APPENDIX

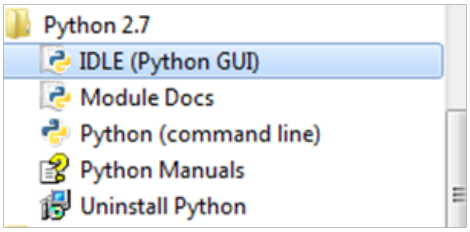
파이썬과 IDLE 시작하기[1]

파이썬 환경을 구축하는 방법과 IDLE을 소개하겠습니다.

파이썬 설치하기

당신의 컴퓨터에서

만약 개별 컴퓨터를 사용한다면 파이썬을 설치해야 합니다. 다음 링크에서 다운받을 수 있습니다. https://www.python.org/downloads/ 버전 3.x(혹은 버전 2.7)를 다운로드해서 설치하면 됩니다.[2] 시작 메뉴에서 Python을 선택한 후, 파이썬 개발 도구인 IDLE(Python GUI)을 선택하여 파이썬 프로그래밍을 시작할 수 있습니다.



Linux 공용 서버에서

여러분이 사용하는 대부분 Linux 공용 서버에는 파이썬이 설치 되어 있을 것입니다. 서버에 접속한 후에, 파이썬이 제대로 동작하는지 확인하기 위해 command prompt에서 'idle' 을 입력하세요. 이제 파이썬 개발 환경 IDLE이 시작될 것입니다. 만약, 서버가 IDLE 명령어를 인식하지 못할 경우, 서버 운용자에게 파이썬 설치를 요청하길 바랍니다.

1. [역자 주] 참고문헌: http://web.mit.edu/6.s189/www/handouts/GettingStarted.html
2. [역자 주] 파이썬 3.x 최신 버전을 다운받아 설치하면 됩니다. 2.7.x 버전도 아직 많이 사용하고 있습니다.

IDLE 사용하기

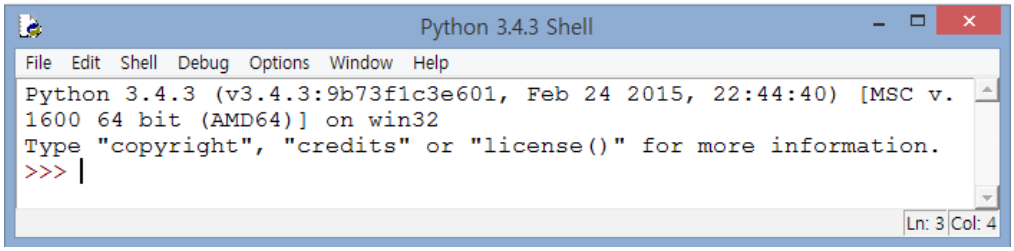
IDLE은 기본 파이썬 개발 환경이고, "Integrated DeveLopment Environment"의 줄임입니다. Linux와 Windows 플랫폼에 모두 동작합니다.

파이썬 쉘Shell이 기본적으로 설치되어 있으며, 이것을 통해 파이썬 **상호작용 모드**interactive mode로 접근할 수 있습니다. 또한 파이썬 소스 파일을 만들고 이미 존재하는 파일을 수정할 수 있도록 도와주는 파일 편집기editor도 있습니다.

IDLE의 특징에 대한 여기에 기술된 것을 눈으로만 읽지 말고 여러분이 직접 실행해보면 좋겠습니다.

상호적인 파이썬 쉘(Shell)

IDLE을 시작하면, 상호적인 파이썬 쉘을 포함하는 창이 나타납니다.



이 쉘의 '>>>' 프롬프트prompt에 바로 파이썬 코드를 입력할 수 있습니다. 코드를 다 입력한 후, 실행할 수 있습니다. 가령 다음을 입력하고

```
>>> print("hello world")
```

ENTER 키를 누르면, 다음이 화면에 출력됩니다.

```
hello world
```

IDLE은 계산기로도 사용할 수 있습니다.

```
1  >>> 4+4
2  8
```

덧셈, 뺄셈, 곱셈 연산자는 파이썬 언어에 내장되어 있습니다. 당장 사용할 수 있다는 말이지요. 만약 계산에 제곱근을 사용하고 싶다면, math 모듈을 import 해야 합니다. 지금은 이게 무슨 의미인지 몰라도 상관 없습니다. 수업을 하며 차차 이런 내용을 다룰 것이기 때문입니다. 아래는 제곱근을 계산하는 예제입니다.

```
>>> import math
>>> math.sqrt(16)
4.0
```

math 모듈을 사용하면 여러 가지 유용한 연산도 가능합니다.

```
>>> import math
>>> math.pow(3, 2)
9.0
>>> math.cos(0)
1.0
```

IDLE을 실행한 후에, import 명령은 한 번만 실행하면 된다는 것을 기억하세요.

연습

(이것은 단지 연습용 문제이기에, 여러분의 답이 어디 기록되지 않으며, 답을 제출할 필요가 없습니다.) IDLE을 사용해서 다음을 계산하세요.

1. 6 + 2 * 10
2. (6 + 2) * 10 (위의 문제와 답을 비교해보세요. 다행히도 수학에서 사용하는 연산법칙들이 파이썬에서도 똑같이 적용됩니다.)
3. 23.0의 5승
4. 다음 수식의 양의 근을 구해보세요:

   34*x^2 + 68*x - 510

   다음과 같은 공식을 기억하고 있나요?

   a*x^2 + b*x + c

   x1 = (- b + sqrt (b*b - 4*a*c)) / (2*a)

문제 세트 0[1]
: 이름을 입력하고 출력하기

서문

이 문제 세트에서는 프로그래밍 환경인 IDLE과 파이썬을 이용한 프로그래밍, 그리고 일반적인 문제 세트의 구조를 소개하겠습니다. IDLE을 설치하고 간단한 파이썬 프로그램을 작성하여 제출할 것입니다. **이번 문제 세트를 철저하게 읽으세요. 특히 공동 작업과 제출 과정에 대한 부분은 중요합니다.**

- **작업량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상한 것보다 너무 오랜 시간이 걸리는 문제들로 인하여 여러분에게 과도한 학업 부담을 주지 않도록 주의하려고 합니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 학생은 각각 개별적으로 답안을 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

파이썬과 IDLE 설치하기

'파이썬과 IDLE 시작하기'[2]에 명시되어 있는 과정을 따라가든지, 아니면 다른 방법을 찾아 여러분이 사용할 컴퓨터에 파이썬과 IDLE을 설치하세요. 여러분이 파이썬이 이미 설치된 공용 서버를 접근할 수 있다면 설치 과정은 생략해도 됩니다.

1. [역자 주] 참고문헌:모든 문제 세트에 필요한 파일은 MIT OCW에서 다운로드 받을 수 있습니다. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/
2. [역자 주] 참고문헌: http://web.mit.edu/6.s189/www/handouts/GettingStarted.html

앞장의 연습문제로 파이썬과 IDLE에 익숙해지세요. 모두 준비가 되었다면 이 과제의 프로그래밍 부분으로 넘어가세요.

매우 간단한 프로그램: 이름을 입력하고 출력하기

이 프로그래밍 연습문제는 IDLE을 사용하는 것이 더 익숙해지고 간단한 파이썬의 요소들을 사용할 준비를 하기 위한 것입니다. 프로그램의 표준 요소로는 결과를 출력하고(print를 사용하여), 콘솔 창에서 사용자의 입력 값을 읽어오고(예를 들어 raw_input 혹은 input을 사용하여), 변수에 값을 저장하여 프로그램이 그 값을 이용하는 것들이 있습니다.

문제 1

다음과 같은 순서로 실행하는 프로그램을 만드세요.

1. 사용자에게 성을 입력 받으세요.
2. 사용자에게 이름을 입력 받으세요.
3. 이름과 성 순으로 출력하세요.

여러분의 프로그램이 아래의 예제와 같이 실행되어야 합니다(기울임꼴italic 글자는 입력 값을 뜻합니다. **은 컴퓨터가 출력하도록 하세요.)

```
1 Enter your last name:
2 **Grimson
3 Enter your first name:
4 **eric
5 eric
6 Grimson
```

힌트 : print 명령어 혹은 print 함수를 사용하는 방법을 알아보고 싶다면 파이썬 Wikibook의 input and output 부분을 찾아보세요. 문자열을 출력하기 위해 print문을 사용하는 방법을 알려줄 것입니다.[3]

사용자의 콘솔 창에서 사용자의 입력 값을 읽어 파이썬 환경으로 가져오는 방법을 알아보고 싶다면 동일한 부분을 읽어보세요(raw_input 혹은 input 함수).[4]

3. [역자 주] https://en.wikibooks.org/wiki/Python_Programming/Input_and_Output
4. [역자 주] 파이썬 3.x에서 raw_input()은 input()으로 이름이 바뀌었습니다. input()은 sys.stdin으로부터 한 줄을 읽고, 끝에 있는 newline을 제거한 것을 돌려줍니다. 만약 입력이 잘못되면 EOFError를 발생합니다. raw_input()와 같은 함수가 필요하면 eval(input())로 대체가 가능합니다.

