35장

유용한 라이브러리

35장 유용한 라이브러리

35.1 라이브러리 임포트하기

35.1 라이브러리 임포트하기

» 다른 사람이 이룩해 놓은 일을 바탕으로 프로그램을 만들 수 있어야 프로그래밍이 더 효율적이고 즐거울 수 있다. 이미 해결된 문제가 많기 때문에 풀려는 문제와 비슷한 문제의 해법을 구현한 코드가 있을 가능성이 높다. 따라서 아무 것도 없는 상태에서 코드를 처음부터 하나하나 구현하면서 문제를 해결해야 할 경우는 거의 없다. 어떤 프로그래밍 언어든 프로그래머가 코드를 작성하는 과정에서 유용하게 쓸 수 있는 라이브러리가 있기 마련이다. 이렇게 이미 만들어지고 효율성과 올바름을 보장하기 위한 테스트 및 디버깅이 이뤄진 코드를 바탕으로 코드를 작성하는 것을 모듈화 방식이라고 말한다.

» 어떤 파일에 함수나 클래스 정의가 들어 있다면, 여러분의 코드 파일 맨 앞에 그 파일을 import해 사용한다.

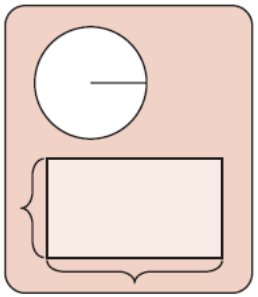
- 함수나 클래스 정의를 다른 파일에 넣는 이유는 코드를 좀 더 체계적으로 관리하기 위함이다. 이는 (함수나 클래스를 사용한) 추상화와 같은 사고 방식이다.

35.1 라이브러리 임포트하기

- Circle과 Rectangle 정의가 들어 있는 파일 이름이 shapes.py라고 하자. 다음과 같이 다른 파일에서 그 파일에 있는 클래스를 불러올 수 있다. 이 과정을 임포트(import)라고 하며, 파이썬에게 shapes.py라는 파일에 있는 모든 클래스 정의를 가져오라고 명령하는 것이다.

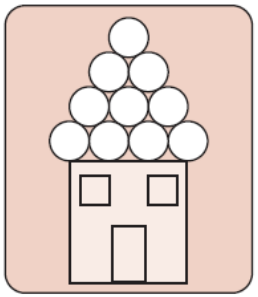
```
import shapes
```

shapes.py
(모양 클래스 정의)



```
class Circle:
  # 코드
class Rectangle:
  # 코드
```

test.py
(모양 객체를 가져와
원하는 대로 사용)



```
import shapes
c1 = Circle()
r1 = Rectangle(1,2)
# 다른 코드
```

- 임포트는 코드를 체계적으로 정리하고 관리하기 위한 가장 일반적인 기법이다. 보통 라이브러리를 코드에 임포트해 사용하게 된다. 라이브러리는 보통 함께 쓰일 수 있는 모듈의 모음이다.
- 언어가 기본으로 제공하는 것(언어를 설치할 때 함께 들어 있는 내장 라이브러리)과 제3자(서드파티(third-party) 라이브러리(웹 등에서 내려 받은 라이브러리)가 있다.
- https://docs.python.org/ko/3/library/index.html

35.2 math 라이브러리로 수학 연산 사용하기

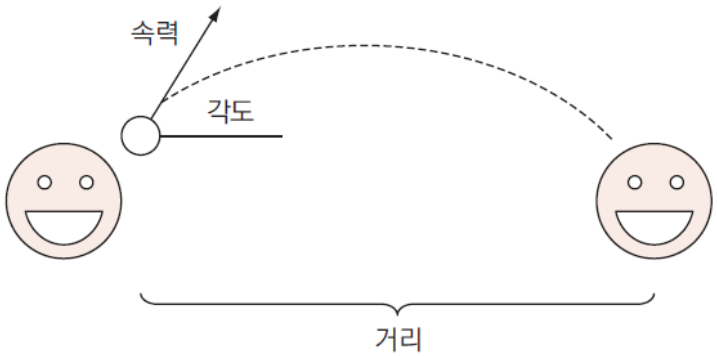
35.2 라이브러리 임포트하기

math 라이브러리는 수를 처리하는 연산 중 파이썬 언어가 기본 제공하지 않는 연산을 처리한다.

