

# 과학 (Science)

## 1 DNA 염기서열의 순서 바꾸기

프로그램 p08-01 ... DNA 염기서열의 순서 바꾸기

관련 학습 ... 딕셔너리(dictionary) 자료형

## 2 자유 낙하와 포물선 운동 궤적 그리기

### 2.1 자유 낙하 운동 궤적 그리기

프로그램 p08-02-1 ... 물체를 자유 낙하시키기

관련 학습 ... 모듈 임포트(import)

### 2.2 자유 낙하 운동 궤적 그리기

프로그램 p08-02-1 ... 물체의 포물선 운동

관련 학습 ... math 수학 모듈, 라디안(radian)

Thinking!

잠깐! Coding

Coding! Programming



## 학습 목표

- DNA 염기 서열 순서를 바꾸는 방법을 이해하고 상보적으로 바꾸는 함수, 역순으로 바꾸는 함수, 상보적 역순으로 바꾸는 함수를 작성할 수 있다.
- 자유 낙하 운동의 궤적을 그리는 방법을 이해하고 터틀 스크린의 한 지점부터 스크린의 하단 부분으로 자유 낙하시켜 궤적을 표시하는 프로그램을 작성할 수 있다.
- 포물선 운동의 궤적을 그리는 방법을 이해하고 터틀 스크린의 한 지점부터 지정한 각도로 스크린의 하단 부분으로 포물선 궤적을 표시하는 프로그램을 작성할 수 있다.



## 관련 학습

- 딕셔너리(dictionary) 자료형
- 모듈 임포트(import)
- 터틀의 마우스 이벤트 처리
- math 수학 모듈
- 라디안(radian)

# 과학 (Science)



## 1 DNA 염기서열의 순서 바꾸기

프로그램 p08-01 ... DNA 염기서열의 순서 바꾸기

관련 학습 ... 딕셔너리(dictionary) 자료형

## 2 자유 낙하와 포물선 운동 궤적 그리기

### 2.1 자유 낙하 운동 궤적 그리기

프로그램 p08-02-1 ... 물체를 자유 낙하시키기

관련 학습 ... 모듈 임포트(import)

### 2.2 자유 낙하 운동 궤적 그리기

프로그램 p08-02-1 ... 물체의 포물선 운동

관련 학습 ... math 수학 모듈, 라디안(radian)

Thinking!

잠깐! Coding

Coding! Programming

## ▶ 유전자

- ▶ 생물의 유전형질을 결정하는 단백질을 지정하는 기본적 단위
- ▶ 모든 생명체는 염기서열을 통해 단백질을 지정하는 원리 따름

## ▶ DNA 염기서열(sequencing)

- ▶ 4종류의 염기 : A(아데닌), T(티민), G(구아닌), C(시토신)
- ▶ 염기들의 배치 순서에 따라 그 생명의 종과 생물학적 특성, 종간의 연관성이 결정됨

## ▶ DNA 염기서열의 순서 변경 작업

### ▶ DNA 염기서열의 분석 과정에서 이루어지는 작업

- ▶ 상보적(complementary) 방식 : 염기 A > T, T > A, G > C, C > G로 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



프로그램

p08-01

## DNA 염기서열의 순서 바꾸기

DNA 염기서열을 입력받아 상보적으로 바꾸는 `complement()` 함수, 역순으로 바꾸는 `reverse()` 함수, 상보적 역순으로 바꾸는 `reverse_complement()` 함수를 작성해보자.

## 1 문제 분석

임의 자릿수의 DNA 염기서열 문자열을 입력받고 변환 방식을 결정한다. 그리고 해당 변환 방식에 맞는 함수를 호출하여 DNA 염기서열 문자열을 변환하여 결과를 출력한다.

함수	<ul style="list-style-type: none"> <li>• comp() : DNA 염기서열 문자열에서 A는 T로, T는 A로, G는 C로, C는 G로 변환</li> <li>• rev() : DNA 염기서열 문자열을 역순으로 변환</li> <li>• rev_comp() : DNA 염기서열 문자열을 상보적 문자열로 변환한 후 역순으로 변환</li> </ul>
입력	<ul style="list-style-type: none"> <li>• DNA 염기서열 문자열, input() 함수 사용</li> <li>• 변환 방식 : int() 함수, input() 함수 사용               <ul style="list-style-type: none"> <li>– input() 함수의 인수로 “1(comp), 2(Rev), 3(Rev_Comp)” 문자열 사용</li> <li>– 입력받은 숫자가 1~3일 경우에만 변환 작업을 진행하고, 1~3 이외의 숫자가 입력되면 “1(comp), 2(Rev), 3(Rev_Comp)!!” 문자열을 출력</li> </ul> </li> </ul>
출력	<ul style="list-style-type: none"> <li>• DNA 염기서열 문자열에 대한 상보적, 역순, 상보적 역순 문자열 출력</li> </ul>
변수	<ul style="list-style-type: none"> <li>• src : 입력받은 DNA 염기서열 문자열 저장</li> <li>• cnvt : 변환 방식(1:comp, 2:Rev, 3:Rev_Comp)</li> </ul>

## 2 알고리즘 설계

문제 분석에서의 함수와 입력, 출력에 대한 알고리즘을 자연어로 표현하면 다음과 같다.

