과학 (Science)

09강

딕셔너리 자료형 Review

- 키(key)와 값(value)을 하나의 원소로 하는 순서가 없는 집합
- 키와 값은 정수나 문자열 등 임의의 자료형 사용 가능
- 키를 이용한 인덱싱으로만 값에 접근 가능

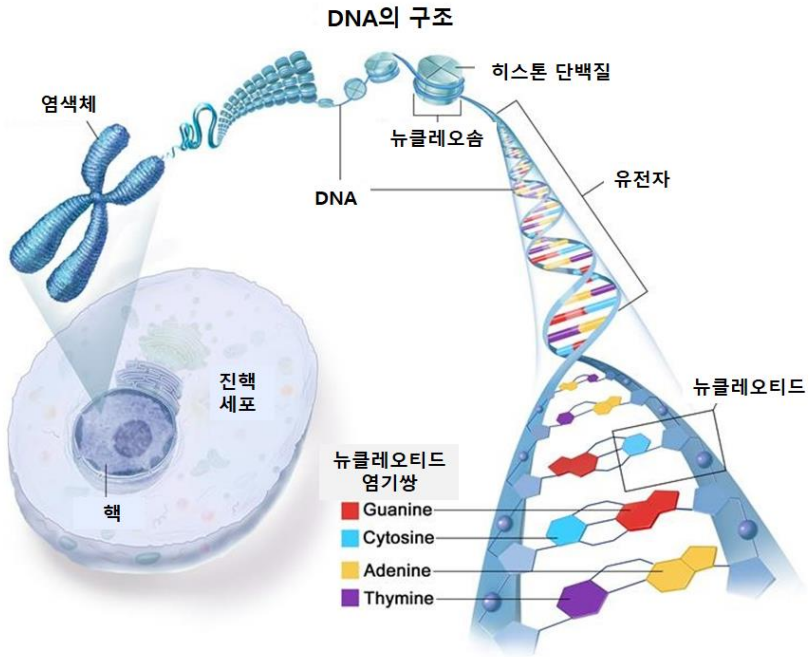
```
comp_dict = {'A':'T', 'T':'A', 'C':'G', 'G':'C'}
comp_dict
len(comp_dict)        # 딕셔너리의 크기(원소개수)
comp_dict.keys()      # 딕셔너리의 모든 키 출력
comp_dict.values()    # 딕셔너리의 모든 값 출력
print(comp_dict['T'])
```

```
digit = {1:'일', 2:'이', 3:'삼'}
digit
len(digit)
digit.keys()
digit.values()
print(digit[1])
print(digit[3])

fruit = {'사과':1000, '바나나':3000, '수박':8000}
fruit
len(fruit)
fruit.keys()
fruit.values()
print(fruit['사과'])
print(fruit['수박'])
```

유전자 ?

- “모든 생물의 유전물질인 DNA 내에서 유전정보가 있는 부위를 의미”
- 이 유전정보는 생체 내의 다양한 단백질 합성을 위한 설계도 역할을 함.
- 다양한 단백질이 생체 내에서 여러 필수적인 기능들을 수행하며 생명의 유지, 개인의 외모, 성격 등의 개인차를 발생시킴.
- 생물의 유전형질을 결정하는 단백질을 지정하는 기본적인 단위
- 유전병 (혈우병, 헌팅턴병, 낫모양 적혈구 빈혈증 등)



- 유전정보 전체 = 유전체 (지놈: genome)
- 유전체 : DNA라는 거대 유기분자로 구성
- 4종류의 염기 :

  A(아데닌), T(티민), G(구아닌), C(시토신)
- 염기들의 배치 순서에 따라 그 생명의 종과 생물학적 특성, 종간의 연관성이 결정

DNA 염기서열의 순서 변경

- DNA 염기서열의 분석 과정에서 이루어지는 작업
- 상보적(complementary) 방식 : 염기 A > T, T > A, G > C, C > G 로 변경
- 역순(reverse) 방식 : 염기서열의 순서를 역순으로 바꿈
- 상보적 역순(reverse-complementary) 방식 : 상보적 염기서열을 다시 역순으로 바꿈

방식	변경 전	변경 후
상보적(complementary)	AATTGGCC	TTAACCGG
역순(reverse)	AATTGGCC	CCGGTTAA
상보적 역순(reverse-complementary)	AATTGGCC	GGCCAATT

DNA 염기서열 순서 변경 프로그램

[문제] DNA 염기서열을 입력받아 상보적으로 변경하는 complement() 함수, 역순으로 변경하는 reverse() 함수, 상보적 역순으로 변경하는 reverse_complement() 함수를 작성하시오.