값을 계속 유지하고 싶다면, 그 값을 변수에 저장해야 합니다(즉, 값을 저장하고 싶다면 값을 지칭할 이름을 지정해 줍니다.) 파이썬 Wikibook의 variables and strings 부분을 찾아보세요.

문제 1에 대한 코드를 ps0_1.py에 저장하세요.

제출 과정

저장. ps0.py 안에 수정한 코드를 작성하세요. 다른 이름으로 파일을 저장하면 안됩니다.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 파일 이름과 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적어주세요. 또한 공동 작업을 한 학우(들)이 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
1  # ps0.py
2  # Problem Set 0
3  # Name: Jane Lee
4  # Collaborators: John Doe
5  # Time: 3:30
6  ... your code goes here ...
```

문제 세트 1
: 소수(素數, Prime Number)를 계산하기

서문

이번 문제 세트에서는 파이썬의 제어 흐름과 문제를 해결하기 위해 계산적으로 접근하는 방법을 소개하겠습니다. 여러분은 간단한 파이썬 프로그램을 디자인하고, 작성하여 테스트한 후에 제출해야 할 것입니다. 이번 문제 세트를 철저하게 읽으세요. 특히 공동 작업과 제출 과정에 대한 부분은 중요합니다.

- **작업량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상한 것보다 너무 오랜 시간이 걸리는 문제들로 인하여 여러분에게 과도한 학업 부담을 주지 않도록 주의하려고 합니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 학생은 각각 개별적으로 답안을 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

소수(素數)Prime Number를 계산하기

일반적으로 컴퓨터에서 계산을 할 때에는 '만들고-확인하기generate-and-test' 방법을 사용합니다. 해당 문제에 대한 가능한 답들을 체계적으로 만들어 내고, 일련의 테스트를 한 번 이상 실행하여 그 답이 유효한지를 결정합니다. 원칙적으로 가능한 답들을 무작위로 혹은 어떠한 확률 분포에 의거하여 만들 수도 있습니다(어떤 경우에는 이렇게 해야만 합니다). 그러나 많은 경우에 답이 될 가능성이 있는 모든 후보 군의 답들을 체계적으로 만드는 방법을 찾아내는 것이 더욱 효율적입니다.

문제 1

1000번째 소수를 계산하고 출력하는 프로그램을 만드세요.

힌트 : 여러분이 코드 작성을 시작하는데 도움이 될 만한 단계들을 나열했습니다.

1. 상태 변수를 초기화하세요.
2. 1보다 큰 모든 정수(홀수)들을 소수의 후보로 만드세요.
3. 각각의 후보 정수에 대해서 그 수가 소수인지 테스트 하세요.
   ① 이것을 실험하기에 간단한 한 가지 방법은 후보 소수를 1보다 큰 다른 모든 정수들로 나눌 때 나머지가 없는지 확인하는 겁니다. 이것을 하기 위해, 모듈로modulo 연산을 사용하면 됩니다. 예를 들어, a%b는 정수 a를 정수 b로 나눈 나머지를 반환합니다.
   ② 어떠한 정수들로 나눠야 하는지 생각해봐야 합니다. 물론 소수 후보보다 큰 정수들로는 나눌 필요가 없겠지요. 하지만 소수 후보까지 가기 전에 어느 정도 큰 정수까지 가서 나누기를 멈출 수 있을까요?
4. 만약 소수 후보가 소수라면, 연산이 현재 어느 정도 진행되고 있는지 확인할 수 있도록 정보를 출력하세요. 후에 상태 변수를 업데이트 하세요.
5. 특정 종료 조건에 도달하면 멈추세요. 이 조건을 만들 때에, 프로그램이 첫 번째 소수(2)는 만들지 않았다는 사실을 잊지 마세요.

이러한 아이디어를 코드를 만드는 데에 있어서 가이드로 사용하세요

문제 1에 대한 코드를 ps1a.py에 저장하세요.

만약 코드가 제대로 소수를 찾았는지 확인하고 싶다면, 다음 링크에서 소수들을 확인할 수 있습니다. http://primes.utm.edu/lists/small/1000.txt

소수들의 곱

n이 충분히 클 경우, n보다 작은 소수들의 곱은 e**n보다 작거나 같으며, n이 증가함에 따라 이것은 tight bound[1]가 된다(즉, n이 증가할수록 소수의 곱과 e**n의 비율이 1에 가까워진다는 의미입니다)는 정수론(整數論)number theory에 있는 하나의 재미있는 결과입니다.

매우 큰 수의 경우, 그 큰 수보다 작은 소수들의 곱을 계산하면, 그 결과값은 매우 큰 숫자가 될 것입니다. 나중에 이런 계산을 하게 된다면 문제를 야기할 수 있겠죠(수업의 뒤쪽 부분에서 컴

1. [역자 주] 9장 알고리즘 복잡도(complexity)를 참고하세요.

퓨터가 어떻게 큰 수를 다루는지 살펴볼 것입니다). 그런데, 우리는 소수들의 곱을 각각의 소수에 로그를 취한 것의 합으로 변환할 수 있습니다.[2] 위에서 언급한 명제[3] 각 부분에 로그를 적용하면서 할 수 있는 것이죠. 결국 이를 종합하면 n보다 작은 소수들에 로그를 취하여, 모두 합한 것은, n보다 작게 됩니다. n 값이 증가할수록 n까지의 로그를 취한 소수들의 합과 n 값의 비율은 1에 가까워지는 것으로 명제를 간소화할 수 있습니다.[4]

로그를 계산하기 위해 우리는 파이썬에 내장된 수학 함수를 사용할 수 있습니다. 이 함수를 사용하기 위해 우선 해당 라이브러리를 파이썬 환경으로 import해야 합니다. 다음 줄을 파일 시작 부분에 추가하세요.