	매개변수	seq : 염기서열 문자열	반환	상보적 문자열
<b>comp()</b>				<ol style="list-style-type: none"> <li>1. A:T, T:A, C:G, G:C의 키:값 쌍으로 구성된 comp_dict 딕셔너리(dictionary) 선언</li> <li>2. 문자열 변수 seq_comp 초기화</li> <li>3. 변수 char의 값을 seq 문자열의 문자 순서대로 변경하며 반복               <ol style="list-style-type: none"> <li>3.1 변수 seq_comp에 comp_dict[char] 값 대입</li> </ol> </li> <li>4. 변수 seq_comp의 값 반환</li> </ol>
	매개변수	seq : 염기서열 문자열	반환	역순 문자열
<b>rev()</b>				<ol style="list-style-type: none"> <li>1. 문자열 변환 함수인 reversed() 함수를 이용하여 매개변수 seq의 문자열 값을 역순으로 반환함, 이 결과를 join() 함수를 이용하여 공백의 빈 문자열과 결합하여 변수 seq_rev에 대입</li> <li>2. 변수 seq_rev의 값 반환</li> </ol>



	매개변수	seq : 염기서열 문자열	반환	상보적 역순 문자열
rev_comp()	<ol style="list-style-type: none"> <li>1. comp() 함수를 이용하여 매개변수 seq의 상보적 문자열 값을 구하고, 임시 변수인 tmp에 대입</li> <li>2. rev() 함수를 이용하여 변수 tmp의 값을 역순 문자열로 변환하고 결과 반환</li> </ol>			
입력 값 검사	<ol style="list-style-type: none"> <li>1. 만약 cnvt &gt;= 10이고 cnvt &lt;= 30이면               <ol style="list-style-type: none"> <li>1.1 만약 cnvt == 10이면 comp() 함수 호출 그렇지 않으면 만약 cnvt == 20이면 rev() 함수 호출 그렇지 않으면 rev_comp() 함수 호출</li> <li>1.2 각 함수의 결과 값 출력</li> </ol> </li> <li>그렇지 않으면 # (만약 cnvt &lt; 1 이거나 cnvt &gt; 30이면) 올바른 입력 방법을 안내하는 문자열 출력</li> </ol>			

### 3 코딩

알고리즘 설계를 이용하여 파이썬 프로그램을 코딩하면 다음과 같다.

```
1  def comp(seq):
2      comp_dict = {'A':'T', 'T':'A', 'C':'G', 'G':'C'}
3      seq_comp = ""
4
5
6      return seq_comp
7
8  def rev(seq):
9
10     return seq_rev
11
12  def rev_comp(seq):
13
14
```

```
15
16 src = input("DNA sequence : ")
17 cnvt = int(input("1(comp), 2(Rev), 3(Rev_Comp): "))
18
19
20
21
22
23
24
25
26
27
```

## 4 테스트/디버깅

입력	결과	확인 및 수정 사항
실행	DNA sequence :	
AATTGGCC		변수 src에 대입
0	1(Comp), 2(Rev), 3(Rev_Comp)!!	입력 안내 글 출력
4	1(Comp), 2(Rev), 3(Rev_Comp)!!	입력 안내 글 출력
1	AATTGGCC → TTAACCGG	
2	AATTGGCC → CCGGTAA	
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

**Thinking!**

1. 매개변수 `seq`에서 A, T, G, C 이외의 문자가 포함된 경우라도 오류를 발생하지 않고 해당 문자 대신에 '?' 문자를 출력하도록 `comp()` 함수를 변경해보자.
2. `rev()` 함수에서 `join()` 함수와 `reversed()` 함수를 사용하지 않고 `while` 문을 이용하여 문자열을 역순으로 변환해보자.



## 5 프로그램 코딩을 위한 관련 학습

### ▶ 딕셔너리 자료형

- ▶ 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
{'A': 'T', 'T': 'A', 'C': 'G', 'G': 'C'}
>>> len(comp_dict)                                # 딕셔너리의 크기(원소들의 개수)
4
>>> comp_dict.keys()                                # 딕셔너리의 모든 키 출력 [제독 없음]
dict_keys(['A', 'T', 'C', 'G'])
>>> comp_dict.values()                              # 딕셔너리의 모든 값 출력
dict_values(['T', 'A', 'G', 'C'])