1. 문제 분석

- 임의 길이의 DNA 염기서열 문자열을 입력 받고 변환 방식을 입력 받음. 변환 방식에 맞는 함수 호출, 결과 출력

함수	• comp() : DNA 염기서열 문자열에서 A는 T로, T는 A로, G는 C로, C는 G로 변환 • rev() : DNA 염기서열 문자열을 역순으로 변환 • rev_comp() : DNA 염기서열 문자열을 상보적 문자열로 변환한 후 역순으로 변환
입력	• DNA 염기서열 문자열, input() 함수 사용 • 변환 방식 : int() 함수, input() 함수 사용 – input() 함수의 인수로 "1(comp), 2(Rev), 3(Rev_Comp)" 문자열 사용 – 입력 받은 숫자가 1~3일 경우에만 변환 작업을 진행하고, 1~3 이외의 숫자가 입력되면 "1(comp), 2(Rev), 3(Rev_Comp)!!"문자열을 출력
출력	• DNA 염기서열 문자열에 대한 상보적, 역순, 상보적 역순 문자열 출력
변수	• src : 입력 받은 DNA 염기서열 문자열 저장 • cnvt : 변환 방식(1:comp, 2:Rev, 3:Rev_Comp)

2. 알고리즘 설계 (1)

- 문제 분석에서의 함수, 입력, 출력에 대한 알고리즘을 자연어로 표현

<table>
<tr><td rowspan="2">comp()</td><td>인수</td><td>seq : 염기서열 문자열</td><td>반환</td><td>상보적 문자열</td></tr>
<tr><td colspan="4">1. A:T, T:A, C:G, G:C의 키:값 쌍으로 구성된 comp_dict 딕셔너리(dictionary) 선언
2. 문자열 변수 seq_comp 초기화
3. 변수 char의 값을 seq 문자열의 문자 순서대로 변경하며 반복
3.1 변수 seq_comp에 comp_dict[char] 값 대입
4. 변수 seq_comp의 값 반환</td></tr>
<tr><td rowspan="2">rev()</td><td>인수</td><td>seq : 염기서열 문자열</td><td>반환</td><td>역순 문자열</td></tr>
<tr><td colspan="4">1. 문자열 변환 함수인 reversed() 함수를 이용하여 seq 인수의 문자열 값을 역순으로 반환함, 이 결과를 join() 함수를 이용하여 공백의 빈 문자열과 결합하여 변수 seq_rev에 대입
2. 변수 seq_rev의 값 반환</td></tr>
</table>

2. 알고리즘 설계 (2)

<table>
<tr><td rowspan="2">rev_comp()</td><td>인수</td><td>seq : 염기서열 문자열</td><td>반환</td><td>상보적 역순 문자열</td></tr>
<tr><td colspan="4">1. comp() 함수를 이용하여 인수 seq의 상보적 문자열 값을 구하고, 임시 변수인 tmp에 대입
2. rev() 함수를 이용하여 변수 tmp의 값을 역순 문자열로 변환하고 결과 값 반환</td></tr>
<tr><td>입력 값 검사</td><td colspan="4">1. 만약 cnvt>=1이고 cnvt<=3이면
1.1 만약 cnvt == 1이면 comp() 함수 호출
그렇지 않으면 만약 cnvt == 2이면 rev() 함수 호출
그렇지 않으면 rev_comp() 함수 호출
1.2 각 함수의 결과 값 출력
그렇지 않으면 # (만약 cnvt<1이거나 cnvt>3이면)
올바른 입력 방법을 안내하는 문자열 출력</td></tr>
</table>

3. 코딩

- 알고리즘 설계를 이용하여 파이썬 프로그램을 코딩

```
def comp(seq):
    comp_dict = {'A':'T', 'T':'A', 'C':'G', 'G':'C'}
    seq_comp = ""
    for char in seq:
        seq_comp = seq_comp + comp_dict[char]
    return seq_comp

def rev(seq):
    seq_rev = "".join(reversed(seq))
    return seq_rev

def rev_comp(seq):
    tmp = comp(seq)
    return rev(tmp)
```

```
src = input("DNA sequence : ")
cnvt = int(input("1(comp), 2(rev), 3(rev_comp): "))
if (cnvt >= 1 and cnvt <= 3):
    if (cnvt == 1):
        rst = comp(src)
    elif (cnvt == 2):
        rst = rev(src)
    else:
        rst = rev_comp(src)
    print(src, '->', rst)
else:
    print("1(comp), 2(rev), 3(rev_comp)!!")
```

4. 테스트/디버깅

- 실행을 하여 데이터 입력에 대한 예상 결과가 나타나는지 결과 확인
- 동작 상태를 확인하고, 오류가 발생하는지 확인하여 디버깅

입력	결과	확인 및 수정 사항
실행	DNA sequence :	
AATTGGCC		변수 src에 대입
0	1(Comp), 2(Rev), 3(Rev_Comp)!!	입력 안내글 출력
4	1(Comp), 2(Rev), 3(Rev_Comp)!!	입력 안내글 출력
1	AATTGGCC -> TTAACCGG	
2	AATTGGCC -> CCGGTTAA	
3	AATTGGCC -> GGCCAATT	
EETTGGCC	오류 발생 seq_comp = seq_comp + comp_dict[char] KeyError: 'E	A,T,G,C에 해당하지 않는 문자로 인해 딕셔너리의 키 값이 맞지 않음을 알리는 오류 발생 => A,T,G,C로 구성된 문자열 입력