```
from math import *
```

이제 코드에 `log` 함수를 사용할 수 있습니다. 예를 들어 `log(2)`는 숫자 2에 대해 e를 밑수로 하는 로그 값을 반환합니다.

문제 2

2 ~ n까지의 모든 소수들의 로그 값을 합하는 프로그램을 만드세요. 그리고 소수에 로그를 취한 값들의 합과 n의 비율을 출력하세요. 다양한 n 값에 대해 시행해보세요.

문제 1을 조금만 수정해서 이 문제를 풀 수 있어야 합니다.

힌트 : 소수에 로그를 취한 값들의 합과 n의 비율이 점점 1에 천천히 근접하는 것을 보겠지만, 단조적으로monotonically 근접하지는 않습니다.

문제 2에 대한 코드를 ps1b.py에 저장하세요.

제출 과정

저장. 문제 1과 문제 2를 각각 ps1a.py, ps1b.py 안에 작성하세요. 파일의 이름을 바꾸지 마세요.

---

2. [역자 주] $\log_b xy = \log_b x + \log_b y$
3. [역자 주] n보다 작은 소수들의 곱은 $e^{**}n$보다 작거나 같다. 즉 $n_1 * n_2 * n_3 * \cdots\cdots \le e^n$, 여기서 n_1, n_2, n_3, ……는 n보다 작은 모든 소수들입니다. e는 오일러 상수(Euler Number), 즉 2.718281828459 … 입니다.
4. [역자 주] $\log n_1 + \log n_2 + \log n_3 + \cdots\cdots \le \log e^n = n$

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 파일 이름과 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적어주세요. 또한 공동 작업을 한 학우(들)이 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# ps1a.py
# Problem Set 1
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

문제 세트 2
: 맥디오판틴(McDiophantine)

서문

이 문제 세트에서는 파이썬에서 제어 구조와 문제 해결 방법으로 전수조사exhaustive search를 사용하는 방법을 소개하겠습니다.

- **작업량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상한 것보다 너무 오랜 시간이 걸리는 문제들로 인하여 여러분에게 과도한 학업 부담을 주지 않도록 주의하려고 합니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 학생은 각각 개별적으로 답안을 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

맥디오판틴McDiophantine: 맥너겟 팔기

수학에서 **디오판틴 방정식**Diophantine equation – 3세기 그리스 수학자인 알렉산드리아의 디오판투스Diophantus의 이름을 딴 것 – 은 변수의 값이 항상 정수인 다항 방정식입니다. 아마 알아채지 못했겠지만, 여러분은 디오판틴 방정식을 전에 본 적이 있습니다. 가장 유명한 디오판틴 방정식 중 하나는

$$x^n + y^n = z^n$$

입니다. $n = 2$일 때, 이 방정식을 만족하는 해 x, y, z는 무한히 많으며, 이를 피타고라스 수Pythagorean triples라고도 하지요(예를 들어 $3^2 + 4^2 = 5^2$). 의 값이 증가했을 때, 페르마의 유명한

"최후의 정리last theorem" 에 따르면 이 방정식을 만족하는 양의 정수해(整數解)integer solution x, y, z는 존재하지 않습니다. 페르마의 최후의 정리 이외에도 몇 세기 동안 수학자들은 다른 디오판틴 방정식을 연구했습니다. 그 중 유명한 방정식은 펠 방정식Pell' s equation과 에르되시-스트라우스 추측Erdos-Strauss conjecture을 포함합니다. 이 수학의 흥미로운 분야에 대해 더 알고 싶다면 위키피디아Wikipedia의 내용을 찾아 보세요.

맥도날드가 디오판틴 방정식을 알고 있다는 것을 우리는 아직 확신할 수 없으며, 맥도날드가 알고 있는지 의심스럽지만 맥도날드는 그것을 사용하고 있습니다! 맥도날드는 봉지에 치킨 맥너겟McNuggets을 6개, 9개, 20개씩 넣어서 판매합니다. 따라서 정확히 15개의 맥너겟을 사는 것이 가능합니다(6개가 들어 있는 봉지 하나와 9개가 들어 있는 봉지 하나). 하지만 정확히 16개의 맥너겟을 사는 것은 불가능합니다. 그 이유는 6, 9, 20를 음이 아닌 정수만큼의 조합으로 더했을 때 16이 나올 수 없기 때문입니다. 정확히 n개의 맥너겟을 사는 것이 가능한지 결정하기 위해 디오판틴 방정식을 풀어야 합니다. 즉,

$$6a + 9b + 20c = n$$

을 만족하는 a, b, c에 대한 음이 아닌 정수해(整數解)integer solution를 찾아야 합니다.

문제 1

디오판틴 방정식의 해를 찾아서 50, 51, 52, 53, 54, 55개의 맥너겟을 사는 것이 가능한지 증명하세요. 이 문제를 종이와 연필로 풀어도 되고 프로그램을 만들어서 풀어도 됩니다. 하지만 각각의 양만큼 맥너겟을 사기 위해 6, 9, 20만큼의 맥너겟이 들어 있는 봉지의 조합을 나열하세요. 6, 9, 20의 조합으로 50, 51, 52, 53, 54, 55만큼의 맥너겟을 사는 것이 가능하다면, 56, 57, …, 65만큼의 맥너겟을 사는 것이 가능한지 증명하세요. 다시 말해서, 50~55개의 경우 풀었던 방법을 통해 어떻게 56~65개의 경우일 때의 해를 이끌어낼 수 있는지 보이세요.

정리(Theorem): 어떤 수 x에 대해 $x, x + 1, \cdots, `x + 5'$ 만큼의 맥너겟을 사는 것이 가능하다면, 보다 같거나 큰 개수 만큼의 맥너겟을 사는 것이 가능합니다. 이때, 맥너겟은 6, 9, 20개씩 들어 있습니다.

문제 2

왜 이 정리가 참인지 글로 설명하세요.

문제 1과 2에 대한 답을 `ps2.txt`에 저장하세요.

디오판틴 방정식 풀기

이 정리를 이용해 우리는 구매할 수 없는 가장 큰 맥너겟 수를 찾기 위해 전수 조사를 사용할 수 있습니다.

1부터 시작해서 정확한 수량으로 살 수 없는 맥너겟 개수를 하나의 인스턴스로 가정하고 시작하세요.

n이라고 정의한 각각의 인스턴스에 대해서,

$$6a + 9b + 20c = n$$

을 만족하는 음이 아닌 정수 a, b, c가 존재하는지 테스트하세요(a, b, c의 가능한 모든 조합을 살펴보면서 테스트할 수 있습니다).

존재하지 않는다면 n만큼은 구매할 수 없으므로 n의 값을 저장하세요.

여러분이 n의 값에 대해서 정확한 해를 갖는지에 대한 테스트를 통과하는 연속되는 여섯 개 숫자들을 찾았다면, (해를 찾은 마지막 n 값이 아니라) 저장한 마지막 해가 정답이 됩니다. 왜냐하면 정리Theorem에 의하면 그보다 더 큰 모든 수에 대해서는 모두 정확한 수량으로 살 수 있기 때문입니다.

문제 3

정확한 수량으로 구매할 수 없는 맥너겟의 최대 수를 찾는 반복 프로그램을 만드세요. 프로그램은 다음과 같은 양식으로 답을 출력해야 합니다(정답이 〈n〉 대신에 출력되어야 합니다).

```
"Largest number of McNuggets that cannot be bought in exact
quantity: <n>"
```

힌트 : 여러분의 프로그램은 위의 개요를 따라야 합니다.

힌트 : 정확하게 n개의 맥너겟을 살 수 있는 가능한 방법을 찾는 루프문을 돌리면서 어떤 정보를 기록해야 하는지 생각해보세요. 이를 통해 여러분이 어떤 상태 변수를 사용해야 하는지 결정하는데 도움을 줄 것입니다.

문제 3에 대한 코드를 ps2a.py에 저장하세요.

우리는 이 생각을 일반화하여 6, 9, 20개의 맥너겟이 들어 있는 봉지가 아닌 다른 양만큼의 맥너겟이 들어 있는 봉지에 대해서도 생각해 볼 수 있습니다. 간단하게 맥도날드가 서로 다른 양의 맥너겟이 들어 있는 봉지 3개를 판매한다고 가정할 것입니다.

문제 4

변수 packages는 길이가 3인 튜플tuple[1]이고, 그 튜플의 값들은 각 봉지에 들어 있는 맥너겟의 개수이며, 오름차순으로 정렬되어 있다고 가정합니다. 전수조사를 사용하여 정확하게 구매할 수 없는 맥너겟의 가장 큰 수(200보다 작은 수)를 찾는 프로그램을 만드세요. 개수를 200보다 작은 수로 제한(임의로 정한 것입니다)하는 이유는 어떤 경우에는 살 수 없는 가장 큰 수가 존재하지 않기 때문에 영원토록 찾고 있을 수 있기 때문입니다. ps2b_template.py를 사용해 여러분의 코드를 구조화시키세요. 다음과 같은 양식으로 결과를 출력하세요.

```
"Given package sizes <x>, <y>, and <z>, the largest number of McNuggets
that cannot be bought in exact quantity is: <n>"
```

packages의 값을 바꿔가면서 여러 가지 경우에 대해 프로그램을 시험해 보세요. (6, 9, 20)일 경우와 여러분이 정한 다른 경우 모두 시험해 보세요.

문제 4에 대한 코드를 ps2b.py에 저장하세요.

제출 과정

저장. 작성하는 문제는 ps2.txt, 코드 문제는 ps2a.py, ps2b.py 안에 작성하세요. 파일의 이름을 바꾸지 마세요.

1. 튜플(tuple) 자료형에 대해서 5장을 참고하세요.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 파일 이름과 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적어주세요. 또한 공동 작업을 한 학우(들)이 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# ps2a.py
# Problem Set 2
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

```
# ps2a.py
###
### template of code for Problem 4 of Problem Set 2, Fall 2008
###

bestSoFar = 0    # variable that keeps track of largest number
             # of McNuggets that cannot be bought in exact quantity
packages = (6,9,20)   # variable that contains package sizes