- https://docs.python.org/3.7/library/math.html

```
import math
help(math)
```

- 운동장에서 친구에게 던진 공을 시뮬레이션한다고 하자. 던진 공을 친구가 받을 수 있는지 살펴보려 한다.
- 먼저 사용자에게 친구와의 거리, 던진 공의 속력, 공을 던진 각도(수평선 기준)를 물어본다. 프로그램은 공이 친구가 잡을 수 있는 위치까지 도달할 수 있는지 알려줄 것이다.



$$\text{거리} = 2 * \text{속력}^2 * \sin(\text{각도}) * \cos(\text{각도}) / 9.8$$

35.2 라이브러리 임포트하기

```
import math ---- 수학 라이브러리인 math 라이브러리를 임포트함

distance = float(input("친구와 당신의 거리는 얼마입니까? (m) "))
speed = float(input("당신이 던진 공의 속력은 얼마입니까? (m/s) "))
angle_d = float(input("던진 공이 지면과 이루는 각도는 얼마입니까? (도) "))
tolerance = 2
angle_r = math.radians(angle_d) ---- 입력받음. 따라서 사용자 입력을 라디안으로 변환함
                                    math.sin과 math.cos는 각도로 도(degree)가 아닌 라디안을
reach = 2*speed**2*math.sin(angle_r)*math.cos(angle_r)/9.8 ---------- 수학 라이브러리의 함수를
                                                                      사용해 공식을 구현
if reach > distance - tolerance and reach < distance + tolerance:
    print("나이스 피처!")
elif reach < distance - tolerance:
    print("충분히 멀리 던지지 못했습니다.")
else:
    print("너무 멀리 던졌습니다.")
```

코드 35-1
math 라이브러리를 사용해 공의 이동 거리 계산하기

» 셀프 체크 35.2

35.3 random 라이브러리를 사용해 난수 활용하기

35.3 random 라이브러리를 사용해 난수 활용하기

» **random 라이브러리는 프로그램에 예측 불가능한 성질을 추가할 때 사용할 수 있는 여러 연산을 제공한다.**

- https://docs.python.org/3.7/library/random.html

- 이 코드는 이름이 들어 있는 리스트의 원소 중 하나를 무작위로 출력한다.

```
import random
people = ["현석","도남","일민","혜원","성원","정원"]
print(random.choice(people))
```

- 이 코드는 똑같은 이름이 두 번 뽑히지 않게 보장하면서 지정한 개수(위 코드에서는 3)만큼 리스트에서 원소를 뽑아 출력한다.

```
import random
people = ["현석","도남","일민","혜원","성원","정원"]
print(random.sample(people, 3))
```

35.3.2 확률 게임 시뮬레이션하기

난수 라이브러리의 일반적인 사용법 중 하나를 더 들면 확률 게임을 플레이하는 경우다. 가위-바위-보 게임

- 가위, 바위, 보의 1/3 확률을 시뮬레이션하기 위해 난수가 0~1/3, 1/3~2/3, 2/3~1 중 어느 범위에 속하는지 검사한다.

```
import random ---- random 라이브러리에 정의된 함수를 불러옴

choice = input("가위, 바위, 보 중 하나를 고르세요: ")
r = random.random() ---- 0 이상 1 미만의 부동소수점 난수를 발생시킴
if r < 1/3: ---- 1/3 확률로 컴퓨터가 바위를 선택한 경우
    print("컴퓨터는 바위를 냈습니다.")
    if choice == "보":
        print("당신이 이겼습니다!")
    elif choice == "가위":
        print("당신이 졌습니다.")
    else:
        print("비겼습니다.")
elif 1/3 <= r < 2/3: ---- 1/3 확률로 컴퓨터가 보를 선택한 경우
```

35.3.2 확률 게임 시뮬레이션하기


```
    print("컴퓨터는 보를 냈습니다.")
    if choice == "가위":
        print("당신이 이겼습니다!")
    elif choice == "바위":
        print("당신이 졌습니다.")
    else:
        print("비겼습니다.")
else: ---- 1/3 확률로 컴퓨터가 가위를 선택한 경우
    print("컴퓨터는 가위를 냈습니다.")
    if choice == "바위":
        print("당신이 이겼습니다!")
    elif choice == "보":
        print("당신이 졌습니다.")
    else:
        print("비겼습니다.")
```