모듈 사용 (import)

- 모듈에 포함된 함수를 사용하려면 모듈을 포함(import)해야 함

```
import 모듈명
```

- 이름이 긴 모듈 등을 사용할 때 모듈의 별명을 만들어 사용 가능

```
import 모듈명 as 별명


>>> import turtle as t
>>> t.setup(500, 600)
```

- 모듈 내의 함수를 호출할 경우 '모듈명.함수()'와 같이 호출함
- 모듈 내의 원하는 함수만 포함할 경우 'from 모듈명 import 함수' 처럼 사용하면 모듈명이 포함되지 않은 채로 함수를 단순하게 표기 가능

```
>>> import math
>>> math.sin(1)
0.8414709848078965
>>> from math import sin
>>> sin(1)
0.8414709848078965
```

터틀 스크린의 마우스 이벤트 처리

- 콜백함수(callback function)
  - 이벤트(event)가 발생하였을 때 이벤트를 처리하기 위해 호출되는 함수
  - 마우스가 클릭되는 이벤트가 발생하였을 때 onscreenclick() 함수의 인수로 콜백함수 등록

- 콜백함수를 사용하지 않고 함수 등을 직접 호출하여 실행할 수 있음.
  - 마우스 클릭이 발생하면 터틀의 goto(x, y) 함수가 직접 실행됨

```
import turtle as t
s = t.Screen()
s.onscreenclick(t.goto)        # 마우스 클릭이 발생하면 처리할 콜백함수 등록
s.listen()                     # 사용자 입력을 위한 포커스 처리 및 이벤트 발생 확인
```

터틀 스크린의 마우스 이벤트 처리

- onscreenclick() 함수의 인수를 통해 마우스를 클릭한 버튼에 따라 다른 콜백함수 호출
- onscreenclick() 함수의 두 번째 인수
  - 보통 생략되어 사용됨
  - 1 : 마우스 왼쪽 버튼, 2 : 가운데 버튼, 3 : 오른쪽 버튼

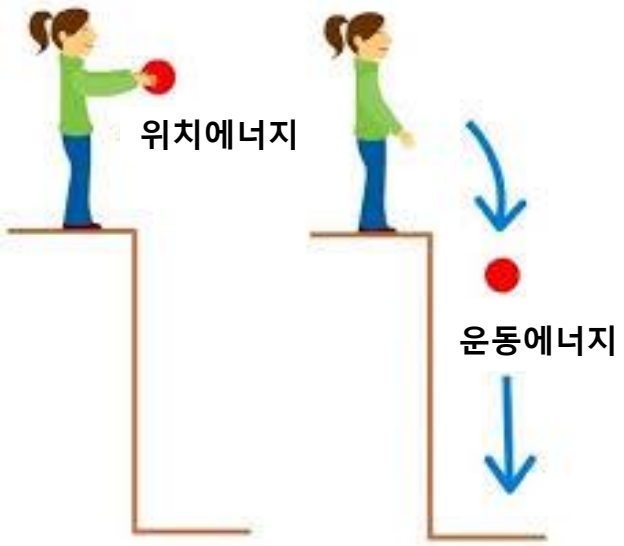
```
import turtle as t

def write_xy(x, y):
    t.goto(x, y)
    t.stamp()
    t.write("x:%d, y:%d"%(x, y))

def screen_clear(x, y):
    t.goto(x, y)
    t.clear()
```

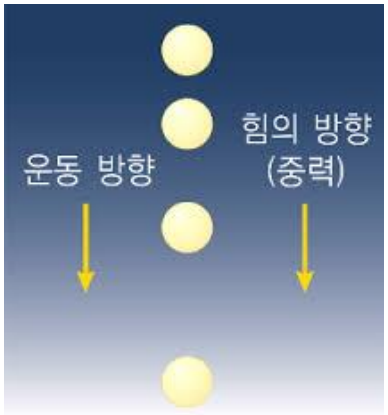
```
t.setup(600, 600)
s = t.Screen()
t.penup()
s.onscreenclick(write_xy, 1)
s.onscreenclick(screen_clear, 3)
s.listen()
```