for n in range(1, 200):   # only search for solutions up to size 200
    ## complete code here to find largest size that cannot be bought
    ## when done, your answer should be bound to bestSoFar
```

문제 세트 3
: 문자열 찾기

서문

이 문제 세트에서는 함수function와 재귀recursion를 사용하는 방법과 더불어 파이썬에서의 문자열 연산을 소개합니다.

- **작업량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상한 것보다 너무 오랜 시간이 걸리는 문제들로 인하여 여러분에게 과도한 학업 부담을 주지 않도록 주의하려고 합니다. .

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 학생은 각각 개별적으로 답안을 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

문자열 그리고 문자열 찾기

강의에서 이미 보았듯이, 문자열은 많은 프로그래밍 언어에 있는 공통적인 자료형이며 문자 정보를 나타내는데 사용합니다. 여러분은 이미 앞에서 문자열 찾기 같은 예제를 경험하였습니다. 예를 들어, 문서에서 단어 혹은 구절을 찾는다는 의미에는 일련의 문자 시퀀스(예를 들어, 문서) 인스턴스에서 다른 문자 시퀀스의 인스턴스들을(찾고자 하는 단어나 문자열)를 찾는다는 개념이 포함되어 있습니다. 같은 맥락으로 구글과 같은 웹 검색은 페이지 순위page rank를 결정하기 위해 문서에 있는 키워드 인스턴스들의 개수를 세어야 합니다.

문자열을 비교하기: 생물학적 관점

문자열 비교는 생물학과 같은 조금 의외적인 환경에서 매우 가치있게 활용되기도 합니다. 현대 생물학의 흔한 문제는 DNA 분자 구조와 그 기능을 결정짓는 특정 구조의 역할을 이해하는데 있습니다. DNA시퀀스는 4가지 뉴클레오티드nucleotide, 즉 아데닌(A), 시토신(C), 구아닌(G), 티민(T)들로 구성되어 있습니다. 하나의 DNA 분자 혹은 DNA 가닥strand은 오직 4가지 뉴클레오티드들의 알파벳으로 구성된 문자열로 나타낼 수 있습니다. 예를 들어 문자열 AAACAACTTCGTAAGTATA은 특정한 DNA 가닥을 나타냅니다.

특정한 DNA 가닥의 기능을 이해하기 위한 한 가지 방법으로 이미 알려진 DNA 시퀀스 라이브러리와 비교해 볼 수 있습니다. 이미 알려졌다는 것은 그 DNA 시퀀스의 기능과 구조가 알려져 있다는 의미이며, 비슷한 구조가 비슷한 기능을 한다는 아이디어를 사용합니다. 박테리아와 같이 간단한 유기체일지라도 DNA 시퀀스에 뉴클레오티드가 수 백만 개가 있으며, 인간의 염색체는 대략적으로 2억 4천 6백만개가 있기 때문에, DNA 가닥을 비교하는 방법이 매우 효율적이어야 그것을 유용하게 사용할 수 있습니다.

이 문제 세트에서 실제적으로 유용하게 쓸 만한 도구를 만들어보라고 요구하는 것은 아닙니다. 단지 몇몇 간단한 비교하는 방법들을 공부해보며 이와 관련된 문제에 대한 감각을 길렀으면 합니다.

파이썬에 내장된 몇몇 함수를 사용해서 시작해보도록 하겠습니다.

파이썬 내장 함수들을 사용하기 위해 파일의 첫 부분에 string 라이브러리를 import합니다. 타겟 문자열 target에서 키워드 문자열 key가 처음 일치하는 시작점을 알고자 한다면 find 함수를 사용하면 됩니다.

```
from string import *
```

`"atgacatgcacaagtatgcat".find("atgc")`와 같이 find를 이용해 연습을 해보세요.[1]

`find` 함수는 target에서 처음으로 찾은 key의 인덱스를 반환한다는 점을 기억하세요. 또한 target에서 key 인트턴스를 찾을 수 없다면 -1을 반환하는 사실을 기억하세요.

`"atgacatgcacaagtatgcat".find("ggcc")`을 실행해보세요.

1. [역자 주] 파이썬 프롬프트(>>>)에서 연습할 수 있습니다.

이제 우리는 조금씩 더 복잡한 문제를 통해 문자열을 찾는 아이디어를 탐색해 보겠습니다.

[주의] DNA 문자열 예제로부터 코드를 작성할 것입니다. 하지만, 여러분이 작성할 코드가 단지 DNA 문자열에만 적용될 거라고 가정하지 마세요. 예를 들어 문자열이 오직 서로 다른 주4개의 문자로 구성되었다고 가정하지 말고, 문자열이 다양한 종류의 문자를 포함한다고 생각하세요.

간단한 문제부터 시작하죠. key 문자열이 target 문자열에 몇 번 등장하는지 세어보기 원한다고 가정합시다. 이 문제를 해결하기 위해 두 함수를 만들겠습니다. 하나는 반복적으로 또 하나는 재귀적으로 구현합시다. 각 함수에 대해 파이썬의 find 함수를 사용해도 됩니다. find 함수를 사용해 문자열에서 처음으로 일치하는 부분을 찾는 것이 아니라, 선택적 인자들optional arguments을 어떻게 구성하는지에 대해 find 함수의 사양을 찾아보아야 합니다.[2] 예를 들어

```
"atgacatgcacaagtatgcat".find("atgc")
```

는 값 5를 반환하며,

```
"atgacatgcacaagtatgcat".find("atgc", 6)
```

는 값 15를 반환합니다. 6번째 인덱스부터 시작한다면, 다음으로 일치하는 부분은 15번째라는 것을 의미합니다.

재귀 함수로 구현할 때, 동일한 문제지만 더 작은 버전(더 작은 target 문자열)에서 여러분이 만든 함수를 사용하는 방법에 대해 생각해야 할 것입니다. 예를 들어 target 문자열에서 key 문자열과 일치하는 첫 인스턴스를 찾았다고 할 때의 결과 값과, 더 작은 target 문자열에 같은 함수를 적용한 것의 결과 값을 어떻게 결합할지 생각해보세요. 문자열의 부분문자열substring을 찾을 때에 문자열을 나누는slicing 연산을 유용하게 사용할 수 있을 겁니다.

문제 1

두 개의 매개 변수, 즉 key 문자열과 target 문자열을 받는 아래와 같은 두 함수를 만드세요. 이 함수들은 반복적으로 그리고 재귀적으로 target 문자열에 있는 key 문자열의 인스턴스 수를 계산합니다. 다음 함수들의 정의를 완성하세요.

2. [역자 주] 파이썬 프롬프트(>>>)에 help(find)를 입력하면 사양을 찾아볼 수 있습니다.

```
def countSubStringMatch(target, key):
```

와

```
def countSubStringMatchRecursive (target, key):
```

문제 1에 대한 코드를 ps3a.py에 저장하세요.

이제 다음은 부분문자열이 일치하는 것을 찾는 아이디어들에 대해 살펴볼 것입니다. 이 문제들은 반복적 혹은 재귀적으로 해결할 수 있습니다. 선형 구조[3]에서 일치하는 것을 찾는 경우에는 반복적으로 접근하는 방법이 더 직관적일지 모르지만, 둘 중에 어느 한 접근 방법을 사용해도 됩니다.

다음으로 find를 일반화한 함수를 만드세요. 이 함수는 첫 번째 인스턴스뿐만이 아니라, target 문자열에서 일치하는 모든 부분들의 시작점들을 튜플tuple로 반환해야 합니다.

문제 2

subStringExact 함수를 만드세요. 이 함수는 두 개의 매개변수를 받습니다. 하나는 target 문자열이고 다른 하나는 key 문자열입니다. 첫 번째 인덱스가 0부터 시작한다고 하고, 일치하는 모든 부분들의 시작점들을 튜플로 해서 반환합니다. 다음 함수의 정의를 완성하세요.

```
def subStringMatchExact(target,key):
```

예를 들어

```
subStringMatchExact("atgacatgcacaagtatgcat","atgc")
```

는 튜플 (5, 15)를 반환해야 합니다. 파일 ps3_template.py는 만든 함수를 확인하기 위한 몇 가지 테스트 문자열이 있습니다. 특히 우리는 두 가지 target 문자열을 제공합니다.

```
1  target1 = 'atgacatgcacaagtatgcat'
2  target2 = 'atgaatgcatggatgtaaatgcag'
```

그리고 네 가지 key 문자열을 제공합니다.

```
1  key10 = 'a'
2  key11 = 'atg'
3  key12 = 'atgc'
4  key13 = 'atgca'
```

3. [역자 주] 비선형 구조로 트리 혹은 그래프를 생각해볼 수 있다.

key 문자열과 target 문자열의 여러 조합으로 함수를 확인해보세요. 스스로도 몇 가지 예제를 만들어 보세요. 문제 2에 대한 코드를 ps3b.py에 저장하세요.

문제 2에서 작성한 함수는 target 문자열에서 key 문자열이 정확하게 일치하는 시작점들을 찾습니다. 그런데, 근접하게 일치한 경우를 찾는 것이 종종 유용합니다. 예를 들어 target 문자열에서 key 문자열과 일치하는 것을 찾을 때, key 문자열에서 한 가지 요소가 다른 요소로 대체된 경우입니다. 예를 들어 ATGACATGCACAAGTATGCAT에서 ATGC를 찾는다고 합시다. 5번째에서 정확하게 일치하고, 다음으로는 15번째에서 정확하게 일치합니다. 하지만, 0번째에는 key에서 C를 A로 대체한다면, target과 ATGC가 일치하게 됩니다. 마찬가지로 key ATTA에서 두 번째 나오는 T를 G로 대체한다면 0번째에서 일치하게 됩니다.

이 문제를 해결하기 위해 문제 2에서 만든 함수에 살을 덧붙이면 됩니다. 특히 다음 단계들을 고려해보세요. 우선, key를 두 부분으로 나누세요(한 부분은 비어있는 문자열이 되겠군요). 이것들을 각각 key1, key2라고 부릅시다. 각 부분들에 대해, 문제 2에서 만든 함수를 사용해서 다음과 같이 함수를 불러서 일치하는 시작점들을 찾으세요.

