fd.color('green');
7  fd.goto(100,-100); fd.pencolor('black'); fd.write(cnt,font=('arial',10,'bold'))
8                                          # fd는 먹이, 변수 cnt는 먹이의 개수
9  t.penup(); t.st(); t.pensize(20); t.pencolor('white'); t.goto(0,0); t.speed(0); t.pendown()
10                                         # st()함수는 showturtle()함수와 동일, t는 거북이
11 while cnt<=5:                           # 5번의 먹이사냥이 끝날 때까지 반복
12     t.forward(2)                        # 2만큼씩 전진(인수를 증가시킬수록 거북이 속도 증가)
13     if t.distance(0,0)>=170:            # 거북이가 울타리에 도달한 경우
14         t.left(rn.randint(10,40))       # 10~40 사이의 임의의 정수만큼 왼쪽으로 회전
15     if t.distance(fd)<20:               # 거북이가 먹이에 도달한 경우
16         fd.undo()                       # 먹이를 지움(undo()함수는 직전 실행을 취소함)
17         while True:                     # break 명령어가 실행될 때까지 무한 반복
18             x=rn.randint(-200,200)      # 다음 먹이의 x좌표
19             y=rn.randint(-200,200)      # 다음 먹이의 y좌표
20             fd.goto(x,y)                # 다음 먹이의 위치로 이동
21             if fd.distance(0,0)<180:    # 다음 먹이의 위치가 울타리 안에 있을 경우
22                 cnt+=1                  # 먹이의 개수를 증가
23                 fd.write(cnt,font=('arial',10,'bold'))
24                 break                   # 다음 먹이의 위치를 선정하기 위한 반복문 탈출
25             fd.undo()                   # 다음 먹이의 위치가 울타리 밖에 있을 경우, 실행 취소
```

[14. 다양한 도형 그리기-various_figure.py, 교과서-133,148,158페이지] 반복문을 활용하여 다양한 도형을 그리는 프로그램을 작성해봅시다.

[실행 결과 예시]



[코드]

```
import turtle as t                          # turtle 모듈을 t로 설정하여 사용
a=['red','blue','pink','green','black']    # 리스트 변수 a에 다양한 색깔을 저장
t.ht()                                      # hideturtle()과 같은 함수로서 펜(거북이)을 숨김
t.pensize(3)
t.penup()
t.goto(0,-200)
t.pendown()
size=50
step=3
for i in range(10):                         # 총 10개의 도형을 그림
    t.pencolor(a[i%5])                      # 색깔이 총 5개이므로 5로 나눈 나머지를 인수로 사용
    t.circle(size,steps=step)                 # 아래의 '참고'를 참고
    size+=20                            # 도형의 크기를 증가
    step+=1                             # 도형의 꼭짓점 개수를 증가
```

[참고]

- circle()함수는 size와 steps를 파라미터로 사용해요. 예) turtle.circle(50,steps=3)
- 여기서, size는 도형의 중심으로부터 각 꼭짓점에 이르는 길이를 뜻하고,
- steps는 도형의 꼭짓점 개수를 뜻해요.
- turtle.circle(50,steps=3)을 실행하면, 중심으로부터 3개의 꼭짓점에 이르는 길이가 50인 삼각형을 그리게 되겠죠.

[15. 단어 검색기-search_machine.py, 교과서-143,144페이지] 파일에 저장되어 있는 내용(문자) 중에서 특정 단어의 위치와 총 개수를 알려주는 프로그램을 작성해봅시다.

[실행 결과 예시]

```
검색할 단어를 입력하세요.이
1 번째 줄
2 번째 줄 9 22
3 번째 줄 27
4 번째 줄
5 번째 줄 27
6 번째 줄 19
7 번째 줄 27
8 번째 줄 1 7
9 번째 줄 27
총 9 개입니다.
```

[실행에 사용된 test.txt파일의 내용]

```
[애국가]
동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세
무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세
남산 위에 저 소나무 철갑을 두른 듯 바람서리 불변함은 우리 기상 일세
무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세
가을 하늘 공활한데 높고 구름 없이 밝은 달은 우리 가슴 일편단심 일세
무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세
이 기상과 이 맘으로 충성을 다하여 괴로우나 즐거우나 나라 사랑하세
무궁화 삼천리 화려 강산 대한 사람 대한으로 길이 보전하세
```

[코드]

```
file=open('c:/test.txt','r')            # open()함수를 사용하여 file 객체 생성(읽기 모드)
text=input('검색할 단어를 입력하세요.') # 검색할 단어를 변수 text에 저장
line=0; cnt=0                           # 파일 내용의 줄번호(line)와 검색 개수(cnt) 초기화
while True:                             # break 명령어가 실행될 때까지 무한 반복
    content=file.readline()             # 파일의 첫줄부터 순서대로 한 줄씩 content에 저장
    if content=='': break               # 만약, content가 공백(마지막)이라면 종료
    else: line+=1; s=0; ss=1            # 아니면 줄번호 1증가, 부분 위치(s), 전체 위치(ss) 초기화
    print(line,'번째 줄 ',end='')
    while content:                      # content에 값이 있는 동안 반복
        content=content[s:]             # 부분 위치(s)를 기준으로 슬라이싱
        n=content.find(text)            # find()함수를 사용하여 검색할 단어(text)의 위치를 변수 n에 저장
        if n==-1: break                 # 만약, n의 값이 -1(검색 단어가 없는 경우)이면 종료
        else :                          # 아니면(검색 단어가 있다면),
            print(ss+n,end=' ')         # 이전 검색의 전체 위치(ss)에 현재 검색의 위치(n)을 더한 값을 출력
            cnt+=1                      # 검색 개수 증가
            s=n+len(text); ss+=s        # 부분 위치(s)와 전체 위치(ss)의 최신화
    print()                             # 줄바꿈
print('총',cnt,'개입니다.')
```

[참고]

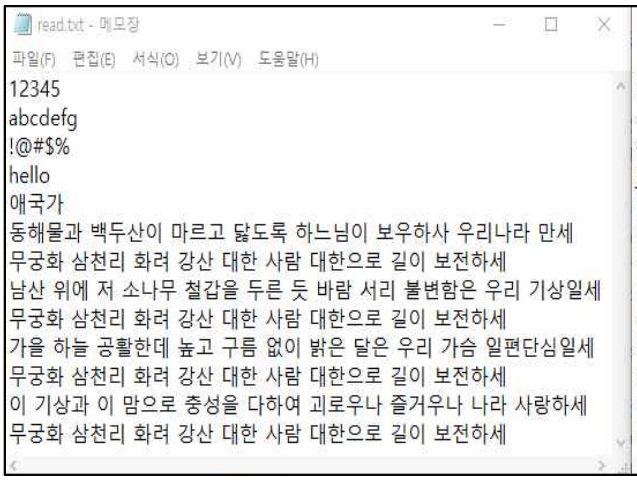
- find()함수는 검색하고자하는 문자의 첫 번째 위치(인덱스)를 반환하거나, 검색하고자하는 문자가 없을 경우에는 -1을 반환해요. 따라서, 전체 내용을 모두 검색하기 위해서는 위의 코드와 같이 조금 복잡한 방법을 사용해야합니다.

[16. 암호의 생성 혹은 해독-cryptograph.py, 교과서-143,144페이지] 파일에 저장되어 있는 내용을 암호화하거나, 암호화되어 있는 내용을 해독하여 다른 파일에 저장하는 프로그램을 작성해봅시다.

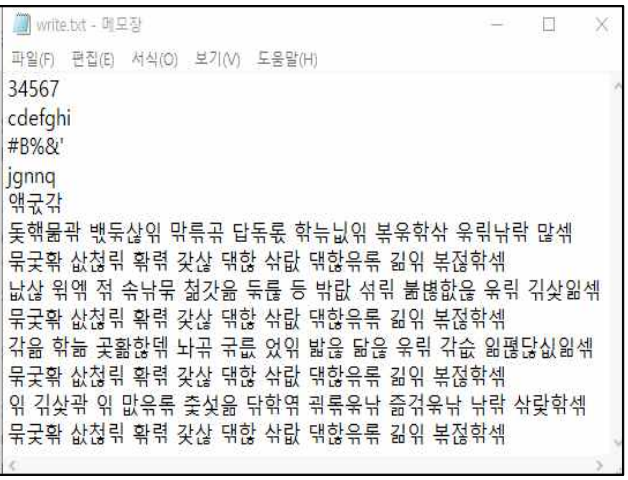
[실행 결과 예시]

암호의 key값을 입력하세요. 2

(원본)



(암호)



[코드]