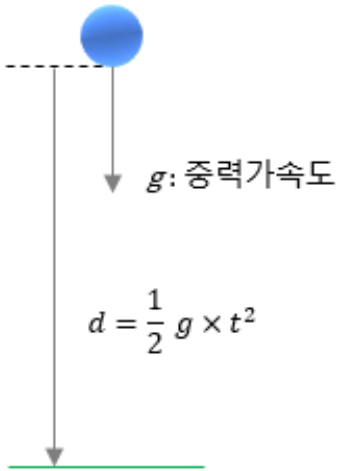
중력

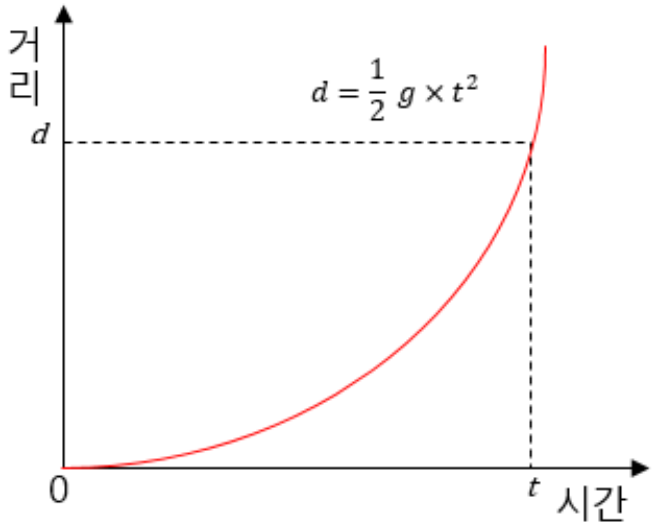


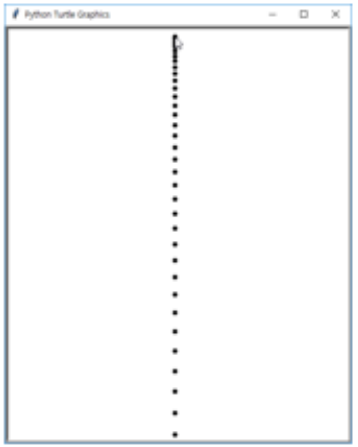
자유 낙하 운동

- 위치에너지
  - 물체가 정지되어 있는 상태에서 중력 작용
  - 물체가 가지게 되는 운동 가능한 잠재적인 힘
- 운동에너지
  - 물체가 떨어지기 시작하며 갖는 에너지
- 자유 낙하 운동
  - 위치에너지를 갖는 물체가 떨어지면서 운동에너지로 변환되며 갖는 운동
  - 물체는 바람이나 다른 힘의 영향을 고려하지 않을 경우 중력으로 인해 1초에 약 9.8m/초씩 속도가 증가









물체의 자유 낙하 시 위치 계산

[문제] 터틀 스크린의 한 지점을 클릭한 위치로부터 스크린의 하단 부분으로 점 형태의 터틀을 자유 낙하시켜 궤적을 표시하시오.

1. 문제 분석

- 터틀 스크린의 한 지점을 클릭
- 그 위치부터 스크린의 하단 부분까지 자유 낙하한다고 가정
- 낙하되는 과정은 중력가속도를 적용하면서 위치 이동하게 됨
- 변화된 위치에 공 모양의 터틀 흔적을 표시함

함수	• draw_pos() : 물체가 떨어지는 좌표마다 터틀의 흔적을 표시
입력	• 마우스 클릭 : 터틀 스크린의 임의의 위치를 클릭, 클릭한 위치부터 스크린의 하단 부분까지 낙하
출력	• 자유 낙하하는 이동 궤적의 좌표 마다 터틀의 흔적을 표시
변수	• t : 터틀 • s : 터틀 스크린

2. 알고리즘 설계 (1)

- 문제 분석에서의 함수, 입력, 출력에 대한 알고리즘을 자연어로 표현

draw_pos()	**인수**	x : x 좌표 y : y 좌표	**반환**	없음

1. 이전에 표시한 터틀 흔적을 모두 지움
2. 터틀의 위치를 x, y로 변경
3. 터틀의 흔적을 남김
4. 스크린의 하단에 해당하는 y축 위치를 계산하여 hl에 대입
5. 시간 변수 tm을 0으로 초기화
6. while 문을 무한 반복
   - 6.1 이동거리를 계산하여 d에 대입
   - 6.2 클릭한 y좌표에서 이동거리(d)를 뺀 후 결과를 ny에 대입
   - 6.3 만약 ny > hl이면
     - 6.3.1 터틀의 위치를 x, ny로 이동
     - 6.3.2 터틀의 흔적을 남김
     - 6.3.3 변수 tm의 값을 1 증가
   - 그렇지 않으면
     - 6.3.1 무한 반복 중단

2. 알고리즘 설계 (2)

터틀 생성 **스크린 생성**	1. 터틀 스크린 크기를 500, 600으로 설정 2. 터틀 모양을 circle로 설정 3. 터틀 크기를 0.3, 0.3으로 설정, 테두리는 0으로 설정하여 표시 안함 4. 터틀 펜을 올림 5. 터틀 스크린 생성
마우스 클릭	1. 터틀 스크린에서 마우스 클릭이 이루어지면 draw_pos() 콜백 함수 호출 2. 터틀 스크린에서의 이벤트 확인