```
1  starts1 = subStringMatchExact(target, key1)
2  starts2 = subStringMatchExact(target, key2)
```

위의 두 함수의 실행 결과는 두 개의 튜플이며, 각각은 target에서 일치하는 key 문자열의 두 부분(key1, key2)의 시작점을 의미합니다. 예를 들어 만약 우리가 key ATGC를 고려한다면, target에 대해 A 와 GC를 매칭하는 것을 고려해 볼 수 있겠네요. ATGACATGCA 안에서 A와 일치하는 부분은 튜플 (0, 3, 5, 9)이며, GC와 일치하는 부분은 튜플 (7,)입니다. 물론, 모든 경우의 부분문자열에 대해 검색을 해야겠죠. 빈 문자열과 TGC, A와 GC, AT와 C, ATG와 빈 문자열입니다. 문제 2에서 만든 답안으로 이 값들을 찾을 수 있다는 사실에 주목하세요.

두 개의 부분문자열에 일치하는 시작점들을 얻었다면, 이 첫 번째 부분문자열과 두 번째 부분문자열의 일치하는 점들 중에 어떤 조합이 우리가 찾는 조합인지를 결정해야 합니다. 이것은 쉽게 확인할 수 있습니다. 첫 번째 부분문자열과 일치하는 부분의 시작점 인덱스를 (start1의) n이라고 합시다. 첫 번째 부분문자열의 길이를 m이라고 합시다. 두 번째 부분문자열과 일치하는 부분의 시작점 인덱스를 (starts2의) k라고 합시다. 그렇다면 유효한 시작점의 인덱스는 를 만족하는 n입니다. 이것은 두 번째 부분문자열과 일치하는 부분의 시작점 인덱스와 첫 번째 부분문자열과 일치하는 부분의 시작점 인덱스가 하나 차이이기 때문이겠죠.

문제 3

세 개의 매개변수를 받는 constrainedMatchPair 함수를 만드세요. 첫 번째 부분문자열의 시작점들을 의미하는 튜플, 두 번째 부분문자열의 시작점들을 의미하는 튜플, 그리고 첫 번째 부분문자열의 길이를 매개변수로 받습니다. 앞에서 설명한 를 만족하는 n값들의 튜플을 반환합니다. 다음의 정의를 완성하세요.

```
def constrainedMatchPair(firstMatch,secondMatch,length):
```

이 함수를 테스트하기 위해 subStringMatchOneSub 함수를 제공했습니다. 이 함수는 두 개의 매개변수 즉 target문자열과 key문자열 받습니다. 이 함수는 완전히 일치하거나, 1개의 요소(뉴클레오티드)만 제외하고 모두 일치하는 경우의 시작점들로 구성된 튜플을 반환합니다. ps3_template.py에 구현되어 있으며, 이 함수 내부에서 여러분이 작성해야 하는 함수를 호출합니다.

문제 3에 대한 코드를 ps3c.py에 저장하세요.

문제 3에서 작성한 함수는 key가 target에 1개의 요소만 제외하고 일치하는 경우와 더불어 완전히 일치하는 경우도 함께 찾아줍니다. key가 target에 1개의 요소만 제외하고 일치하는 경우만 찾고 싶다고 가정합시다. 이것을 쉽게 해결하기 위해 문제 2와 문제 3에서 구현한 함수를 사용하면 됩니다. 이 함수들은 각각 완전히 일치하는 경우의 시작점들의 튜플과 1개의 요소만 제외하고 일치하는 경우와 더불어 완전히 일치하는 경우의 시작점들의 튜플을 반환합니다. 두 번째 튜플에서 첫 번째 튜플의 요소들을 제외한다면, 1개의 요소만 제외하고 일치하는 경우만 찾을 수 있겠죠.

문제 4

두 개의 매개변수, 즉 target 문자열과 key 문자열을 받는 subStringMatchExactly-OneSub 함수를 만드세요. 이 함수는 key가 target에서 1개의 요소만 제외하고 일치하는 시작점들의 튜플을 반환해야 합니다. 다음 정의를 완성하세요.

```
def subStringMatchExactlyOneSub(target,key):
```

문제 4에 대한 코드를 ps3d.py에 저장하세요.

제출 과정

저장. 각 문제마다 지정한 파일 안에 코드를 작성하세요. 파일의 이름을 바꾸지 마세요.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 파일 이름과 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적어주세요. 또한 공동 작업을 한 학우(들)이 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# ps3a.py
# Problem Set 3
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

```
# ps3_template.py
#
# this is a code file that you can use as a template for submitting your
# solutions

# these are some example strings for use in testing your code

#  target strings
target1 = 'atgacatgcacaagtatgcat'
target2 = 'atgaatgcatggatgtaaatgcag'
# key strings
key10 = 'a'
key11 = 'atg'
key12 = 'atgc'
key13 = 'atgca'

### the following procedure you will use in Problem 3
def subStringMatchOneSub(key,target):
    """search for all locations of key in target, with one substitution"""
    allAnswers = ()
    for miss in range(0,len(key)):
        # miss picks location for missing element
        # key1 and key2 are substrings to match
        key1 = key[:miss]
        key2 = key[miss+1:]
        print('breaking key',key,'into',key1,key2)
        # match1 and match2 are tuples of locations of start of matches
        # for each substring in target
        match1 = subStringMatchExact(target,key1)
        match2 = subStringMatchExact(target,key2)
        # when we get here, we have two tuples of start points
        # need to filter pairs to decide which are correct
        filtered = constrainedMatchPair(match1,match2,len(key1))
        allAnswers = allAnswers + filtered
        print('match1',match1)
        print('match2',match2)
        print('possible matches for',key1,key2,'start at',filtered)
     return allAnswers
```

문제 세트 4
: 미래를 준비하기

서문

이 문제 세트에서는 연속 근사법successive approximation의 사용과 더불어 튜플과 리스트와 같은 데이터 구조를 소개합니다.

- **작업량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상한 것보다 너무 오랜 시간이 걸리는 문제들로 인하여 여러분에게 과도한 학업 부담을 주지 않도록 주의하려고 합니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 학생은 각각 개별적으로 답안을 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

미래를 준비하기

최근 주식 시장에서의 사건들은 당신과 조금 멀게 느껴질 수 있습니다. 하지만, 주식 시장의 불안은 곧 미래를 준비하는데 불확실성이 있다는 것을 분명히 보여줍니다. 내년 혹은 수년 안에 퇴직을 생각하고 있다면 미래를 준비해야 합니다. 왜냐면, 401K[1] 계좌의 가치가 현저하게 떨어지고 있기 때문이죠. 여러분은 은퇴할 때까지 아직 너무나 많은 시간이 남아 있다고 생각할지도 모르겠지만, 그래도 우리는 펀드를 저축해가는 아이디어들을 간단히 탐구해볼 것입니다. 그 과

1. [역자 주] 401K는 미국에서 직장인들에게 가장 보편적으로 이용되고 있는 은퇴연금플랜입니다. 직원이 연금을 불입하면 회사에서도 직원의 불입액의 일정 비율로 연금을 불입해주는 복지 수단입니다. 또한 축적된 연금을 뮤추얼펀드에 투자할 수 있습니다.

정 중에 우리는 연속 근사법의 사용과 더불어 리스트와 같은 간단한 데이터 구조를 사용할 것입니다. 우선 퇴직을 대비해서 돈을 저축하기 위한 간단한 모델을 만드는 것으로 시작하겠습니다. 많은 고용주들이 당신 급여의 5%가 되는 금액을 퇴직연금에 넣어 줄 것입니다. 그리고 당신이 연금 불입금을 추가하는 만큼 같은 액수(1달러당 1달러)를 추가로 5%까지 퇴직연금에 더해 줄 것입니다. 이렇게 하여 당신은 급여의 최대 15%까지의 금액을 퇴직연금에 안전하게 투자하게 되는 것이죠(그러므로, 사실상 당신은 고용주로부터 10% 보너스를 받는 것입니다).

퇴직연금을 몇 가지 간단한 수식으로 모델링할 수 있습니다. 당신의 시작 급여를 salary라고 가정하죠. 퇴직연금에 넣는 급여의 백분율을 save라고 합시다. 퇴직연금의 연간 수익률은 growthRate입니다. 그렇다면 당신의 퇴직연금은 리스트 F로 표현되며 다음과 같이 증가합니다.

	퇴직연금
1차년도 말	F[0] = salary * save * 0.01
2차년도 말	F[1] = F[0] * (1 + 0.01 * growthRate) + salary * save * 0.01
3차년도 말	F[2] = F[1] * (1 + 0.01 * growthRate) + salary * save * 0.01

이 과정은 매년 계속되며, 새로 추가하는 납입금과 원금이 증가함에 따라 여러분의 퇴직연금은 증가합니다. 이 문제 세트에서의 수익률은 항상 양수입니다(아쉽게도 현실에서는 그렇지 않죠!).

문제 1

4개의 매개 변수를 받는 함수 `nestEggFixed`를 만드세요. 매개 변수로는 salary, save(기여율: 급여에 대한 연금납입액의 비율), growthRate(연금계좌의 연 수익률: 퍼센트), years(재직 기간)입니다. 이 함수는 리스트 반환해야 하며, 그 값들은 매년 말 퇴직연금의 금액입니다. 최근 금액일수록 리스트의 뒤에 오게끔 만드세요.

다음 함수를 완전히 구현하세요.