```
1  file=open('read.txt','r')     # open()함수를 사용해 'r' 모드(읽기) 객체 생성
2  file1=open('write.txt','w')   # open()함수를 사용해 'w' 모드(쓰기) 객체 생성
3     # 읽고, 쓰기 위한 text파일이 코드와 같은 폴더에 저장되어 있는 경우, 파일명만으로 가능
4  key=int(input('암호의 key값을 입력하세요. '))
5  a=file.read()                 # read()함수를 사용해 file 객체에 있는 내용(read.txt) 저장
6  b=''                          # 문자열 변수 b 선언 및 초기화
7  for i in a:                   # read.txt 파일의 모든 내용을 대상으로 1개 문자씩 암호화 작업 수행
8      if i==' ': b=b+' '        # 한 칸 띄움(스페이스)일 경우, 암호화하지 않고 그대로 출력
9      elif i=='\n': b=b+'\n'    # 한 줄 띄움(개행)일 경우, 암호화하지 않고 그대로 출력
10     else: b=b+chr(ord(i)+key) # 원본 문자를 아스키코드 값(정수)으로 변환(ord)하여 암호화
11                                 (key값과 더함)한 후에 다시 문자(chr)로 변경 후 b에 저장
12 file1.write(b)                # 암호화 된 내용(b)을 'write.txt' 파일에 저장
13 file.close()                  # 파일 닫기 수행
14 file1.close()                 # 파일을 닫지 않을 경우, 'write.txt.' 파일에 저장이 안됨
```

[참고]

- ord()함수 : 문자를 받아서 그 문자의 아스키코드 값(정수)을 반환
- chr()함수 : 아스키코드 값(정수)를 받아서 그 값에 해당하는 문자로 반환

[17. 요리의 완료 시각을 알려주는 시간 계산기-cook_time.py, 교과서-132,135페이지]
요리하는데 걸리는 시간(분 단위)을 입력하면, 완료 시각을 알려주는 프로그램을 작성해봅시다. (단, 요리 시간(분 단위)는 최대 10,000분을 넘지 않음)

[실행 결과 예시]

```
현재 시각
2018년 8월 26일
0시 56분

요리 시간(분)을 알려주세요. 10000

완료 시각
2018년 9월 1일
23시 36분
```

[참고]

- 아래의 코드는 윤년의 여부를 고려하지 않은 것이에요.
- 윤년의 여부를 고려하는 코드를 추가해보면 좋을 것 같아요.
  - 윤년의 조건(윤년-2월 달의 날 수가 29일)
    1) 연도의 수가 4로 나누어떨어질 경우, 윤년(29일)
    2) 1) 중에서 100으로 나누어떨어질 경우, 평년(28일)
    3) 2) 중에서 400으로 나누어떨어질 경우, 윤년(29일)
  - 아래의 코드를 13번째 줄에 삽입

```
1  if (y%4==0 and y%100!=0) or y%400==0:
2  c[1]=29
```

[코드]