3. 코딩

- 알고리즘 설계를 이용하여 파이썬 프로그램을 코딩

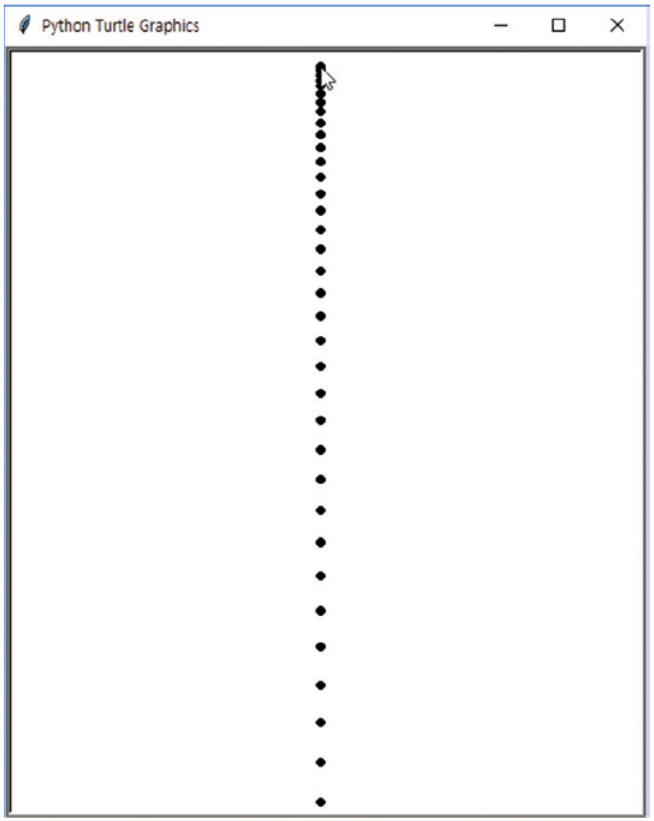
```
import turtle as t

def draw_pos(x, y):
    t.clear()
    t.setpos(x,y)
    t.stamp()
    hl = -(t.window_height()/2)
    tm = 0
    while True:
        d = (9.8*tm**2)/2
        ny = y - int(d)
        if ny > hl:
            t.goto(x, ny)
            t.stamp()
            tm = tm + 0.3
        else:
            break
```

```
t.setup(500, 600)
t.shape("circle")
t.shapesize(0.3, 0.3, 0)
t.penup()
s = t.Screen()
s.onscreenclick(draw_pos)
s.listen()
```

4. 테스트/디버깅

- 실행을 하여 데이터 입력에 대한 예상 결과가 나타나는지 결과 확인
- 동작 상태를 확인하고, 오류가 발생하는지 확인하여 디버깅

입력	결과	확인 및 수정 사항
마우스 클릭		마우스 클릭 위치부터 스크린 하단까지 궤적 점이 표시됨 스크린 하단 부분에 도달한 경우 궤적 표시가 중단됨

수학 모듈 (내장함수)

- 숫자 계산과 처리를 위해 abs(), max(), min(), sum(), pow() 함수 등의 내장 함수 제공

함수	기능	반환값
abs(x)	x의 절댓값을 구함	int, float 등
max(arg1, arg2, ...)	두 개 이상의 인수일 경우 그 중 최댓값을 구함	int, float 등
min(arg1, arg2, ...)	두 개 이상의 인수일 경우 그 중 최솟값을 구함	int, float 등
sum(iterable)	iterable에 지정되는 값의 총 합을 구함	int, float 등
pow(x, y)	x의 y승(즉, 거듭제곱)을 구함	int, float 등

- 내장 함수 공식 문서 : https://docs.python.org/3/library/functions.html

```
a = [-3, 7, 9, 4, 2, 3]
print(a[0], abs(a[0]), max(a), min(a), sum(a), pow(2, 3))
```

수학 모듈 (math)

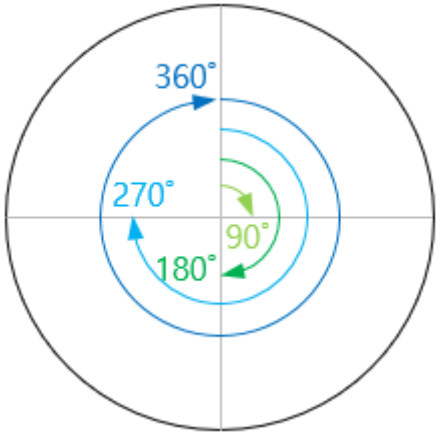
- 삼각함수나 지수, 로그 등의 함수와 상수 등은 math 모듈을 통해 제공

함수/상수	기능	반환값
log(x)	x의 로그를 구함	float
sqrt(x)	x의 제곱근을 구함	float
radians(x)	각도 x를 라디안 값으로 변환	float
sin(x)	라디안 x의 사인(sin)을 구함	float
cos(x)	라디안 x의 코사인(cosine)을 구함	float
pi, e	pi: 3.141592..., e: 2.718281...	