```
def nestEggFixed (salary, save, growthRate, years):
```

ps4.py 템플릿의 적당한 곳에 코드를 채워 완성하세요. 함수를 테스트하기 위해 테스트 함수인 `testNestEggFixed()`의 시험케이스를 실행하세요. 또한, 시험케이스들을 여러 개 추가하여 여러분의 코드를 테스트해야 합니다.

첫 번째 모델은 꽤 간단합니다. 한 가지 분명한 것은 주식 시장이 결코 일정한 비율로 증가하지 않는다는 것입니다. 따라서 매년 수익률의 변화를 반영하기 위해 더 나은 모델이 필요합니다.

문제 2

3개의 매개 변수를 받는 함수 `nestEggVariable`을 만드세요. 매개 변수로 salary, save(기여율: 급여에 대한 연금납입액의 비율), growthRates(연금계좌의 연 수익률을 담고 있는 리스트)를 받습니다.

`growthRates`의 길이는 당신이 앞으로 직장에 몇 년간 다니고 있을지를 의미합니다. `growthRates[0]`은 첫 번째 해의 수익률이며, `growthRates[1]`은 두 번째 해의 수익률이고 그 후도 이와 같습니다. 퇴직연금의 시작 금액이 0이기 때문에 `growthRates[0]`은 사실 아무 영향이 없다는 사실에도 주목하세요. 이 함수는 매년 말 퇴직연금의 총액을 담은 리스트를 반환해야 합니다.

다음 함수를 완전히 구현하세요.

```
def nestEggVariable(salary, save, growthRates):
```

ps4.py 템플릿의 적당한 곳에 코드를 채워 완성하세요. 함수를 테스트하기 위해 테스트 함수인 `testNestEggVariable()`의 시험 케이스를 실행하세요. 또한, 시험케이스들을 여러 개 추가하여 여러분의 코드를 테스트해야 합니다.

물론 퇴직 후에 여러분은 생활자금으로 얼마 간의 금액을 매년 인출해서 사용하길 원할 것입니다. 이전의 코드를 사용하면 퇴직하는 시점에서 총 퇴직연금이 얼마인지 계산할 수 있습니다. 이제 퇴직 후에 매년 어느 정도의 금액을 인출해서 사용할 수 있는지 모델링 해봅시다.

가령 퇴직 후에 퇴직연금이 매년 연금 수익률의 리스트에 의거해서[growthRates] 계속해서 증가한다고 가정합시다. 반면에 당신이 매년 인출하는 금액은 일정하며(인플레이션이 0퍼센트라면 좋지 않을까요?) 이를 expenses라고 부릅시다. 퇴직 후에 당신의 퇴직연금이 어떻게 변할지 확인하기 위해 다음 도표를 사용하겠습니다. F는 퇴직하는 당시에 퇴직연금의 총 금액을 의미하며 expenses는 생활자금으로 첫 해에 인출하는 급여를 의미합니다.

	퇴직연금
1번째 연도 말	F[0] = savings * (1 + 0.01 * growthRates[0]) - expenses
2번째 연도 말	F[1] = F[0] * (1 + 0.01 * growthRates[1]) - expenses
3번째 연도 말	F[2] = F[1] * (1 + 0.01 * growthRates[2]) - expenses

문제 3

3개의 매개 변수를 받는 함수 `postRetirement`를 만드세요. 매개 변수로 savings(당신의 퇴직연금에 들어 있는 금액의 초기값), growthRates(퇴직 이후에 퇴직연금의 수익률), expenses(연 급여)를 받습니다. savings이라는 퇴직연금은 연 급여 expenses를 제외하기 전에 증가된다는 사실에 주목하세요.

(위의 표에서 그렇게 계산한 것입니다) 함수는 퇴직 후, 매년 말 퇴직연금의 총액을 담는 리스트를 반환해야 합니다. 문제 2와 같이 `growthRates`의 길이는 당신이 퇴직한 상태로 있을 햇수를 의미합니다.

만약 퇴직연금의 잔고가 부족negative할지라도 급여expenses는 계속해서 일어나야 하며, 수익률은 빚에 대한 이자율을 의미하게 됩니다(위의 표에 있는 공식은 계속 적용된다는 것을 의미합니다).

다음 정의를 완성하세요.

```
def postRetirement(savings, growthRates, expenses):
```

ps4.py 템플릿의 적당한 곳에 코드를 넣어 완성하세요. 함수를 테스트하기 위해 테스트 함수인 `testPostRetirement()`의 시험케이스를 실행하세요. 또한, 시험 케이스들을 여러 개 추가하여 여러분의 코드를 테스트해야 합니다.

가령 여러분이 살아있는 동안에 퇴직연금을 모두 사용하겠다는 방식으로 연 급여에 대한 예산을 세운다고 가정합시다(후손에게 유산을 물려주지 않는 앤드류 카네기Andrew Carnegie의 모델을 따른다고 볼 수 있죠). 연 급여의 예산을 어떻게 해야 결정할 수 있는 방법들 중 하나는 `postretirement()`을 호출할 때에 다양한 연 급여expenses들을 넘겨 주는 방법을 시도해보는 것입니다. 강의에서 소개한 연속 근사법을 사용해서 이 방법을 자동화할 수 있습니다.

이전 강의에서 우리가 이분 검색을 공부한 적이 있습니다. 그때 사용한 예제에서, 우리는 한 숫자의 제곱근을 찾으려고 할 때에, 찾는 값의 위 아래 경계가 되는 high, low 값을 선택하였습니다(예를 들어 정답이 low와 high 사이에 있습니다). 후에 high와 low 값의 평균을 구하고 그 값이 정답에 근접한지 확인합니다(epsilon의 절대값 안에 있는지 말이죠). 만약 값이 충분히 정답에 근접하다면 멈춥니다. 그렇지 않다면, 테스트의 결과에 의거해서 값의 범위를 바꿔가며 확인합니다. 값이 average보다 크다면, low값을 average에 바인딩하고, 반대의 경우에는 high값을 average에 바인딩 하겠죠. 여기서도 이와 같은 방법을 사용하면 됩니다.

문제 4

5개의 매개 변수를 받는 함수 `findMaxExpenses`를 만드세요. 매개 변수로 salary, save(연봉의 기여율), `preRetireGrowthRates`(일하는 동안의 투자에 대한 연간 수익률을 담는 리스트), `postRetireGrowthRates`(퇴직한 후의 투자에 대한 연간 수익률을 담는 리스트), epsilon을 받습니다. 문제 2와 문제 3에서와 같이 `preRetireGrowthRates`의 길이와 `postRetireGrowthRates`의 길이는 각각 기대하는 재직년수, 퇴직 후 기대하는 은퇴기간을 의미합니다.

이분 검색의 아이디어를 사용하여 퇴직연금에서 인출할 수 있는 급여를 계산하세요. 계산할 때, 퇴직 후 은퇴기간이 끝났을 때, 퇴직연금의 잔고가 epsilon보다 작아지게끔 만드세요(원금 보다 매우 작은 양의 돈을 더 인출할 수 있는 사실에 주목하세요).

퇴직 후부터 인출 가능한 연 급여의 리스트를 만드는 것으로 시작하세요. 범위는 0부터 퇴직할 때의 퇴직연금 총액으로 하면 되겠죠(힌트 #1: 문제 2의 해답을 이용해서 결정할 수 있습니다). 이분 검색의 각 단계마다 현재 예상되는 급여를 출력하세요(힌트 #2: 이분 검색에서 문제 3의 해답을 이용하면 됩니다). 그리고 인출할 급여의 추정 값을 반환하세요(힌트 #3: 답은 0과 퇴직연금의 초기값+epsilon 사이에 있어야 합니다).

다음 함수를 구현하세요.