```
1  import time as t                # time 모듈을 t로 설정하여 사용
2  time=t.localtime()              # localtime() 함수는 컴퓨터의 현재 시각(년,월,일,시,분,초)을 반환
3  y=time.tm_year                  # 현재 시각의 년을 변수 y에 저장
4  mon=time.tm_mon                 # 현재 시각의 달을 변수 mon에 저장
5  d=time.tm_mday                  # 현재 시각의 날을 변수 d에 저장
6  h=time.tm_hour                  # 현재 시각의 시를 변수 h에 저장
7  m=time.tm_min                   # 현재 시각의 분을 변수 m에 저장
8  print('현재 시각')
9  print('{}년 {}월 {}일'.format(y,mon,d))
10 print('{}시 {}분'.format(h,m))
11 ct=int(input('\n요리 시간(분)을 알려주세요. '))
12 cal=h*60+m+ct                   # 현재 시각을 분으로 환산한 뒤에 요리 시간을 합산
13 c=[31,28,31,30,31,30,31,31,30,31,30,31]
14 tmp=c[mon-1]                    # 리스트의 인덱스는 0부터 시작하기 때문에 mon-1을 해줌
15 d+=int(cal/1440)                # 합산된 요리 완료 시각에 하루(1440분)를 나눔
16 mon+=int(d/tmp)                 # 날이 해당 달의 최대 날수(tmp)를 넘은 경우, 월을 증가시키고
17 y+=int((mon-1)/12)              # 증가된 월이 12월을 넘었다면, 년을 증가시킴
18 if mon%12==0: mon=int(12)       # 만약, 월이 12의 배수라면, 월을 12로 저장
19 else: mon=int(mon%12)           # 아니면, 월을 12로 나눈 나머지를 월로 저장
20 if d%tmp==0: d=int(tmp)         # 만약, 날이 tmp의 배수라면, 날을 tmp로 저장
21 else: d=int(d%tmp)              # 아니면, 날을 tmp로 나눈 나머지를 날로 저장
22 h=int((cal%1440)/60)            # 요리 완료 시각에 시간(60분)을 나누어 몫은 시간으로
23 m=(cal%1440)%60                 # 나머지는 분으로 저장
24 print('\n완료 시각')
25 print('{}년 {}월 {}일'.format(y,mon,d))
26 print('{}시 {}분'.format(h,m))
```

[18. D-day 계산기-d_day.py, 교과서-132,135페이지] D-day를 입력하면, 지금부터 며칠이 남았는지를 알려주는 D-day 계산기 프로그램을 작성해봅시다.
(단, 년은 최대 10000년까지로 함)

[실행 결과 예시]

현재 날짜: 2018년 8월 27일
D-day의 년, 월, 일을 공백으로 구분하여 입력하세요. 5555 10 16
오늘은 D-1291912일 입니다.

[코드]

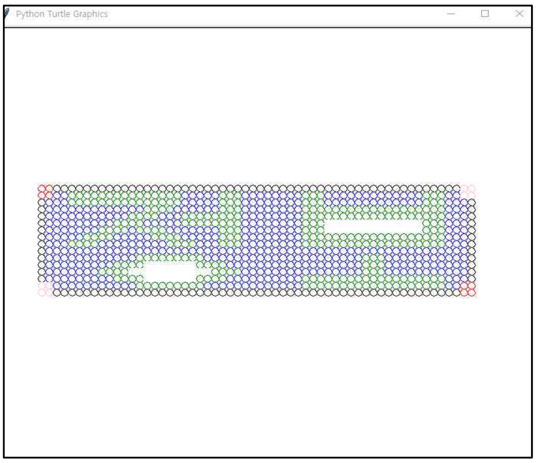
```
import time as t                # time 모듈을 t로 설정하여 사용
time=t.localtime()              # localtime() 함수는 컴퓨터의 현재 시각(년,월,일,시,분,초)을 반환
y=time.tm_year                  # 현재 시각의 년을 변수 y에 저장
m=time.tm_mon                   # 현재 시각의 달을 변수 m에 저장
d=time.tm_mday                  # 현재 시각의 날을 변수 d에 저장
print('현재 날짜: {}년 {}월 {}일'.format(y,m,d))
yy,mm,dd=map(int, input('D-day의 년, 월, 일을 공백으로 구분하여 입력하세요. ').split())
a=[0]*10000                     # 크기가 10000, 초기값이 0인 리스트 선언
s=0                             # 윤년의 개수를 저장하는 변수 s 초기화
for i in range(1,10000):        # 1부터 10000까지 윤년의 누적합을 저장하는 반복문
    if (i%4==0 and i%100!=0) or i%400==0: s+=1
    a[i]=s
b=[0,31,59,90,120,151,181,212,243,273,304,334,365] # 월별 일수의 누적합
if (y%4==0 and y%100!=0) or y%400==0:     # 현재의 년이 윤년일 경우,
    for i in range(2,13): b[i]+=1         # 리스트 b[2](2월)부터 끝(12월)까지 1씩 더함
p=(y-1)*365+a[y-1]+b[m-1]+d               # 0년0월0일부터 현재까지의 일수를 계산
b=[0,31,59,90,120,151,181,212,243,273,304,334,365]
if (yy%4==0 and yy%100!=0) or yy%400==0:  # D-day의 년이 윤년일 경우,
    for i in range(2,13): b[i]+=1         # 리스트 b[2](2월)부터 끝(12월)까지 1씩 더함
q=(yy-1)*365+a[yy-1]+b[mm-1]+dd           # 0년0월0일부터 D-day까지의 일수를 계산
print('오늘은 D-{}일 입니다.'.format(q-p))
```

[참고]

- 윤년이란? 2월달의 일수가 29일이 되는 해를 뜻해요.
- 윤년의 조건
  1) 연도의 수가 4로 나누어떨어질 경우, 윤년(29일)
  2) 1) 중에서 100으로 나누어떨어질 경우, 평년(28일)
  3) 2) 중에서 400으로 나누어떨어질 경우, 윤년(29일)

[19. 픽셀 그리기-pixel_rendering.py, 교과서-138,144페이지] 픽셀 그리기 프로그램을 작성해봅시다.

[실행 결과 예시]



[참고] - pixel.txt 파일의 내용

```
rrkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkpp
rrbbgggggggggggggggbbbbbgggbbbbbbbbgggbbbbbbbbbbbbbgggbbpp
kbbbgggggggggggggggbbbbbgggbbbbbbbbgggbbbbbbbbbbbbbgggbbbk
kbbbbbbbbbbbbbbggggbbbbbgggbbbbbbbbgggggggggggggggggggbbbk
kbbbbbbbbbbbggggbbbggggggggbbbbbbbbgggggggggggggggggggbbbk
kbbbbbbbbbbggggbbbbggggggggbbbbbbbbgggwwwwwwwwwwwwwgggbbbk
kbbbbbbbggggbggggbbbbbbgggbbbbbbbbgggwwwwwwwwwwwwwgggbbbk
kbbbbbggggbbbbggggbbbbbgggbbbbbbbbgggggggggggggggggggbbbk
kbbbggggbbbbbbbbggggbbgggbbbbbbbbgggggggggggggggggggbbbk
kbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbk
kbbbbbbbbbbbbgggggggggbbbbbbbbbbbbbbbbbbbbbgggbbbbbbbbbbbk
kbbbbbbbbbggggwwwwwwwwggggbbbbbbbbbbbbbbbbbgggbbbbbbbbbbbk
kbbbbbbbggggwwwwwwwwwwwggggbbbbbbbbbbbbbbbbgggbbbbbbbbbbbk
kbbbbbbbbbggggwwwwwwwggggbbbbbbbbbbgggggggggggggggggggbbbk
ppbbbbbbbbbbbgggggggggbbbbbbbbbbbbbgggggggggggggggggggbbrr
ppkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkrr
```

[코드]

```
import turtle as t               # turtle 모듈을 t로 설정하여 사용
file=open('pixel.txt','r')          # open()함수를 사용해 ‘r’모드(읽기) 객체 생성
a=file.read()                    # read()함수를 사용해 file 객체에 있는 내용 저장
t.speed(0)
t.ht()                           # ht() 함수 - 거북이를 숨김(=hideturtle())
def draw(x,y,c):                 # draw() 함수 정의
    t.color(b[c])
    t.penup()
    t.goto(x,y)
    t.pendown()
    t.circle(5)
b={'b':'blue','p':'pink','r':'red','w':'white','g':'green','k':'black'}
x=-300;y=100                     # 시작 위치의 좌표
for i in a:                      # 문자열 변수 a에 대한 반복문
    if i=='\n':                  # 만약, 개행(줄바꿈) 문자라면,
        x=-300                   # x좌표는 출발 위치로 이동
        y-=10                    # y좌표 한줄 아래로 이동
    else:
        draw(x,y,i)              # draw() 함수 호출
        x+=10                    # 우측으로 이동
```

[20. 문자열 압축기-run_length.py, 교과서-144,148페이지] 문자열을 압축하는 프로그램을 작성해봅시다.

[실행 결과 예시]