>>> print(comp_dict['T'])
'A'
```

## &gt;&gt;&gt; 잠깐! Coding

1. { 1:'일', 2:'이', 3:'삼' }의 원소들을 갖는 딕셔너리 digit를 선언하고, 딕셔너리 digit의 크기, 모든 키와 값을 출력해보자. 그리고 키 1과 3에 해당하는 값을 출력해보자.
2. { '사과':100, '바나나':50, '수박':1000 }의 원소들을 갖는 딕셔너리 fruit을 선언하고, 딕셔너리 fruit의 크기, 모든 키와 값을 출력해보자. 그리고 키 '사과'와 '수박'에 해당하는 값을 출력해보자.



# 과학 (Science)

## 1 DNA 염기서열의 순서 바꾸기

프로그램 p08-01 ... DNA 염기서열의 순서 바꾸기

관련 학습 ... 딕셔너리(dictionary) 자료형

## 2 자유 낙하와 포물선 운동 궤적 그리기

### 2.1 자유 낙하 운동 궤적 그리기

프로그램 p08-02-1 ... 물체를 자유 낙하시키기

관련 학습 ... 모듈 임포트(import)

### 2.2 자유 낙하 운동 궤적 그리기

프로그램 p08-02-1 ... 물체의 포물선 운동

관련 학습 ... math 수학 모듈, 라디안(radian)

Thinking!

잠깐! Coding

Coding! Programming





## ▶ 위치에너지

- ▶ 물체가 정지되어 있는 상태에서 중력 작용
- ▶ 물체가 가제게 되는 운동 가능한 잠재적인 힘

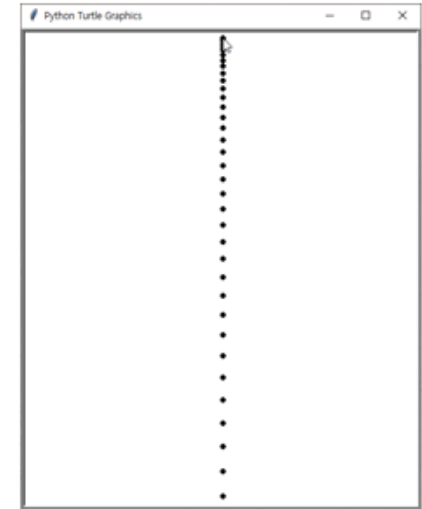
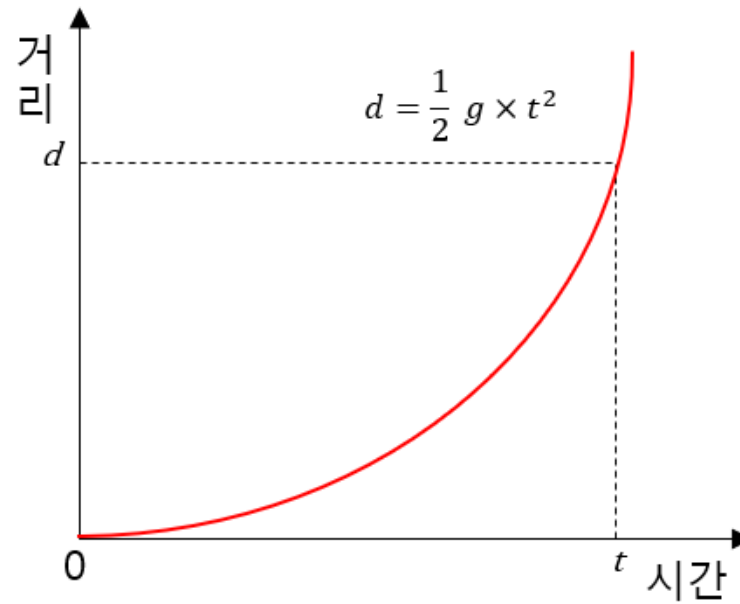
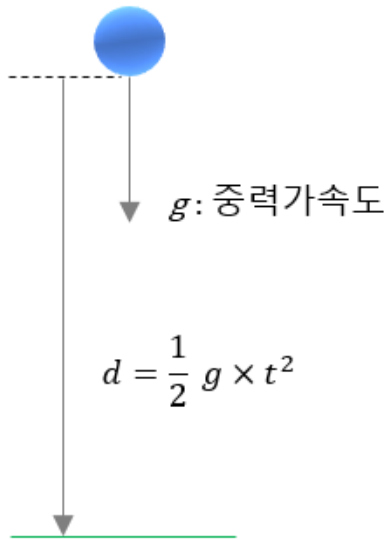
## ▶ 운동에너지

- ▶ 물체가 떨어지기 시작하며 갖는 에너지

## ▶ 자유 낙하 운동

- ▶ 물체가 떨어지는 운동에너지를 갖는 운동
- ▶ 물체는 바람이나 다른 힘의 영향을 고려하지 않을 경우 중력으로 인해 1초에 약  $9.8\text{m/초}$ 씩 속도가 증가

## ▶ 시간 변화에 따른 거리의 변동





## 프로그램 p08-02-1 물체를 자유 낙하시키기

터틀 스크린의 한 지점을 클릭한 위치로부터 스크린의 하단 부분으로 점 형태의 터틀을 자유 낙하시켜 궤적을 표시해보자.

## 1 문제 분석

터틀 스크린의 한 지점을 클릭한 위치부터 스크린의 하단 부분으로 점 형태의 터틀을 자유 낙하시킨다. 낙하 되는 과정은 중력가속도를 적용한 위치 이동을 하면서 변화된 각각의 지점에 터틀의 흔적을 표시한다.

함수	<ul style="list-style-type: none"> <li>draw_pos() : 떨어지는 물체의 이동 궤적 좌표마다 터틀의 흔적을 표시</li> </ul>
입력	<ul style="list-style-type: none"> <li>마우스 클릭 : 터틀 스크린의 임의 위치를 클릭, 클릭한 위치부터 스크린의 하단 부분까지 낙하</li> </ul>
출력	<ul style="list-style-type: none"> <li>떨어지는 물체의 이동 궤적 좌표마다 터틀의 흔적을 표시</li> </ul>
변수	<ul style="list-style-type: none"> <li>t : 터틀</li> <li>s : 터틀 스크린</li> </ul>

## 2 알고리즘 설계

매개변수

 $x$  :  $x$  좌표 $y$  :  $y$  좌표

반환

없음

draw\_pos()

1. 이전에 표시한 터틀 흔적을 모두 지움
2. 터틀의 위치를  $x$ ,  $y$ 로 변경
3. 터틀의 흔적을 남김
4. 스크린의 하단에 해당하는  $y$ 축 위치를 계산하여  $h$ 에 대입
5. 시간 변수  $tm$ 을 0으로 초기화
6. while 문을 무한 반복
  - 6.1 이동 거리를 계산하여  $d$ 에 대입
  - 6.2 클릭한  $y$  좌표에서 이동 거리( $d$ )를 뺀 후 결과를  $ny$ 에 대입
  - 6.3 만약  $ny > h$ 이면
    - 6.3.1 터틀의 위치를  $x$ ,  $ny$ 로 이동
    - 6.3.2 터틀의 흔적을 남김
    - 6.3.3 변수  $tm$ 의 값을 1 증가
  - 그렇지 않으면
    - 6.3.1 무한 반복 중단

### 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

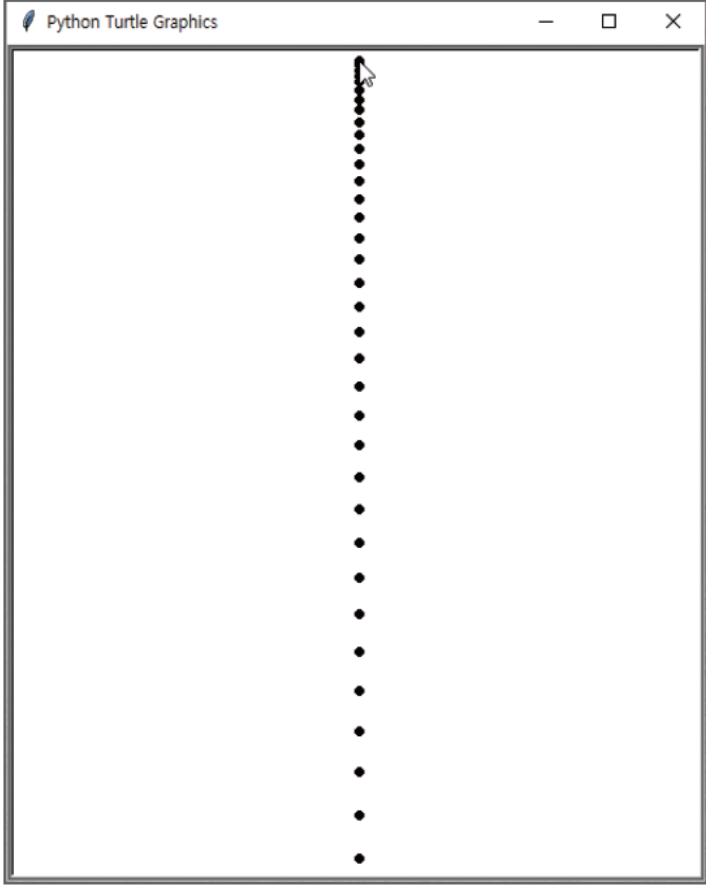
### 3 코딩