코드 35-2
random 라이브러리를 사용해 가위-바위-보 게임 시뮬레이션하기

35.3.3 시드를 사용해 똑같은 난수 결과를 다시 만들어내기

- 난수 라이브러리가 만든 난수는 진짜 난수는 아니고 의사난수다. 그 수들은 난수처럼 보이지만 실제로는 미리 정해진 수학적 규칙으로 만들어진 수의 시퀀스다. 시퀀스를 결정하는 핵심이 되는 수를 **시드(seed)**라고 부른다. random.seed(N)을 사용해 난수의 시드를 지정할 수 있다(N은 임의의 정수).
- 다음은 2 이상 17 미만, 30 이상 88 미만의 난수를 발생시킨다. 이 프로그램을 여러 번 실행하면 출력되는 값이 매번 다를 것이다.

```
import random
print(random.randint(2,17))
print(random.randint(30,88))
```

- 하지만 시드를 지정하면 프로그램을 실행할 때마다 같은 수가 출력된다.

```
import random
random.seed(0)
print(random.randint(2,17))
print(random.randint(30,88))
```

» 셀프 체크 35.3

35.4 time 라이브러리를 사용해 프로그램 실행에 걸리는 시간 측정하기

35.4 time 라이브러리를 사용해 프로그램 실행에 걸리는 시간 측정하기

1부터 100만까지 카운트를 세는 데 걸리는 시간을 측정해 보자.

- 카운터를 증가시키는 루프 바로 앞에서 현재 시간을 나타내는 컴퓨터 클럭을 저장한다.
- 그 후 루프를 실행하고, 루프가 끝나면 다시 컴퓨터 클럭이 가리키는 현재 시간을 가져온다.
- 시작 시간과 끝 시간 사이의 차이가 루프가 실행되는 데 걸린 전체 시간이다

```
import time ---- time 라이브러리에 정의된 함수를 불러옴

# 원서 코드는 time.clock()이었으나 3.7부터는 사용이 금지됐다.
# 대신 time.perf_counter()를 써야 한다.
# pref_counter()의 값은 나노초 단위까지 돌려주며, 정밀도는 시스템에 따라 다르다

start = time.perf_counter() ---- 클럭의 현재 시간을 나노초(ns) 단위로 가져옴

count = 0                   ----
for i in range(1000000):    ---- 100만 번 카운트를 세는 코드
    count += 1              ----

end = time.perf_counter() ---- 클럭의 현재 시간을 나노초(ns) 단위로 가져옴
print(end-start) ---- start와 end 사이의 차이를 출력
```

코드 35-3 time 라이브러리를 사용해 프로그램 실행 시간 측정하기

35.4.2 프로그램 일시 중단하기

프로그램을 일시적으로 중단시킬 수 있는 sleep이라는 함수가 있다.

- 이 함수는 인자로 전달받은 시간 동안 프로그램이 다음 줄로 진행하지 못하게 막는다.
- 0.5초마다 10%씩 늘어나는 프로그래스 바(progress bar)를 보여준다.

```
import time ---- time 라이브러리에 정의된 함수를 불러옴
print("로딩중...")
for i in range(10): ---- 10%씩 증가하는 프로그래스 바를 표현하는 루프
    print("[",i*"*",(10-i)*" ","]",i*10,"% 완료") ---- * 문자 여러 개로 표시되는 프로그래스 바를 출력
    time.sleep(0.5) ---- 0.5초간 프로그램을 일시 중단
```

```
로딩중...
[          ] 0 % 완료
[ *        ] 10 % 완료
[ **       ] 20 % 완료
[ ***      ] 30 % 완료
[ ****     ] 40 % 완료
[ *****    ] 50 % 완료
[ ******   ] 60 % 완료
[ *******  ] 70 % 완료
[ ******** ] 80 % 완료
[ *********] 90 % 완료
```

코드 35-4
time 라이브러리를 사용해 프로그래스 바 표시하기

셀프 체크 35.4

35.5 요약

35.5 요약

- 비슷한 기능을 하는 코드를 별도의 파일로 분리해 체계적으로 관리하면 코드를 좀 더 읽기 좋게 만들 수 있다.
- 라이브러리는 한 그룹에 속하는 동작과 관련 있는 함수와 클래스를 한 곳에 저장한다

35.5 요약

» (Q35.1) 사용자가 컴퓨터와 주사위를 굴리는 게임을 프로그램으로 작성하라. 먼저 사용자가 정육면체 주사위를 굴리는 동작을 시뮬레이션해서 결과를 사용자에게 표시하라. 그 후 컴퓨터가 정육면체 주사위를 굴리는 동작을 시뮬레이션하고, 2초 지연 시간을 둔 다음, 결과를 보여준다. 매번 주사위를 굴릴 때마다 사용자에게 다시 주사위를 굴릴지 물어보라. 사용자가 게임을 모두 마치면 얼마나 오랫동안 게임을 진행했는지 초 단위로 시간을 표시하라.