```
r*2 k*54 p*2
r*2 b*2 g*15 b*5 g*3 b*8 g*3 b*13 g*3 b*2 p*2
k*1 b*3 g*15 b*5 g*3 b*8 g*3 b*13 g*3 b*3 k*1
k*1 b*13 g*4 b*6 g*3 b*8 g*19 b*3 k*1
k*1 b*11 g*4 b*3 g*8 b*8 g*19 b*3 k*1
k*1 b*9 g*4 b*5 g*8 b*8 g*3 w*13 g*3 b*3 k*1
k*1 b*7 g*4 b*1 g*4 b*7 g*3 b*8 g*3 w*13 g*3 b*3 k*1
k*1 b*5 g*4 b*5 g*4 b*5 g*3 b*8 g*19 b*3 k*1
k*1 b*3 g*4 b*9 g*4 b*3 g*3 b*8 g*19 b*3 k*1
k*1 b*56 k*1
k*1 b*12 g*9 b*21 g*3 b*11 k*1
k*1 b*9 g*4 w*7 g*4 b*18 g*3 b*11 k*1
k*1 b*7 g*4 w*11 g*4 b*16 g*3 b*11 k*1
k*1 b*9 g*4 w*7 g*4 b*10 g*19 b*3 k*1
p*2 b*11 g*9 b*13 g*19 b*2 r*2
p*2 k*54 r*2
```

[참고] - pixel.txt 파일의 내용

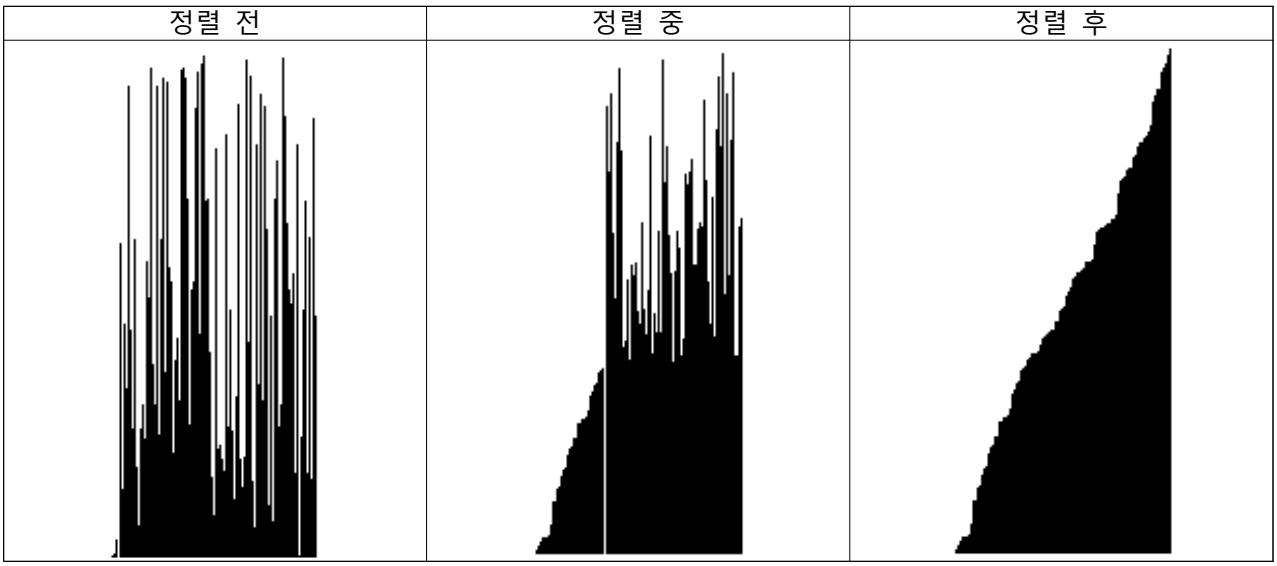
```
rrkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkpp
rrbbgggggggggggggggbbbbbgggbbbbbbbbgggbbbbbbbbbbbbbgggbbpp
kbbbgggggggggggggggbbbbbgggbbbbbbbbgggbbbbbbbbbbbbbgggbbbk
kbbbbbbbbbbbbbggggbbbbbbgggbbbbbbbbgggggggggggggggggggbbbk
kbbbbbbbbbbbggggbbbggggggggbbbbbbbbgggggggggggggggggggbbbk
kbbbbbbbbbggggbbbbbggggggggbbbbbbbbgggwwwwwwwwwwwwwgggbbbk
kbbbbbbbggggbggggbbbbbbbgggbbbbbbbbgggwwwwwwwwwwwwwgggbbbk
kbbbbbggggbbbbbggggbbbbbgggbbbbbbbbgggggggggggggggggggbbbk
kbbbggggbbbbbbbbbggggbbbgggbbbbbbbbgggggggggggggggggggbbbk
kbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbk
kbbbbbbbbbbbbgggggggggbbbbbbbbbbbbbbbbbbbbbgggbbbbbbbbbbbk
kbbbbbbbbbggggwwwwwwwggggbbbbbbbbbbbbbbbbbbgggbbbbbbbbbbbk
kbbbbbbbggggwwwwwwwwwwwggggbbbbbbbbbbbbbbbbgggbbbbbbbbbbbk
kbbbbbbbbbggggwwwwwwwggggbbbbbbbbbbgggggggggggggggggggbbbk
ppbbbbbbbbbbbgggggggggbbbbbbbbbbbbbgggggggggggggggggggbbrr
ppkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkrr
```

[코드]

```
file=open('pixel.txt','r')              # open()함수를 사용해 'r'모드(읽기) 객체 생성
a=file.read()                           # read()함수를 사용해 file 객체에 있는 내용 저장

r=''                                    # 변수 r 초기화 (압축된 문자열 저장)
p=0                                     # 변수 p 초기화 (문자 개수 계산을 위한 위치 저장)

for i in range(len(a)):                 # 문자열 변수 a의 길이만큼 반복
    if a[i]=='₩n':                      # 만약, 개행(줄바꿈) 문자라면,
        r+='₩n'                         # 변수 r에 개행(줄바꿈) 문자 추가
        p+=1                            # 계산 위치의 1 증가(개행 문자를 추가한 만큼 증가)
    elif a[i]!=a[i+1]:                  # 현재의 문자와 다음 문자가 같지 않다면,
        r+=(a[i]+'*'+str(i-p+1)+' ')    # 변수 r에 현재의 문자와 '*' 그리고, 개수를 추가
        p=i+1                           # 다음 문자 계산을 위한 위치 저장

print(r)
```

[21. 선택 정렬-selection_sort.py, 교과서-114,115페이지] 선택 정렬 알고리즘을 구현해 봅시다.

[실행 결과 예시]

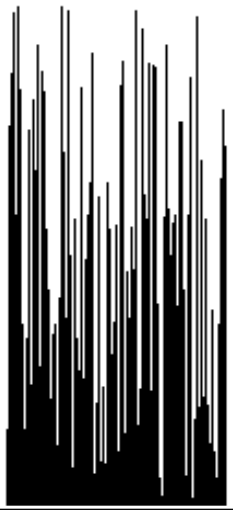


[코드]