```
1  import turtle as t
2
3  def draw_pos(x, y):
4      t.clear()
5      t.setpos(x,y)
6      t.stamp()
7
8      hl = -(t.window_height() / 2)
9
10     tm = 0
11     while True:
12
13
14
15
16
17
18
19
```

```
20
21 t.setup(500, 600)
22 t.shape("circle")
23 t.shapesize(0.3, 0.3, 0)
24 t.penup()
25 s = t.Screen()
26 s.onscreenclick(draw_pos)
27 s.listen()
```



## 4 테스트/디버깅

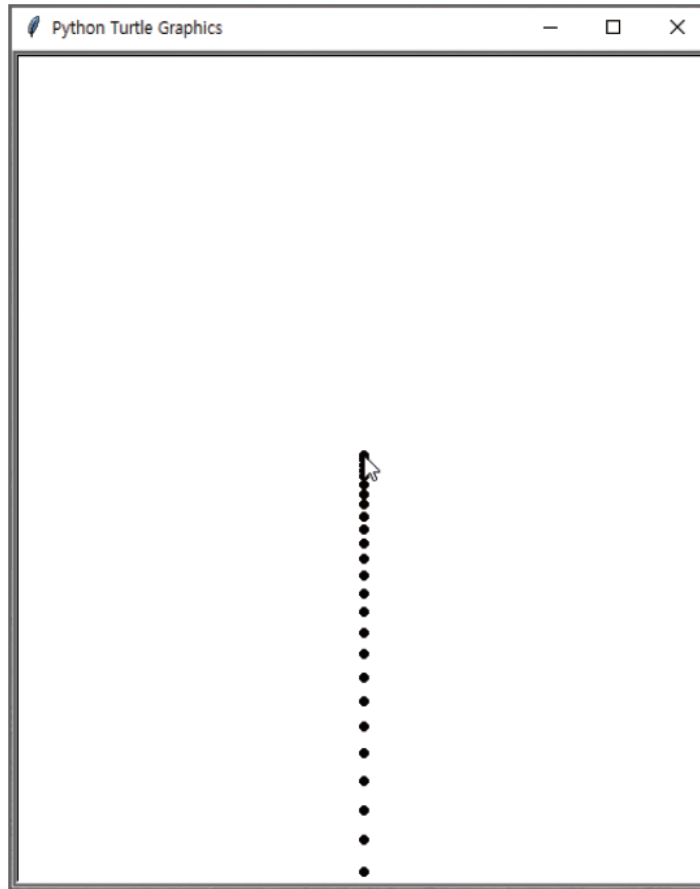
입력	결과	확인 및 수정 사항
<p>마우스 클릭</p>	 <p>The image shows a window titled 'Python Turtle Graphics'. Inside the window, a vertical line of black dots is drawn, starting from the top center and extending down to the bottom edge of the window. A mouse cursor is positioned at the top dot of this line.</p>	<p>마우스 클릭 위치부터 스크린 하단까지 궤적 점이 표시됨</p> <p>스크린 하단 부분에 도달한 경우 궤적 표시가 중단됨</p>

입력

결과

확인 및 수정 사항

마우스 클릭



스크린의 다른 위치에서 마우스를 클릭하면 기존의 궤적 점들이 모두 사라지고 해당 위치부터 스크린 하단까지 궤적 점이 다시 표시됨



### Thinking!

3. 프로그램을 실행하면 스크린 중앙에 점 형태의 터틀이 나타나 있게 된다. 또한, 궤적이 이미 표시된 후 다시 스크린의 한 지점을 클릭하여 다시 낙하를 시킬 때 클릭 위치로 이동하는 동작이 나타난다. 프로그램을 실행한 처음에 터틀이 표시되지 않도록 하고, 실행 도중 다른 위치를 클릭할 때 이동하는 동작이 나타나지 않도록 프로그램을 변경해보자.
4. 낙하 도중 터틀의 흔적을 남길 때 y 좌표인  $y$ 와 이동 거리인  $dx$ 를 터틀의 `write()` 함수를 이용하여 출력해보자.
5. 다음 결과와 같이 스크린 하단에 지면에 해당하는 선을 그린 후, 스크린을 클릭하여 궤적 점들이 나타날 때 이 선을 넘지 않도록 프로그램을 변경해보자.



## 5 프로그램 코딩을 위한 관련 학습

### ▶ 모듈 импорт(import)

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