```
1  def findMaxExpenses(salary, save, preRetireGrowthRates,
2  postRetireGrowthRates, epsilon):
```

ps4.py 템플릿의 적당한 곳에 코드를 작성하세요. 함수를 테스트하기 위해 테스트 함수인 `testFindMaxExpenses()`의 시험케이스를 실행하세요. 또한, 시험 케이스들을 여러 개 추가하여 여러분의 코드를 테스트해야 합니다.

제출 과정

저장. ps1a.py, ps4.py 안에 코드를 작성하세요. 파일의 이름을 바꾸지 마세요.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 파일 이름과 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적어주세요. 또한 공동 작업을 한 학우(들)이 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# ps4.py
# Problem Set 1
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

```
# ps4.py
#
# Problem Set 4
# Name:
# Collaborators:
# Time:

#
# Problem 1
#

def nestEggFixed(salary, save, growthRate, years):
    """
    - salary: the amount of money you make each year.
    - save: the percent of your salary to save in the investment account each
      year (an integer between 0 and 100).
    - growthRate: the annual percent increase in your investment account (an
      integer between 0 and 100).
    - years: the number of years to work.
    - return: a list whose values are the size of your retirement account at
       the end of each year.
    """
    # TODO: Your code here.

def testNestEggFixed():
    salary     = 10000
    save       = 10
    growthRate = 15
    years      = 5
    savingsRecord = nestEggFixed(salary, save, growthRate, years)
    print savingsRecord
    # Output should have values close to:
    # [1000.0, 2150.0, 3472.5, 4993.375, 6742.3812499999995]

    # TODO: Add more test cases here.
#
# Problem 2
#

def nestEggVariable(salary, save, growthRates):
    # TODO: Your code here.
    """
```

```
    - salary: the amount of money you make each year.
    - save: the percent of your salary to save in the investment account each
      year (an integer between 0 and 100).
    - growthRate: a list of the annual percent increases in your investment
      account (integers between 0 and 100).
    - return: a list of your retirement account value at the end of each year.
    """

def testNestEggVariable():
    salary     = 10000
    save       = 10
    growthRates = [3, 4, 5, 0, 3]
    savingsRecord = nestEggVariable(salary, save, growthRates)
    print savingsRecord
    # Output should have values close to:
    # [1000.0, 2040.0, 3142.0, 4142.0, 5266.2600000000002]

    # TODO: Add more test cases here.
#
# Problem 3
#

def postRetirement(savings, growthRates, expenses):
    """
    - savings: the initial amount of money in your savings account.
    - growthRate: a list of the annual percent increases in your investment
      account (an integer between 0 and 100).
    - expenses: the amount of money you plan to spend each year during
      retirement.
    - return: a list of your retirement account value at the end of each year.
    """
    # TODO: Your code here.

def testPostRetirement():
    savings     = 100000
    growthRates = [10, 5, 0, 5, 1]
    expenses    = 30000
    savingsRecord = postRetirement(savings, growthRates, expenses)
    print savingsRecord
    # Output should have values close to:
    # [80000.000000000015, 54000.000000000015, 24000.000000000015,
    # -4799.9999999999854, -34847.999999999985]

    # TODO: Add more test cases here.

#
# Problem 4
#

def findMaxExpenses(salary, save, preRetireGrowthRates, postRetireGrowthRates,
                    epsilon):
```

```
    """
    - salary: the amount of money you make each year.
    - save: the percent of your salary to save in the investment account each
      year (an integer between 0 and 100).
    - preRetireGrowthRates: a list of annual growth percentages on investments
      while you are still working.
    - postRetireGrowthRates: a list of annual growth percentages on investments
      while you are retired.
    - epsilon: an upper bound on the absolute value of the amount remaining in
      the investment fund at the end of retirement.
    """
    # TODO: Your code here.

def testFindMaxExpenses():
    salary                = 10000
    save                  = 10
    preRetireGrowthRates  = [3, 4, 5, 0, 3]
    postRetireGrowthRates = [10, 5, 0, 5, 1]
    epsilon               = .01
    expenses = findMaxExpenses(salary, save, preRetireGrowthRates,
                               postRetireGrowthRates, epsilon)
    print expenses
    # Output should have a value close to:
    # 1229.95548986

    # TODO: Add more test cases here.
```

문제 세트 5
: 단어 게임 I

서문

어린 시절에 우리는 Ghost와 같은 단어 게임을 좋아했습니다. 그러므로, 좋은 부모님이나 선생님들이 항상 그러하신 것처럼, 우리가 어렸을 때 흥미로웠던 것들을 이제는 여러분도 하도록 강요할 것입니다.^^ 여러분은 이 문제 세트에서 두 개의 단어 게임을 구현할 것입니다. 처음에는 우리가 여러분이 6.00 단어 게임을 구현하는 것을 도울 것이며, 그 뒤에 여러분 스스로 Ghost를 구현하게 될 것입니다.

이 문제 세트의 길이에 겁먹지 마세요. 읽을 내용은 매우 많지만, 여러분이라면 충분히 할 수 있습니다.

6.00 단어 게임을 설명하겠습니다. 게임은 Scrabble 혹은 Text Twist와 유사합니다. 이 게임들을 해 본 적이 있다면 말이죠. 플레이어들에게 문자들이 주어집니다. 플레이어들은 문자로 하나 혹은 그 이상의 단어들을 만듭니다. 각각의 유효 단어는 그 단어의 길이와 사용한 문자들에 의거하여 점수를 얻습니다.

게임의 규칙은 다음과 같습니다.

딜링(Dealing)

- 플레이어는 무작위로 n개의 문자를 받습니다(n=7이라고 가정합시다).
- 플레이어는 손 안에 있는 문자들을 조합해서 단어 세트를 만듭니다. 각 문자는 최대 한 번씩 사용 가능합니다.
- 몇몇 문자는 사용 못하고 남게 됩니다(이 문자들은 점수에 포함되지 않습니다).

득점

- hand의 점수는 만든 단어들의 점수의 합입니다.
- word의 점수는 단어를 구성하는 문자들의 점수의 합입니다. 만약 한 번에 모든 문자를

사용했다면 추가로 50점을 받습니다.

- Scrabble에서 점수를 매기는 방법과 동일하게 점수를 합산합니다. A는 1점, B는 3점, C는 3점, D는 2점, E는 1점 등등입니다. 다음과 같은 딕셔너리에 이 정보를 정의해 두었습니다.

  `SCRABBLE_LETTER_VALUES`

  이 딕셔너리는 각각의 소문자에 대하여 Scrabble 문자 값을 매핑합니다.
- 예를 들면, 'weed' 는 8점 (4 + 1 + 1 + 2 = 8)입니다. 물론, hand가 'w' 1개, 'e' 2개, 'd' 1개를 가지고 있다는 가정하에 말이죠.
- 다른 예로, 만약이며, 한 번에 'waybill' 이라는 단어를 만들었다면 65점을 얻게 됩니다. 그 이유는, 점과 또한 주어진 문자 7개를 모두 사용했기 때문에 'bingo' 보너스 50점을 받기 때문입니다.

- **작업량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상한 것보다 너무 오랜 시간이 걸리는 문제들로 인하여 여러분에게 과도한 학업 부담을 주지 않도록 주의하려고 합니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 학생은 각각 개별적으로 답안을 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

시작하기

- **다운로드하고 저장하세요.**[1]
  - ps5.py: 문제 1부터 5까지 여러분이 채워야 할 스켈레톤skeleton(골격, 뼈대) 코드입니다.
  - test_ps5.py: 여러분의 코드에 대한 단위 테스트unit tests입니다(자세한 것은 뒤에 보도록 하죠).
  - ps5_ghost.py: 여러분이 채워야 할 문제 6에 대한 스켈레톤 코드입니다.

1. [역자 주] 참고문헌: 모든 문제 세트에 필요한 파일은 MIT OCW에서 다운로드 받을 수 있습니다. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/

- words.txt: 유효한 단어 목록입니다('North American Scrabble' 에서 인정하는 최대 10개의 문자로 구성된 모든 단어들입니다).

위의 파일을 동일한 경로에 넣어야 합니다!

- **코드를 실행하세요**

모든 것이 제대로 설치되었는지 확인하기 위하여, 아무 수정도 하지 말고 ps5.py를 실행하세요. 우리가 제공한 코드는 파일(words.txt)에서 유효한 단어 목록을 읽어온 후에 `play_game` 함수를 호출합니다. 여러분은 이 함수가 작동하는데 필요한 것들을 구현하면 됩니다.

만약 모든 것이 제대로 되어있다면, 약간의 시간이 지난 후에, 다음이 출력됨을 볼 수 있습니다.

```
Loading word list from file...
   83667 words loaded.
play_game not implemented.
play_hand not implemented.
```

만약 `IOError`(예를 들어 No such file or directory)라는 문구가 나오면, `WORDLIST_FILENAME`의 내용을 바꿔야 할 것입니다. 파일의 위쪽에 정의된 부분을 words.txt가 저장된 곳의 **전체 경로**로 바꾸세요(파일을 어디에 저장했는지에 따라 경로가 바뀔 겁니다).

- **주어진 코드**

파일 ps5.py에는 여러분이 문제를 해결하는 데 있어 필요한 함수들이 여러 개 이미 구현되어 있습니다. 다음 주석 안에 있는 코드는 읽고 이해하는 것이 좋지만, 무시해도 됩니다.

```
# -----------------------------------
# Helper code
# (you don't need to understand this helper code)
. . .
# (end of helper code)
# -----------------------------------
```

- **단위 테스팅**

이 문제 세트에서는 여러 모듈 함수를 작성하고 이것들을 통해 완전한 워드 게임을 만들 수 있도록 구성되어 있습니다. 전체 게임이 준비될 때까지 기다리기보다 다음 부분으로 넘어가기 전

에 해당 함수를 테스트하는 방법을 권장합니다. 이러한 접근을 단위 테스팅이라고 하고 코드를 디버그하는데 도움을 줄 것입니다.

여러분이 시작하는 데 도움을 주는 테스트 함수들을 몇 개 제공했습니다. 문제 세트를 진행하며 test_ps5.py를 실행하세요.

만약 코드가 단위 테스트를 통과한다면 SUCCESS 메시지가 나타날 것이며, 그렇지 않다면 FAILURE 메시지가 나타날 것입니다. 이 테스트들은 전수검사exhaustive test가 아닙니다. 따라서 또 다른 방법으로 여러분이 작성한 코드를 테스트해야 할지도 모릅니다.

주어진 ps5.py 스켈레톤 코드를 사용해서, test_ps5.py를 실행하면, 여러분은 모든 테스트들이 실패하는 것을 보게 될 것입니다.

다음은 제공된 테스트 함수입니다.