```
import turtle as t; import random as rn     # turtle 모듈과 random 모듈을 t와 rn으로 설정
a=[0]*101                                   # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(101): a[i]=rn.randint(-50,200) # 정렬할 데이터 생성
t.ht(); t.penup(); t.speed(0)               # 펜 숨기기, 펜 들기, 펜 속도(가장빠름) 설정
for i in range(-50,51):                     # 정렬할 데이터를 막대그래프 형태로 시각화
    x=i; y=a[i+50]; t.goto(x,-50);
    t.pendown(); t.goto(x,y); t.penup()
def swap(x,y):                  # 정렬하는 과정에서 데이터 간 서로 위치를 변경할 때 사용하는 함수
    t.goto(x-50,-50); t.pendown(); t.color('white')
    t.goto(x-50,201); t.goto(y-50,201); t.goto(y-50,-50)
    t.color('black'); t.goto(y-50,a[y]); t.penup()
    t.goto(x-50,-50); t.pendown(); t.goto(x-50,a[x])
    t.penup()
for i in range(len(a)):         # 선택 정렬 구현
    min_v=9999999999
    for j in range(i,len(a)):
        if min_v>a[j]:
            min_v=a[j]
            min_i=j
    a[i],a[min_i]=a[min_i],a[i]
    swap(i,min_i)               # 위치가 서로 변경된 데이터의 막대그래프(시각화) 수정
```

[22. 버블 정렬-bubble_sort.py, 교과서-114,115페이지] 버블 정렬 알고리즘을 구현해봅시다.

[실행 결과 예시]

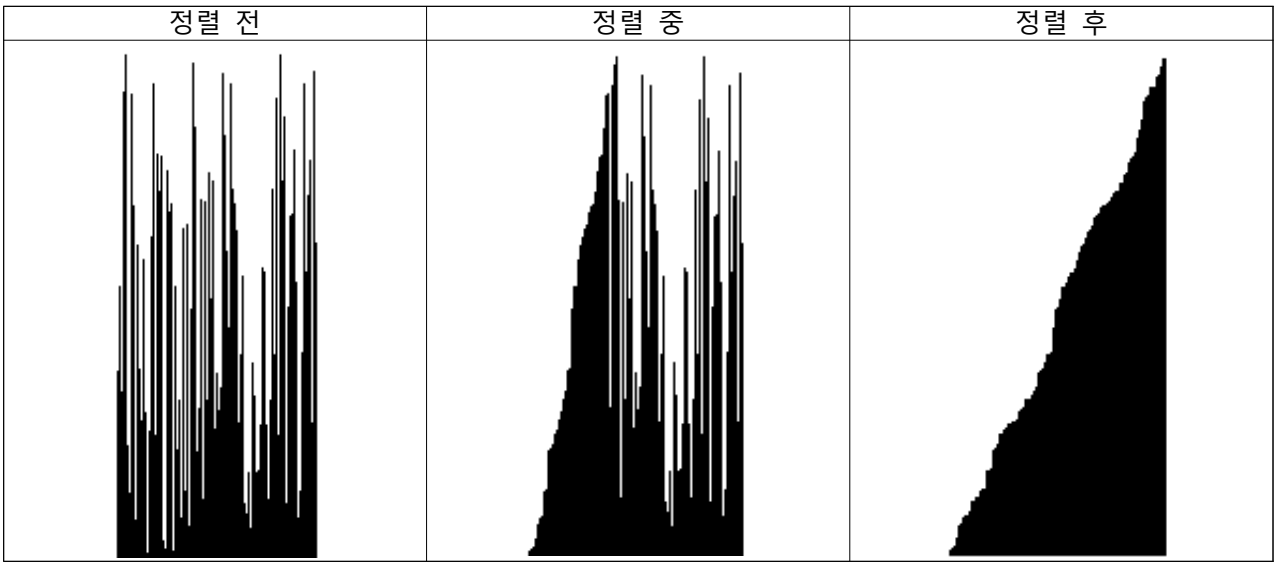
정렬 전	정렬 중	정렬 후
		

[코드]

```
import turtle as t; import random as rn     # turtle 모듈과 random 모듈을 t와 rn으로 설정

a=[0]*101                                   # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(101): a[i]=rn.randint(-50,200) # 정렬할 데이터 생성

t.ht(); t.penup(); t.speed(0)               # 펜 숨기기, 펜 들기, 펜 속도(가장빠름) 설정
for i in range(-50,51):                     # 정렬할 데이터를 막대그래프 형태로 시각화
    x=i; y=a[i+50]; t.goto(x,-50);
    t.pendown(); t.goto(x,y); t.penup()

def swap(x,y):                # 정렬하는 과정에서 데이터 간 서로 위치를 변경할 때 사용하는 함수
    t.goto(x-50,-50); t.pendown(); t.color('white')
    t.goto(x-50,201); t.goto(y-50,201); t.goto(y-50,-50)
    t.color('black'); t.goto(y-50,a[y]); t.penup()
    t.goto(x-50,-50); t.pendown(); t.goto(x-50,a[x])
    t.penup()

for i in range(len(a)-1,0,-1):   # 버블 정렬 구현
    for j in range(i):
        if a[j]>a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]
            swap(j,j+1)          # 위치가 서로 변경된 데이터의 막대그래프(시각화) 수정
```

[23. 삽입 정렬-insertion_sort.py, 교과서-114,115페이지] 삽입 정렬 알고리즘을 구현해 봅시다.

[실행 결과 예시]

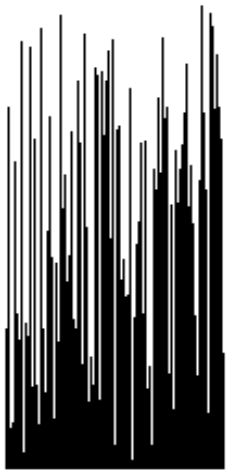


[코드]