```
import 모듈명
```

- ▶ 모듈명이 잘못되었거나 존재하지 않는 모듈을 포함할 경우 오류 발생

```
>>> import turtle
>>> turtle.shape("turtle")
>>> import turtte
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    import turtte
ModuleNotFoundError: No module named 'turtte'
```

## ▶ 모듈 импорт(import)

- ▶ 이름이 긴 모듈 등을 사용할 때 모듈의 별명을 만들어 사용 가능

```
import 모듈명 as 별명
```

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

- ▶ 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

### ▶ 콜백 함수(callback function)

- ▶ 이벤트(event)가 발생하였을 때 이벤트를 처리하기 위해 호출되는 함수
- ▶ 마우스가 클릭되는 이벤트가 발생하였을 때 onclick() 함수의 인수로 콜백 함수 등록

```
def draw_pos(x, y):          # x, y : 마우스 클릭 위치
    ...

...

s = turtle.Screen()
s.onclick(draw_pos) # 마우스 클릭이 발생하면 처리할 콜백 함수 등록
s.listen()          # 사용자 입력 위한 포커스 처리 및 이벤트 발생 확인
```

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

```
s.onclick(t.goto) # 마우스 클릭이 발생하면 터틀의 goto() 함수 호출
s.listen()
```

## ▶ 터틀 스크린의 마우스 이벤트 처리

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

```
def draw_pos(x, y):                # x, y : 마우스 클릭 위치
    ...
def move_pos(x, y):                # x, y : 마우스 클릭 위치
    ...

...
s = turtle.Screen()
s.onclick(draw_pos, 1)             # 마우스 왼쪽 버튼
s.onclick(draw_pos, 3)             # 마우스 오른쪽 버튼
s.listen()
```



## TIP

### 실행하자마자 터틀 스크린이 꺼질 경우

IDLE가 아닌 PyCharm 등의 파이썬 개발 도구를 사용할 경우 터틀 스크린이 유지되지 않고 바로 꺼지기도 한다. 이때 다음 문장을 프로그램 코드 마지막에 추가하면 된다. `mainloop()` 함수는 터틀 스크린이 종료될 때까지 마우스, 키보드 입력을 기다린다.