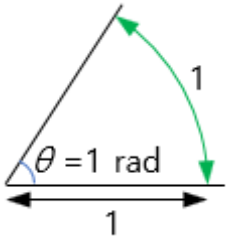
- math 모듈 공식 문서 : https://docs.python.org/3/library/math.html

각도의 표시방법 (degree vs. radian)

- 도(degree)
  - 일상적으로 사용하는 각도의 단위, 원 한 바퀴를 360도로 표현하는 방법
  - 반원은 180도, 직각은 90도와 같이 표현
  - 도 → 라디안 변환
    - 도 * 3.14 / 180
    - radians() 함수



- 라디안(radian)
  - 각도를 표현하는 또 다른 방법
  - 호의 길이가 반지름과 같게 되는 만큼의 각도를 1라디안이라고 정의
  - 1라디안은 약 57.3도 해당 각도
  - 라디안 → 도 변환
    - 라디안 * 180 / 3.14
    - degrees() 함수



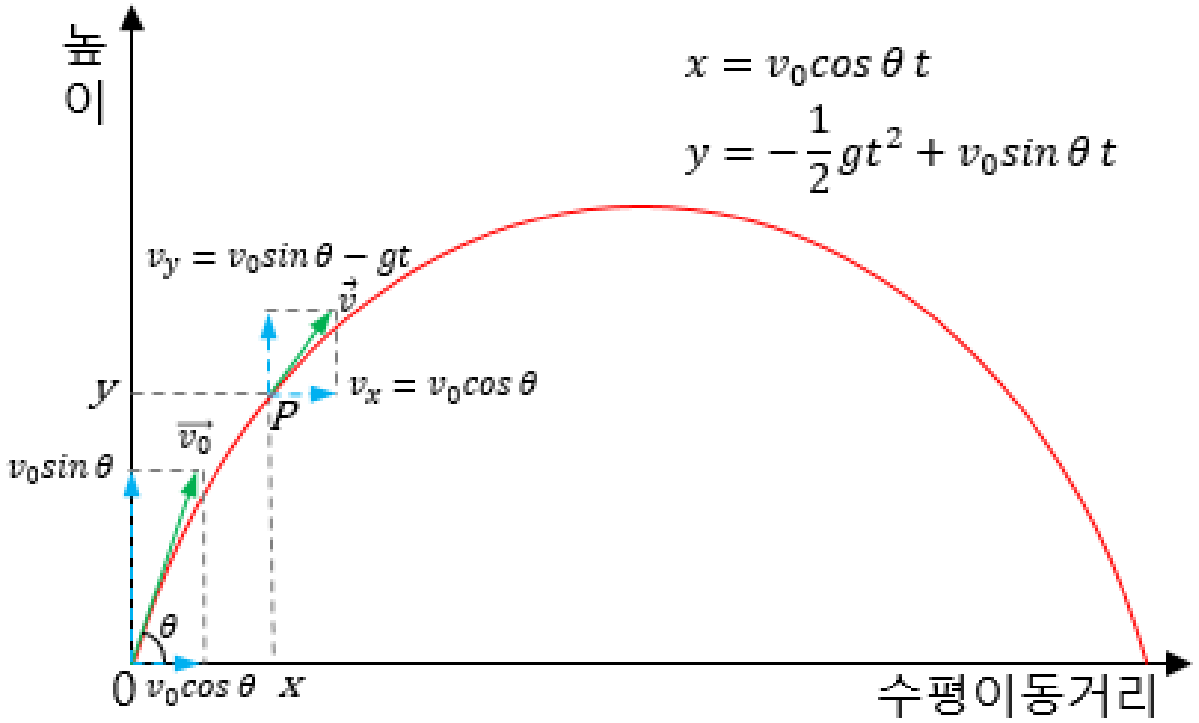

```
import math
a = 45
r = a *3.14/180.0          # 도 -> 라디안
print(r)
r1 = a * math.pi/180.0   #  도 -> 라디안
print(r1)
r2 = math.radians(a)      # 도 -> 라디안
print(r2)
print(math.sin(r2), math.cos(r2))

r = 1
a = r * 180.0/3.14         # 라디안 -> 도
print(a)
a1 = r*180.0/math.pi    # 라디안 -> 도
print(a1)
a2 = math.degrees(r)    # 라디안 -> 도
print(a2)

print(math.radians(90))
print(math.degrees(2))
```

포물선 운동

- 물체를 비스듬하게 위로 던질 경우 일정한 높이까지 물체가 올라간 후 중력의 영향으로 점차 낙하함 = 포물선 운동



물체 포물선 운동의 위치 계산

[문제] 터틀 스크린의 한 지점을 클릭한 위치부터 지정한 각도로 물체를 스크린의 하단 부분으로 떨어지는 포물선 운동의 궤적을 점 형태의 터틀을 이용하여 표시하시오.