```
import turtle as t; import random as rn     # turtle 모듈과 random 모듈을 t와 rn으로 설정

a=[0]*101                                   # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(101): a[i]=rn.randint(-50,200) # 정렬할 데이터 생성

t.ht(); t.penup(); t.speed(0)               # 펜 숨기기, 펜 들기, 펜 속도(가장빠름) 설정
for i in range(-50,51):                     # 정렬할 데이터를 막대그래프 형태로 시각화
    x=i; y=a[i+50]; t.goto(x,-50);
    t.pendown(); t.goto(x,y); t.penup()
def swap(x,y):                  # 정렬하는 과정에서 데이터 간 서로 위치를 변경할 때 사용하는 함수
    t.goto(x-50,-50); t.pendown(); t.color('white')
    t.goto(x-50,201); t.goto(y-50,201); t.goto(y-50,-50)
    t.color('black'); t.goto(y-50,a[y]); t.penup()
    t.goto(x-50,-50); t.pendown(); t.goto(x-50,a[x])
    t.penup()
for i in range(1,len(a)):       # 삽입 정렬 구현
    tmp=a[i]
    for j in range(i-1,-1,-1):
        if tmp<a[j]:
            a[j],a[j+1]=a[j+1],a[j]
            swap(j,j+1)         # 위치가 서로 변경된 데이터의 막대그래프(시각화) 수정
        else: break
```

[24. 병합 정렬-merge_sort.py, 교과서-114,115페이지] 병합 정렬 알고리즘을 구현해봅시다.

[실행 결과 예시]

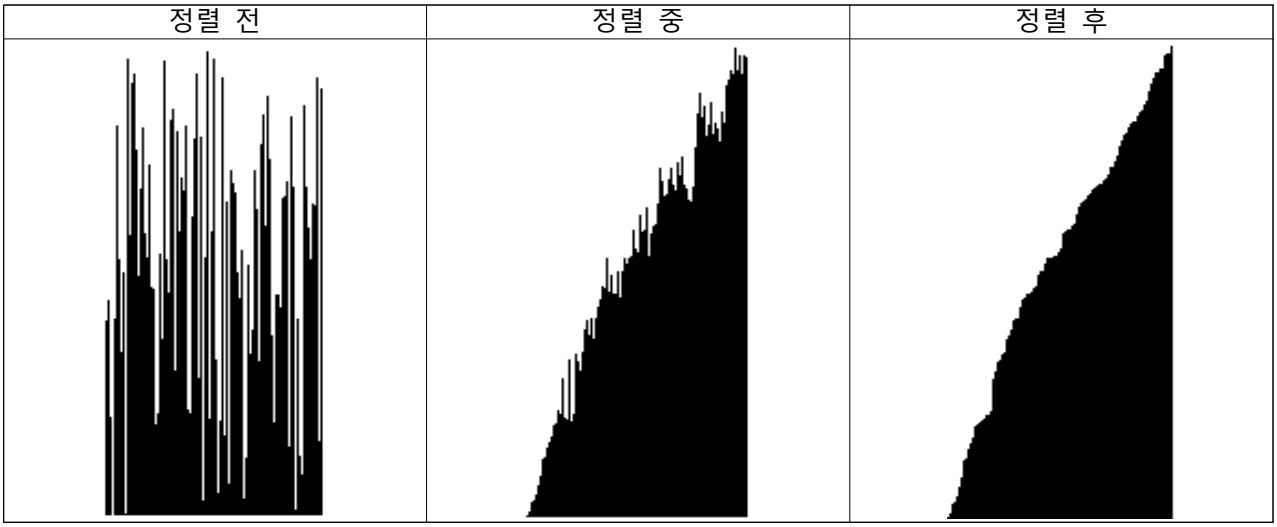
정렬 전	정렬 중	정렬 후
		

[코드]

```
import turtle as t; import random as rn     # turtle 모듈과 random 모듈을 t와 rn으로 설정
a=[0]*101                                   # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(101): a[i]=rn.randint(-50,200) # 정렬할 데이터 생성
t.ht(); t.penup(); t.speed(0)               # 펜 숨기기, 펜 들기, 펜 속도(가장빠름) 설정
for i in range(-50,51):                     # 정렬할 데이터를 막대그래프 형태로 시각화
    x=i; y=a[i+50]; t.goto(x,-50);
    t.pendown(); t.goto(x,y); t.penup()
tmp=[0]*len(a)                              # 병합하는 과정에서 필요한 임시 저장소 초기화
def merge(a,p,q,r):                         # 정렬을 하면서 병합을 진행하는 함수
    i=p; j=q+1; ti=0
    while i<=q and j<=r:
        if a[i]<=a[j]: tmp[ti]=a[i]; ti+=1; i+=1
        else: tmp[ti]=a[j]; ti+=1; j+=1
    while i<=q: tmp[ti]=a[i]; ti+=1; i+=1
    while j<=r: tmp[ti]=a[j]; ti+=1; j+=1
    i=p; ti=0
    while i<=r:
        a[i]=tmp[ti]                        # 병합이 진행된 것을 기존 리스트에 반영
        t.goto(i-50,-50)                    # 병합하는 과정을 시각화
        t.pendown(); t.color('white'); t.goto(i-50,201); t.penup(); t.goto(i-50,-50);
        t.pendown(); t.color('black'); t.goto(i-50,a[i]); t.penup(); i+=1; ti+=1
def m_sort(a,p,r):                          # 병합 정렬을 실행하는 함수(재귀함수)
    if p<r:
        q=(p+r)//2; m_sort(a,p,q); m_sort(a,q+1,r); merge(a,p,q,r)
m_sort(a,0,len(a)-1)
```

[25. 퀵 정렬-quick_sort.py, 교과서-114,115페이지] 퀵 정렬 알고리즘을 구현해봅시다.

[실행 결과 예시]

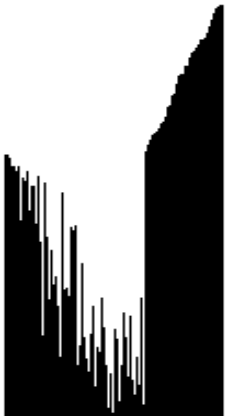


[코드]

```
import turtle as t; import random as rn     # turtle 모듈과 random 모듈을 t와 rn으로 설정
a=[0]*101                                   # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(101): a[i]=rn.randint(-50,200) # 정렬할 데이터 생성
t.ht(); t.penup(); t.speed(0)               # 펜 숨기기, 펜 들기, 펜 속도(가장빠름) 설정
for i in range(-50,51):                     # 정렬할 데이터를 막대그래프 형태로 시각화
    x=i; y=a[i+50]; t.goto(x,-50);
    t.pendown(); t.goto(x,y); t.penup()
def swap(x,y):                  # 정렬하는 과정에서 데이터 간 서로 위치를 변경할 때 사용하는 함수
    t.goto(x-50,-50); t.pendown(); t.color('white')
    t.goto(x-50,201); t.goto(y-50,201); t.goto(y-50,-50)
    t.color('black'); t.goto(y-50,a[y]); t.penup()
    t.goto(x-50,-50); t.pendown(); t.goto(x-50,a[x])
    t.penup()
def q_sort(a,p,r):              # 퀵 정렬을 실행하는 함수(재귀함수)
    if p<r:
        q=partition(a,p,r); q_sort(a,p,q-1); q_sort(a,q+1,r)
def partition(a,p,r):           # a[r]값을 기준으로 파티션을 나누는 함수
    x=a[r]; i=p-1
    for j in range(p,r):
        if a[j]<=x:
            i+=1; a[i],a[j]=a[j],a[i]
            swap(i,j)           # 위치가 서로 변경된 데이터의 막대그래프(시각화) 수정
    a[i+1],a[r]=a[r],a[i+1]
    swap(i+1,r)                 # 위치가 서로 변경된 데이터의 막대그래프(시각화) 수정
    return i+1
q_sort(a,0,len(a)-1)
```

[26. 힙 정렬-heap_sort.py, 교과서-114,115페이지] 힙 정렬 알고리즘을 구현해봅시다.

[실행 결과 예시]

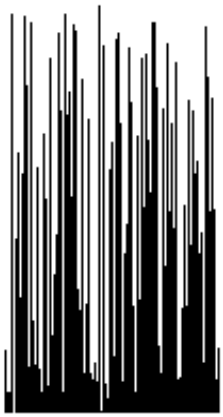
정렬 전	정렬 중	정렬 후
		

[코드]