```
t.mainloop()
```



## &gt;&gt;&gt; 잠깐! Coding

3. 터틀 스크린에서 마우스 왼쪽 버튼을 클릭하면 해당 위치에 `stamp()` 함수로 터틀의 흔적을 남기고 `x, y` 좌표를 출력해보자. 그리고 마우스 오른쪽 버튼을 클릭하면, 클릭한 위치로 이동한 후 앞서 출력된 모든 내용을 지워보자.



# 과학 (Science)

## 1 DNA 염기서열의 순서 바꾸기

프로그램 p08-01 ... DNA 염기서열의 순서 바꾸기

관련 학습 ... 딕셔너리(dictionary) 자료형

## 2 자유 낙하와 포물선 운동 궤적 그리기

### 2.1 자유 낙하 운동 궤적 그리기

프로그램 p08-02-1 ... 물체를 자유 낙하시키기

관련 학습 ... 모듈 임포트(import)

### 2.2 자유 낙하 운동 궤적 그리기

프로그램 p08-02-1 ... 물체의 포물선 운동

관련 학습 ... math 수학 모듈, 라디안(radian)



Thinking!

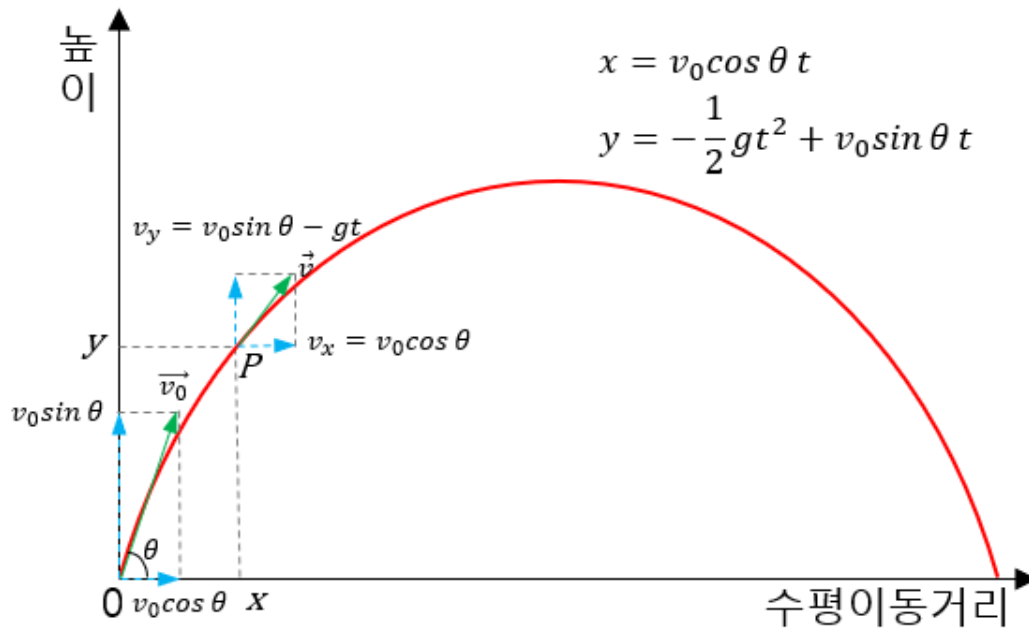
잠깐! Coding

Coding! Programming

## ▶ 포물선 운동

- ▶ 일정한 힘이 작용하는 공간에서 힘의 방향과 비스듬하게 던져진 물체가 포물선을 그리는 운동

## ▶ 포물선 운동방정식





프로그램

p08-02-2

물체의 포물선 운동

터틀 스크린의 한 지점을 클릭한 위치부터 지정한 각도로 물체를 던져 스크린의 하단 부분으로 떨어지는 포물선 운동의 궤적을 점 형태의 터틀을 이용하여 표시해보자.

## 1 문제 분석

터틀 스크린의 한 지점을 클릭한 위치부터 지정한 각도로 점 형태의 터틀을 던진다. 점 형태의 터틀은 포물선 운동 형태의 위치 이동을 하면서 변화된 각각의 지점에 터틀의 흔적을 표시한다.

함수	• draw_pos() : 포물선 운동을 하는 이동 궤적의 좌표마다 터틀의 흔적을 표시	
입력	• 마우스 클릭 : 터틀 스크린의 임의 위치를 클릭, 클릭한 위치부터 포물선 운동을 하며 스크린의 하단 부분까지 낙하	
출력	• 포물선 운동을 하는 이동 궤적의 좌표마다 터틀의 흔적을 표시	
변수	<ul style="list-style-type: none"> <li>• tm : 시간 간격</li> <li>• dx, dy : x 이동거리, y 이동거리</li> <li>• velo : 속도</li> <li>• t : 터틀</li> </ul>	<ul style="list-style-type: none"> <li>• ux, uy : x 속도, y 속도</li> <li>• g : 중력가속도 (g: 9.8)</li> <li>• ang : 각도</li> <li>• s : 터틀 스크린</li> </ul>

## 2 알고리즘 설계

매개변수

x : x 좌표

y : y 좌표

반환

없음

draw\_pos()

1. 속도를 입력받아 `velo`에 대입(기본값:50, 최솟값:10, 최댓값:100)
2. 각도를 입력받아 `ang`에 대입(기본값:45, 최솟값:0, 최댓값:360)
3. 이전에 표시한 터틀 흔적을 모두 지움
4. 터틀을 숨김
5. 터틀의 위치를 `x`, `y`로 변경
6. 터틀을 나타냄
7. 터틀의 흔적을 남김
8. 스크린의 하단에 해당하는 `y`축 위치를 계산하여 `hi`에 대입
9. `x` 속도와 `y` 속도를 계산하여 각각 `ux`, `uy`에 대입

draw_pos()	<p>10. while 문을 무한 반복</p> <p>10.1 중력가속도가 반영된 y 속도를 계산하여 uy에 대입</p> <p>10.2 y 이동 거리를 계산하여 dy에 대입</p> <p>10.3 x 이동 거리를 계산하여 dx에 대입</p> <p>10.4 만약 <math>dy &gt; h</math>이면</p> <p>10.4.1 터틀의 위치를 dx, dy로 이동</p> <p>10.4.2 터틀의 흔적을 남김</p> <p>그렇지 않으면</p> <p>10.4.1 무한 반복 중단</p>
터틀 생성 스크린 생성	<p>1. 터틀 스크린 크기를 600, 600으로 설정</p> <p>2. 터틀 모양을 circle로 설정</p> <p>3. 터틀 크기를 0.3, 0.3으로 설정, 테두리는 0으로 설정하여 표시 안함</p> <p>4. 터틀 펜을 올림</p> <p>5. 터틀 스크린 생성</p>
마우스 클릭	<p>1. 터틀 스크린에서 마우스 클릭이 이루어지면 draw_pos() 콜백 함수 호출</p> <p>2. 터틀 스크린에서의 이벤트 확인</p>

### 3 코딩

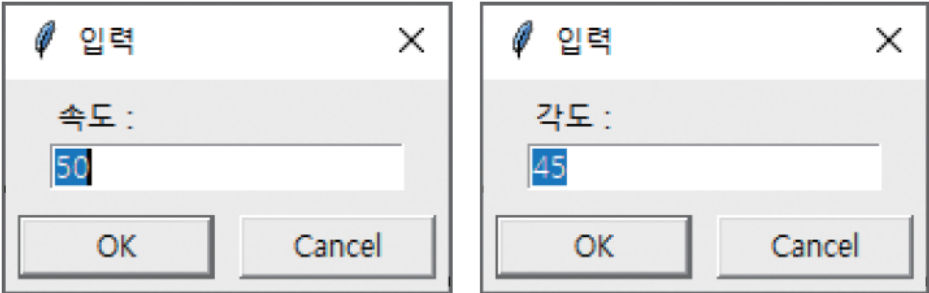
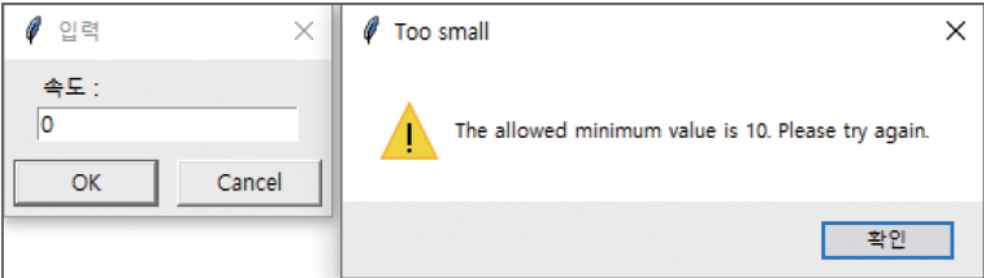
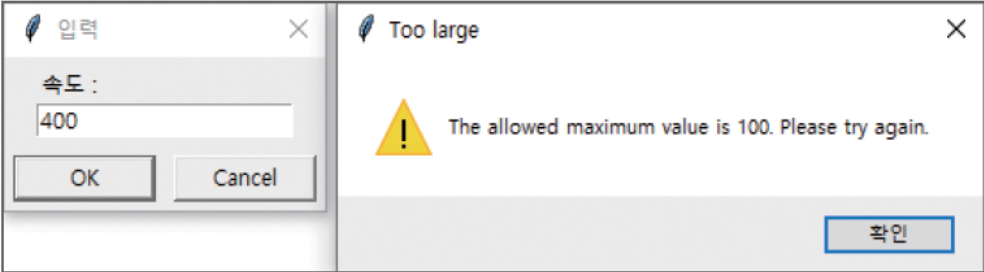
```
1  import turtle as t
2  import math
3
4  tm = 0.3
5  ux = 0
6  uy = 0
7  dx = 0
8  dy = 0
9  g = 9.8
10 velo = 0
11 ang = 0
12
```