1. 문제 분석

- 터틀 스크린의 한 지점을 클릭하여 그 위치부터 지정한 각도로 점 형태의 터틀을 던짐
- 점 형태의 터틀은 포물선 운동 형태의 위치 이동을 함
- 변화된 각 지점에 터틀의 흔적을 표시
- 바닥에 도달할 때까지 이를 반복함

함수	• draw_pos() : 포물선 운동을 하는 이동 궤적의 좌표마다 터틀의 흔적을 표시
입력	• 마우스 클릭 : 터틀 스크린의 임의의 위치를 클릭, 클릭한 위치부터 포물선 운동을 하며 스크린의 하단 부분까지 낙하
출력	• 포물선 운동을 하는 이동 궤적의 좌표마다 터틀의 흔적을 표시
변수	• tm : 시간 간격 • dx, dy : x 이동거리, y 이동거리 • velo : 속도 • t : 터틀 • ux, uy : x 속도, y 속도 • g : 중력가속도 (g: 9.8) • ang : 각도 • s : 터틀 스크린

2. 알고리즘 설계 (1)

- 문제 분석에서의 함수, 입력, 출력에 대한 알고리즘을 자연어로 표현

draw_pos()	**인수**	x : x 좌표 y : y 좌표	**반환**	없음

1. 속도를 입력 받아 velo에 대입(기본값:50, 최솟값:10, 최댓값:100)
2. 각도를 입력 받아 ang에 대입(기본값:45, 최솟값:0, 최댓값:360)
3. 이전에 표시한 터틀 흔적을 모두 지움
4. 터틀을 숨김
5. 터틀의 위치를 x, y로 변경
6. 터틀을 나타냄
7. 터틀의 흔적을 남김
8. 스크린의 하단에 해당하는 y축 위치를 계산하여 hl에 대입
9. x 속도와 y 속도를 계산하여 각각 ux, uy에 대입
10. while 문을 무한 반복
    - 10.1 중력가속도가 반영된 y 속도를 계산하여 uy에 대입
    - 10.2 y 이동거리를 계산하여 dy에 대입
    - 10.3 x 이동거리를 계산하여 dx에 대입
    - 10.4 만약 dy > hl이면
        - 10.4.1 터틀의 위치를 dx, dy로 이동
        - 10.4.2 터틀의 흔적을 남김
    - 그렇지 않으면
        - 10.4.1 무한 반복 중단

2. 알고리즘 설계 (2)

터틀 생성 **스크린 생성**	1. 터틀 스크린 크기를 600, 600으로 설정 2. 터틀 모양을 circle로 설정 3. 터틀 크기를 0.3, 0.3으로 설정, 테두리는 0으로 설정하여 표시 안함 4. 터틀 펜을 올림 5. 터틀 스크린 생성
마우스 클릭	1. 터틀 스크린에서 마우스 클릭이 이루어지면 draw_pos() 콜백 함수 호출 2. 터틀 스크린에서의 이벤트 확인

3. 코딩 (1)

- 알고리즘 설계를 이용하여 파이썬 프로그램을 코딩

```
import turtle as t
import math

tm = 0.3
ux = 0
uy = 0
dx = 0
dy = 0
g = 9.8
velo = 0
ang = 0
```

```
def draw_pos(x, y):
    velo = t.numinput("입력", "속도 : ", 50, 10, 100)
    ang = math.radians(t.numinput("입력", "각도 : ", 45, 0, 360))

    t.clearstamps()
    t.hideturtle()
    t.setpos(x, y)
    t.showturtle()
    t.stamp()

    hl = -(t.window_height()/2)

    ux = velo * math.cos(ang)
    uy = velo * math.sin(ang)
```

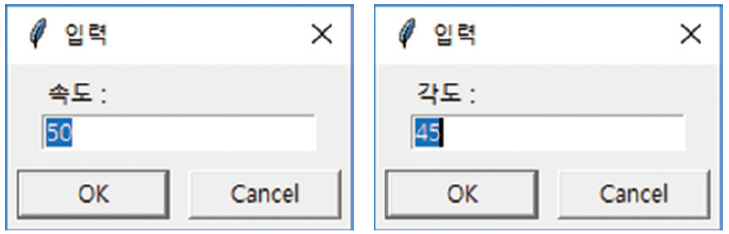
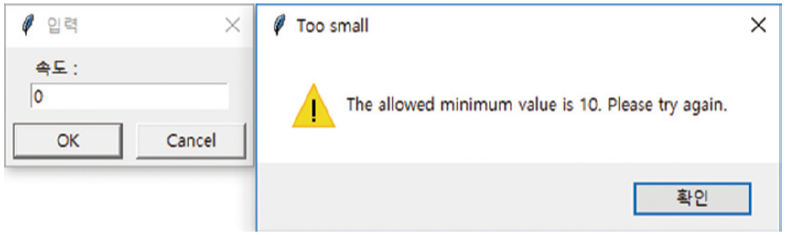
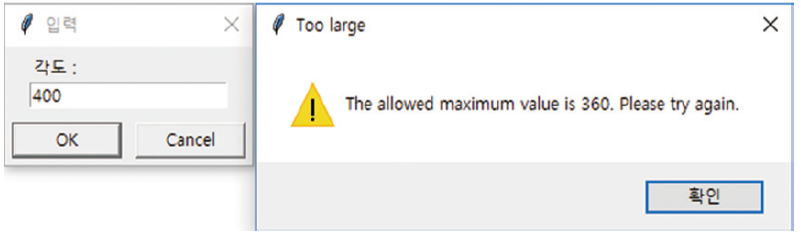
3. 코딩 (2)

```
while True:
    uy = uy +(-1*g)*tm
    dy = t.ycor() + (uy*tm) - (g*tm**2)/2
    dx = t.xcor() + (ux*tm)
    if dy > hl:
        t.goto(dx, dy)
        t.stamp()
    else:
        break
```