```
import turtle as t; import random as rn     # turtle 모듈과 random 모듈을 t와 rn으로 설정
a=[0]*101                                   # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(101): a[i]=rn.randint(-50,200) # 정렬할 데이터 생성
t.ht(); t.penup(); t.speed(0)               # 펜 숨기기, 펜 들기, 펜 속도(가장빠름) 설정
for i in range(-50,51):                     # 정렬할 데이터를 막대그래프 형태로 시각화
    x=i; y=a[i+50]; t.goto(x,-50);
    t.pendown(); t.goto(x,y); t.penup()
def swap(x,y):                  # 정렬하는 과정에서 데이터 간 서로 위치를 변경할 때 사용하는 함수
    t.goto(x-50,-50); t.pendown(); t.color('white')
    t.goto(x-50,201); t.goto(y-50,201); t.goto(y-50,-50)
    t.color('black'); t.goto(y-50,a[y]); t.penup()
    t.goto(x-50,-50); t.pendown(); t.goto(x-50,a[x])
    t.penup()
def buildheap(a,n):             # 힙을 만드는 함수
    for i in range(int(n/2),-1,-1): heapify(a,i,n)
def heapify(a,k,n):             # 힙을 수선하는 함수
    left=2*k+1; right=2*k+2
    if right<=n:
        if int(a[left])<int(a[right]): bigger=right
        else: bigger=left
    elif left<=n: bigger=left
    else: return
    if int(a[bigger])>int(a[k]):
        a[k],a[bigger]=a[bigger],a[k]
        swap(k,bigger)          # 위치가 서로 변경된 데이터의 막대그래프(시각화) 수정
        heapify(a,bigger,n)
buildheap(a,len(a)-1)
for i in range(len(a)-1,0,-1):
    a[0],a[i]=a[i],a[0]         # 힙 트리의 루트에 있는 데이터 추출
    swap(0,i)                   # 위치가 서로 변경된 데이터의 막대그래프(시각화) 수정
    heapify(a,0,i-1)
```

[27. 기수 정렬-radix_sort.py, 교과서-114,115페이지] 기수 정렬 알고리즘을 구현해봅시다.

[실행 결과 예시]

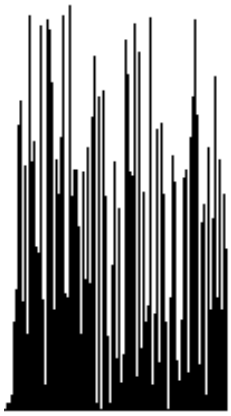
정렬 전	정렬 중	정렬 후
		

[코드]

```
import turtle as t; import random as rn     # turtle 모듈과 random 모듈을 t와 rn으로 설정
a=[0]*101                                   # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(101): a[i]=rn.randint(1,200) # 정렬할 데이터 생성
t.ht(); t.penup(); t.speed(0)               # 펜 숨기기, 펜 들기, 펜 속도(가장빠름) 설정
for i in range(-50,51):                     # 정렬할 데이터를 막대그래프 형태로 시각화
    x=i; y=a[i+50]; t.goto(x,0);
    t.pendown(); t.goto(x,y); t.penup()
def radix_sort(a):                          # 기수 정렬을 실행하는 함수
    b=10; d=1; s=len(a); max_v=max(a)
    while d <= max_v:
        tmp=[0]*s; cnt=[0]*b
        for i in range(s): cnt[(a[i]//d)%b]+=1  # 데이터의 각 자릿수를 기준으로 정렬
        for i in range(1,b): cnt[i]+=cnt[i-1]   # 계수 정렬의 기법을 활용
        for i in range(s-1,-1,-1):
            j = (a[i]//d)%b
            tmp[cnt[j]-1]=a[i]                  # 각 자릿수를 기준으로 정렬된 데이터를..
            cnt[j]-=1                           # 리스트 변수 tmp에 저장
        for i in range(s):
            a[i]=tmp[i]                         # tmp에 저장된 데이터를 반영
            t.goto(i-50,0)                      # 반영된 데이터를 시각화
            t.pendown(); t.color('white'); t.goto(i-50,201); t.penup(); t.goto(i-50,0)
            t.pendown(); t.color('black'); t.goto(i-50,a[i]); t.penup()
        d*=b                                    # 자릿수 증가를 위한 연산
radix_sort(a)
```

[28. 계수 정렬-counting_sort.py, 교과서-114,115페이지] 계수 정렬 알고리즘을 구현해 봅시다.

[실행 결과 예시]

정렬 전	정렬 중	정렬 후
		

[코드]

```
import turtle as t; import random as rn     # turtle 모듈과 random 모듈을 t와 rn으로 설정
a=[0]*101                                   # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(101): a[i]=rn.randint(1,200) # 정렬할 데이터 생성
t.ht(); t.penup(); t.speed(0)               # 펜 숨기기, 펜 들기, 펜 속도(가장빠름) 설정
for i in range(-50,51):                     # 정렬할 데이터를 막대그래프 형태로 시각화
    x=i; y=a[i+50]; t.goto(x,0);
    t.pendown(); t.goto(x,y); t.penup()
def counting_sort(a):                       # 계수 정렬을 실행하는 함수
    s=len(a); min_v=min(a); max_v=max(a)    # 데이터의 길이, 최소값, 최대값 설정
    cnt=[0]*(max_v+1); tmp=[0]*s            # 리스트 변수들의 초기화
    for i in range(s): cnt[a[i]]+=1         # 리스트의 인덱스를 활용하여 데이터 정렬
    for i in range(min_v,max_v+1): cnt[i]+=cnt[i-1]
    for i in range(s-1,-1,-1):
        tmp[cnt[a[i]]-1]=a[i]; cnt[a[i]]-=1 # 정렬된 데이터를 리스트 변수 tmp에 저장
    for i in range(s):
        a[i]=tmp[i]                         # tmp에 저장된 데이터를 반영
        t.goto(i-50,0)                      # 반영된 데이터를 시각화
        t.pendown(); t.color('white'); t.goto(i-50,201); t.penup(); t.goto(i-50,0)
        t.pendown(); t.color('black'); t.goto(i-50,a[i]); t.penup()
counting_sort(a)
```

[29. 각종 정렬 알고리즘의 수행시간 비교하기-sort_compare.py, 교과서-114,115페이지]
각종 정렬 알고리즘의 수행시간을 비교하는 프로그램을 작성해봅시다.

[실행 결과 예시]

1. selection_sort: 7.86000
2. bubble_sort : 26.06531
3. insertion_sort: 14.39889
4. merge_sort : 0.07809
5. quick_sort : 0.06240
6. heap_sort : 0.21867
7. radix_sort : 0.03124
8. counting_sort : 0.01562

[코드]