```
13 def draw_pos(x, y):
14     velo = t.numinput("입력", "속도 : ", 50, 10, 100)
15     ang = math.radians(t.numinput("입력", "각도 : ", 45, 0, 360))
16
17     t.clearstamps()
18     t.hideturtle()
19     t.setpos(x,y)
20     t.showturtle()
21     t.stamp()
22
23     hl = -(t.window_height() / 2)
24
25     ux =
26     uy =
27
```

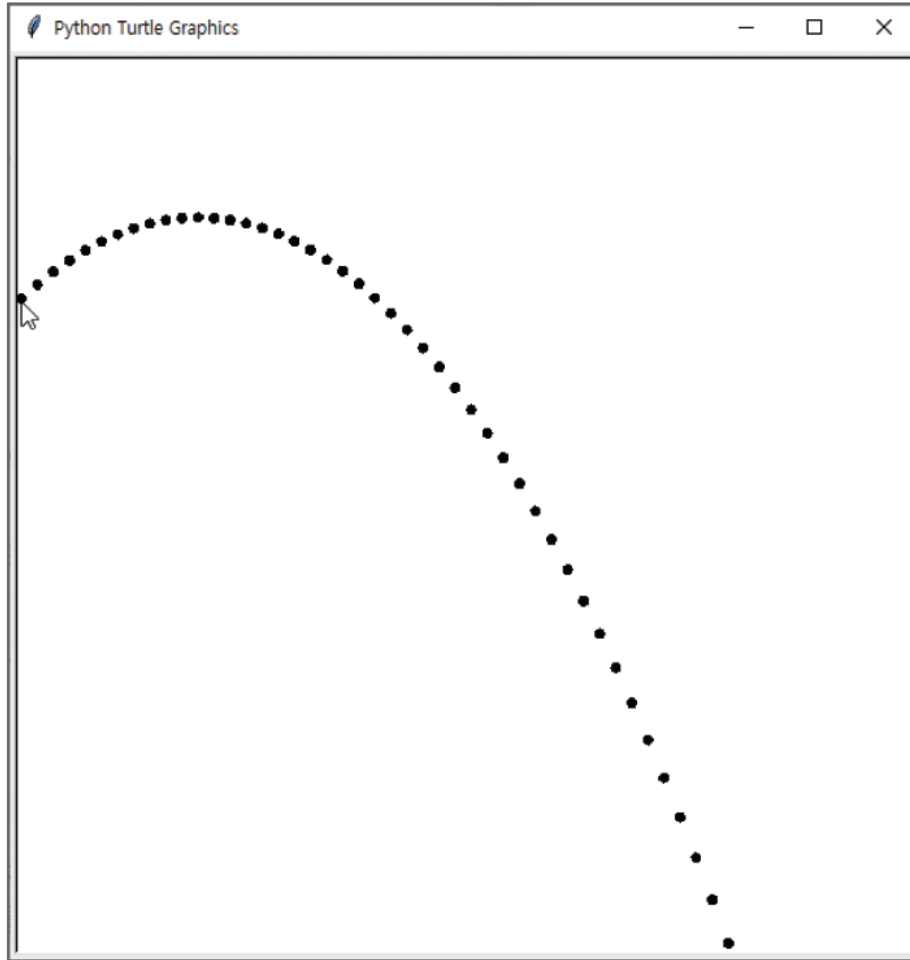
```
28     while True:
29
30
31
32
33
34
35
36
37
38     t.setup(600, 600)
39     t.shape("circle")
40     t.shapesize(0.3, 0.3, 0)
41     t.penup()
42     s = t.Screen()
43     s.onscreenclick(draw_pos)
44     s.listen()
```

## 4 테스트/디버깅

입력	결과	확인 및 수정 사항
마우스 클릭		속도와 각도를 입력받기 위한 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

속도: 50  
각도: 45

마우스 클릭



마우스 클릭 위치부터 스크린 하단까지 궤적 점이 표시됨

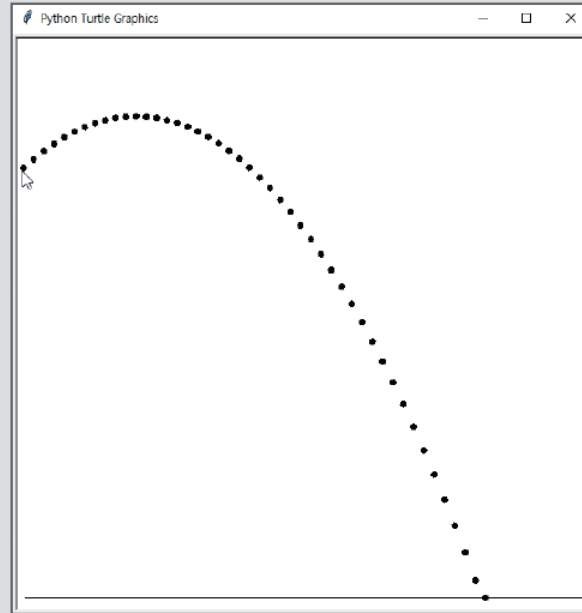
스크린 하단 부분에 도달한 경우 궤적 표시가 중단됨

스크린의 다른 위치에서 마우스를 클릭하면 기존의 궤적 점들이 모두 사라지고 해당 위치부터 스크린 하단까지 궤적 점이 다시 표시됨



## Thinking!

6. 다음 결과와 같이 스크린 하단에 지면에 해당하는 선을 그린 후, 스크린을 클릭하여 궤적 점들이 나타날 때 이 선을 넘지 않도록 프로그램을 변경해보자.



## 5 프로그램 코딩을 위한 관련 학습

### ▶ math 수학 모듈

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

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

- ▶ 내장 함수 공식 문서 :

<https://docs.python.org/3/library/functions.html>

## ▶ math 수학 모듈

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

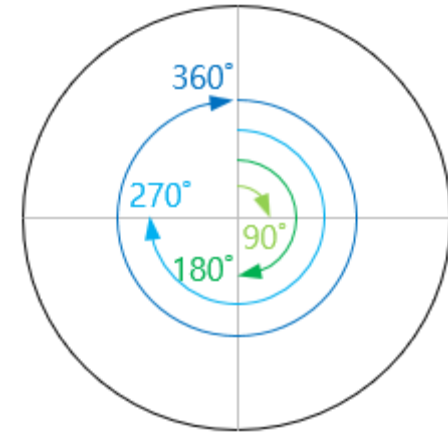
함수/상수	기능	반환값
<code>log(x)</code>	x의 로그를 구함	float
<code>sqrt(x)</code>	x의 제곱근을 구함	float
<code>radians(x)</code>	각도 x를 라디안 값으로 변환	float
<code>sin(x)</code>	라디안 x의 사인(sin)을 구함	float
<code>cos(x)</code>	라디안 x의 코사인(cosine)을 구함	float
<code>pi, e</code>	pi: 3.141592..., e: 2.718281...	

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

## ▶ 라디안(radian)

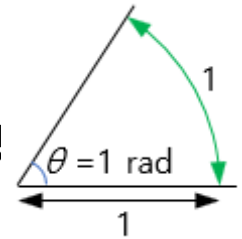
### ▶ 도(degree)

- ▶ 일상적으로 사용하는 각도의 단위, 원 한 바퀴를 360도로 표현하는 방법
- ▶ 반원은 180도, 직각은 90도와 같이 표현



### ▶ 라디안(radian)

- ▶ 각도를 표현하는 또 다른 방법
- ▶ 호의 길이가 반지름과 같게 되는 만큼의 각도를 1라디안이라고 정
- ▶ 1라디안은 약 57.3도 해당 각도





## ▶ 라디안(radian)

### ▶ 도 > 라디안 변환

- ▶ 도 \* 3.14 / 180
- ▶ radians() 함수