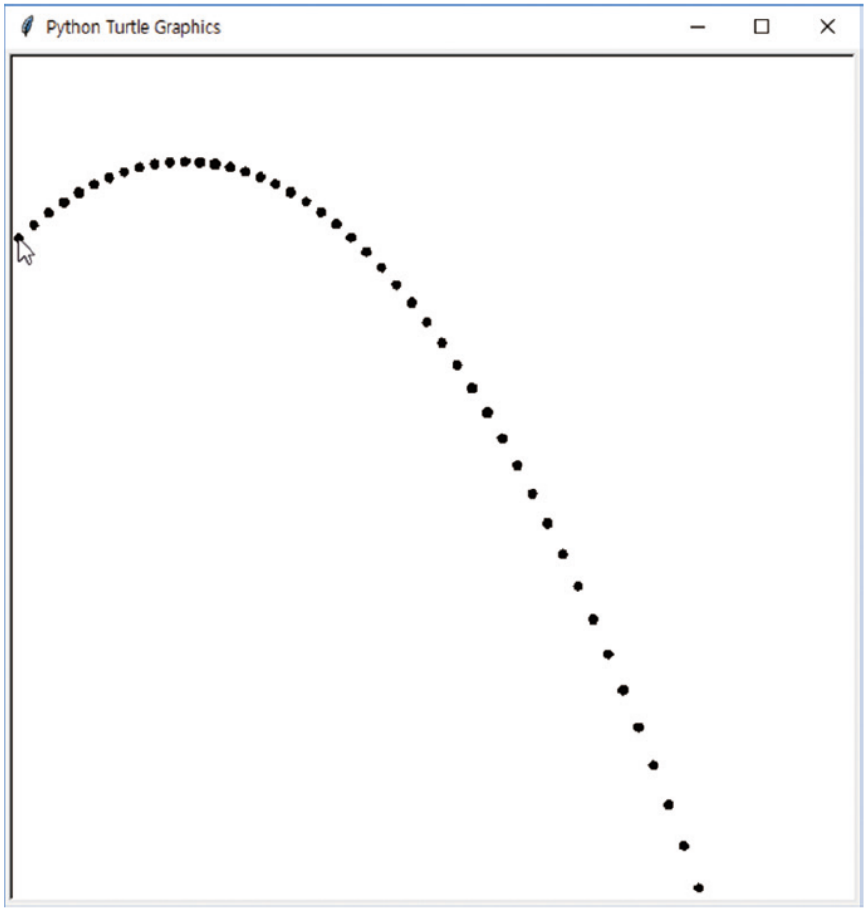
```
t.setup(600, 600)
t.shape("circle")
t.shapesize(0.3, 0.3, 0)
t.penup()
s = t.Screen()
s.onscreenclick(draw_pos)
s.listen()
```

4. 테스트/디버깅 (1)

- 실행을 하여 데이터 입력에 대한 예상 결과가 나타나는지 결과 확인
- 동작 상태를 확인하고, 오류가 발생하는지 확인하여 디버깅

입력	결과	확인 및 수정 사항
마우스 클릭		속도와 각도를 입력 받기 위해 입력 대화상자가 나타남
속도: 0		설정된 10~100 이외의 속도 값이 입력되면 경고 대화상자가 나타남
각도: 400		설정된 0~360 이외의 각도 값이 입력되면 경고 대화상자가 나타남

4. 테스트/디버깅 (2)

속도: 50 각도: 45		마우스 클릭 위치부터 스크린 하단까지 궤적 점이 표시됨 스크린 하단 부분에 도달한 경우 궤적 표시가 중단됨
마우스 클릭		스크린의 다른 위치에서 마우스 클릭 을 할 경우 기존의 궤적 점들이 모두 사라지고 해당 위치부터 스크린 하단까지 궤적 점이 다시 표시됨

EOD