```
import random as rn; import time as tm   # random 모듈과 time 모듈을 rn과 tm으로 설정

arr=[0]*20001                            # 정렬할 데이터를 저장할 리스트 변수 초기화
for i in range(20001): arr[i]=rn.randint(1,30000) # 정렬할 데이터 생성
arr=tuple(arr)                           # 리스트를 튜플로 형변환(데이터의 수정을 막기 위함)

def selection_sort(a):                   # 선택 정렬
    ts=tm.time()                         # 시간 확인 시작
    for i in range(len(a)):
        min_v=9999999999
        for j in range(i,len(a)):
            if min_v>a[j]: min_v=a[j]; min_i=j
        a[i],a[min_i]=a[min_i],a[i]
    te=tm.time()                         # 시간 확인 종료
    print('%s: %.5f'%('1. selection_sort',te-ts))

def bubble_sort(a):                      # 버블 정렬
    ts=tm.time()
    for i in range(len(a)-1,0,-1):
        for j in range(i):
            if a[j]>a[j+1]:
                a[j],a[j+1]=a[j+1],a[j]
    te=tm.time()
    print('%s: %.5f'%('2. bubble_sort   ',te-ts))

```

```
26 def insertion_sort(a):                    # 삽입 정렬
27     ts=tm.time()
28     for i in range(1,len(a)):
29         tmp=a[i]
30         for j in range(i-1,-1,-1):
31             if tmp<a[j]:
32                 a[j],a[j+1]=a[j+1],a[j]
33             else: break
34     te=tm.time()
35     print('%s: %.5f'%('3. insertion_sort',te-ts))
36
37 def merge_sort(a):                          # 병합 정렬
38     ts=tm.time()
39     tmp=[0]*len(a)
40     def merge(a,p,q,r):
41         i=p;j=q+1;ti=0
42         while i<=q and j<=r:
43             if a[i]<=a[j]:
44                 tmp[ti]=a[i]
45                 ti+=1;i+=1
46             else:
47                 tmp[ti]=a[j]
48                 ti+=1;j+=1
49         while i<=q:
50             tmp[ti]=a[i]
51             ti+=1;i+=1
52         while j<=r:
53             tmp[ti]=a[j]
54             ti+=1;j+=1
55         i=p;ti=0
56         while i<=r:
57             a[i]=tmp[ti]
58             i+=1;ti+=1
59
60     def m_sort(a,p,r):
61         if p<r:
62             q=(p+r)//2
63             m_sort(a,p,q)
64             m_sort(a,q+1,r)
65             merge(a,p,q,r)
66     m_sort(a,0,len(a)-1)
67     te=tm.time()
68     print('%s: %.5f'%('4. merge_sort    ',te-ts))
```

```
69  def quick_sort(a):                          # 퀵 정렬
70      ts=tm.time()
71      def q_sort(a,p,r):
72          if p<r:
73              q=partition(a,p,r)
74              q_sort(a,p,q-1)
75              q_sort(a,q+1,r)
76      def partition(a,p,r):
77          x=int(a[r])
78          i=p-1
79          for j in range(p,r):
80              if int(a[j])<=x:
81                  i+=1
82                  a[i],a[j]=a[j],a[i]
83          a[i+1],a[r]=a[r],a[i+1]
84          return i+1
85      q_sort(a,0,len(a)-1)
86      te=tm.time()
87      print('%s: %.5f'%('5. quick_sort    ',te-ts))
88
89  def heap_sort(a):                           # 힙 정렬
90      ts=tm.time()
91      def buildheap(a,n):
92          for i in range(int(n/2),-1,-1): heapify(a,i,n)
93      def heapify(a,k,n):
94          left=2*k+1
95          right=2*k+2
96          if right<=n:
97              if int(a[left])<int(a[right]): bigger=right
98              else: bigger=left
99          elif left<=n:
100             bigger=left
101         else: return
102         if int(a[bigger])>int(a[k]):
103             a[k],a[bigger]=a[bigger],a[k]
104             heapify(a,bigger,n)
105     buildheap(a,len(a)-1)
106     for i in range(len(a)-1,0,-1):
107         a[0],a[i]=a[i],a[0]
108         heapify(a,0,i-1)
109     te=tm.time()
110     print('%s: %.5f'%('6. heap_sort     ',te-ts))
111
```

```
112 def radix_sort(a):                          # 기수 정렬
113     ts=tm.time()
114     b=10; d = 1
115     s = len(a)
116     max_v = max(a)
117     while d <= max_v:
118         tmp = [0]*s
119         cnt = [0]*b
120         for i in range(s): cnt[(a[i]//d)%b]+=1
121         for i in range(1,b): cnt[i]+=cnt[i-1]
122         for i in range(s-1,-1,-1):
123             j = (a[i]//d)%b
124             tmp[cnt[j]-1]=a[i]
125             cnt[j]-=1
126         for i in range(s): a[i]=tmp[i]
127         d*=b
128     te=tm.time()
129     print('%s: %.5f'%('7. radix_sort    ',te-ts))
130 
131 def counting_sort(a):                      # 계수 정렬
132     ts=tm.time()
133     s=len(a)
134     min_v=min(a)
135     max_v=max(a)
136     cnt=[0]*(max_v+1)
137     tmp=[0]*s
138     for i in range(s): cnt[a[i]]+=1
139     for i in range(min_v,max_v+1): cnt[i]+=cnt[i-1]
140     for i in range(s-1,-1,-1):
141         tmp[cnt[a[i]]-1]=a[i]
142         cnt[a[i]]-=1
143     for i in range(s): a[i]=tmp[i]
144     te=tm.time()
145     print('%s: %.5f'%('8. counting_sort ',te-ts))
146 
147 selection_sort(list(arr))
148 bubble_sort(list(arr))
149 insertion_sort(list(arr))
150 merge_sort(list(arr))
151 quick_sort(list(arr))
152 heap_sort(list(arr))
153 radix_sort(list(arr))
 15 counting_sort(list(arr))
```