```
>>> import math
>>> a = 45
>>> r = a * 3.14 / 180.0          # 도 -> 라디안
>>> r
0.785
>>> r = a * math.pi / 180.0     # 도 -> 라디안
>>> r
0.7853981633974483
>>> r = math.radians(a)         # 도 -> 라디안
>>> r
0.7853981633974483
>>> print(math.sin(r), math.cos(r))
0.7071067811865475 0.7071067811865476
```

## ▶ 라디안(radian)

### ▶ 라디안 > 도 변환

- ▶ 라디안 \* 180 / 3.14
- ▶ degrees() 함수

```
>>> import math
>>> r = 1
>>> a = r * 180.0 / 3.14          # 라디안 -> 도
>>> a
57.324840764331206
>>> a = r * 180.0 / math.pi      # 라디안 -> 도
>>> a
57.29577951308232
>>> a = math.degrees(r)          # 라디안 -> 도
>>> a
57.29577951308232
```

## &gt;&gt;&gt; 잠깐! Coding

4. 변수 `a`에 `[-3, 7, 9, 4, 2, 3]`을 대입한 후, `a[0]` 값을 출력하고, 내장 함수를 이용하여 `a[0]`의 절댓값, `a`의 최댓값, `a`의 최솟값, `a`의 총합을 구해보자. 또한, 2의 3승 값을 구해보자.
5. `math` 모듈을 이용하여 90도의 라디안을 구해보자. 또한, 2 라디안의 도를 구해보자.