```
1  test_get_word_score() - Test the get_word_score() implementation.
2  test_update_hand() - Test the update_hand() implementation.
3  test_is_valid_word() - Test the is_valid_word() implementation.
```

Ⅰ. 단어 점수

첫 단계에서는 한 단어에 대한 점수를 계산하는 코드를 구현합니다.

문제 1

함수 get_word_score는 소문자로 구성된 문자열(word)을 받고, 게임의 점수 규칙에 따라 해당 단어의 점수를 정수로 반환합니다.

ps5.py 안에 get_word_score의 코드를 작성하세요.

```
1  def get_word_score(word, n):
2      """
3      Returns the score for a word. Assumes the word is a
4      valid word.
5
6      The score for a word is the sum of the points for letters
7      in the word, plus 50 points if all n letters are used on
8      the first go.
```

```
9
10      Letters are scored as in Scrabble; A is worth 1, B is
11      worth 3, C is worth 3, D is worth 2, E is worth 1, and so on.
12      word: string (lowercase letters)
13      returns: int >= 0
14      """
15      # TO DO ...
```

입력 받은 word는 항상 소문자로 구성된 문자열 혹은 빈 문자열 ""이라고 가정하세요. ps5.py 윗부분에 정의된 딕셔너리 SCRABBLE_LETTER_VALUES를 사용하세요. 그 값은 바꾸지 마세요.

hand에 있는 문자들이 항상 7개라고 가정하지 마세요! 매개변수 n은 보너스 점수를 위한 문자 수 입니다(hand에 있는 최대 문자 수와 같습니다).

테스팅: 만약 이 함수가 제대로 구현되어 있다면, test_ps5.py를 실행했을 때에 test_get_word_score() 테스트가 통과하는 것을 볼 수 있습니다. 또한 적절한 다른 영어 단어들을 사용해서 get_word_score를 테스트 해보세요.

힌트: 문자열(단어)에 있는 문자를 하나씩 접근하기 위해 다음과 같은 반복문을 사용하세요. word에 있는 문자를 각각 출력하는 반복문입니다.

```
1  for letter in word:
2      print letter
```

II. hand에 딜링하기

• hands를 표현하기

hand는 게임을 하는 동안 플레이어가 가지고 있는 문자 세트입니다. 플레이어는 시작할 때에 무작위로 문자 세트를 받습니다. 예를 들어 플레이어는 다음의 hand로 게임을 시작할 수 있습니다.

```
a, q, l, m, u, i, l
```

파이썬에서 hand를 간단히 표현하기 위해 list를 사용하면 됩니다.

```
hand = ['a', 'q', 'l', 'm', 'u', 'i', 'l']
```

하지만, 우리는 조금 **다른** 방법으로 hand를 표현해 봅시다. 그 이유는 나중에 update_hand와 is_valid_word 함수를 사용할 때, 더 쉽게 구현할 수 있기 때문이죠(일반적으로 이러한 개념들을 코드로 표현하는 다양한 방법들이 있지만, 어떤 특정한 목적에 따라 다른 방법들 보다 더 적합한 것이 있기도 하죠).

우리 프로그램에서 hand는 딕셔너리로 구현할 것입니다. 딕셔너리 키는 소문자 문자를 나타내고, 딕셔너리 값은 특정 문자가 hand 안에 몇 번 있는지를 나타냅니다. 예를 들어 위의 hand는 다음과 같이 표현할 수 있습니다.

```
hand = {'a':1, 'q':1, 'l':2, 'm':1, 'u':1, 'i':1}
```

반복된 문자 'l' 이 어떻게 표현되었는지에 주목하세요.

딕셔너리 표현 방법에서 값에 접근하는 일반적인 방법은 hand['a'] 입니다. 'a'는 우리가 찾고자 하는 키입니다. 하지만, 위 방법은 키가 딕셔너리에 있는 경우만 동작합니다. 키가 없다면, 에러가 나오게 되죠. 이것을 방지하기 위해 우리는 hand.get('a', 0)을 사용하겠습니다. 이것은 키가 딕셔너리에 있다는 사실이 불확실할 때, 딕셔너리 값에 접근하는 안정한 방법입니다. 위 방법을 사용하면, 딕셔너리에 키가 있는 경우 그 값을 반환하며, 키가 없는 경우 0을 반환합니다.

- **단어를 딕셔너리 표현방법으로 전환하기**

ps5.py의 위쪽에 유용하게 사용할 수 있는 함수 get_frequency_dict를 정의했습니다. 입력 값으로 문자열을 받으면, 딕셔너리를 반환합니다. 딕셔너리 키는 문자를 나타내고, 딕셔너리 값은 그 문자가 몇 번 문자열에 나오는 회수를 나타내는 딕셔너리 입니다. 예를 들어

```
1 >> get_frequency_dict("hello")
2 {'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

hands를 표현하기 위해 위와 같은 딕셔너리를 사용합니다.

문제 2

딕셔너리로 표현된 hand를 우리는 사용자 친화적으로 보여줄 것입니다.

display_hand 함수를 구현하세요.

```
def display_hand(hand):
    """
    Displays the letters currently in the hand.

    For example:
      display_hand({'a':1, 'x':2, 'l':3, 'e':1})
    Should print(out something like:)
      a x x l l l e
    The order of the letters is unimportant.

    hand: dictionary (string -> int)
    """
# TO DO ...
```

- **무작위로 hand 만들어내기**

플레이어가 받은 hand는 무작위로 고른 문자 세트로 구성되어 있습니다. 이제 이 무작위 hand를 만들어내는 함수가 필요합니다. hand를 무작위로 고른다는 것에 유의해야 합니다. 또한 플레이어가 단어들을 만들어 내려면 hand에 충분한 모음VOWEL이 있어야 한다는 사실에 유의해야 합니다.

아래 구현을 살펴보면, 우리는 random 모듈에 있는 randrange 함수를 사용해서 무작위로 숫자를 만들어 냅니다. ps5.py 윗부분에 import random을 하고, 문법 `random.randrange()`를 사용해 random 모듈에 있는 randrange를 호출한 것에 주목하세요. 우리가 구현한 `deal_hand`를 유의해서 보고 이것의 역할과 어떻게 동작하는지 이해하세요.

```
def deal_hand(n):
    """
    Returns a random hand containing n lowercase letters.
    At least n/3 the letters in the hand should be VOWELS.

    Hands are represented as dictionaries. The keys are
    letters and the values are the number of times the
    particular letter is repeated in that hand.

    n: int >= 0
    returns: dictionary (string -> int)
    """
    hand={}
```

```
14      num_vowels = n // 3
15
16      for i in range(num_vowels):
17         x = VOWELS[random.randrange(0,len(VOWELS))]
18      hand[x] = hand.get(x, 0) + 1
19
20      for i in range(num_vowels, n):
21         x = CONSONANTS[random.randrange(0,len(CONSONANTS))]
22      hand[x] = hand.get(x, 0) + 1
23
24      return hand
```

`hand.get(x, 0)`를 사용해서 딕셔너리 값에 접근한 것에 주목하세요. 딕셔너리 키 x가 딕셔너리에 있는지 확실하지 않기에 이 방법을 사용하며, 앞의 '**hands를 표현하기**' 단원에서 설명한 바 있습니다.

- **hand에서 문자를 제거하기**

플레이어는 하나의 문자 세트인 하나의 hand로 시작합니다. 플레이어가 문자들을 사용하며 단어를 만들어 내면, 이 세트의 문자들은 없어집니다. 예를 들어 플레이어가 다음 hand로 시작했다고 가정합시다.

```
a, q, l, m, u, i, l
```

플레이어는 단어 quail을 만들어 낼 수 있습니다. 이제 플레이어의 hand에는 다음 문자들이 남아있게 됩니다.

```
l, m
```

이제 hand와 word를 입력으로 받는 함수를 작성하세요. hand에서 단어 word를 만들어 내고, hand에 남아 있는 문자를 반환하세요.

예를 들어

```
1  >> hand = {'a':1, 'q':1, 'l':2, 'm':1, 'u':1, 'i':1}
2  >> display_hand(hand)
3  a q l l m u i
4  >> hand = update_hand(hand, 'quail')
5  >> hand
```

```
{'l': 1, 'm': 1}
>> display_hand(hand)
l m
```

[주의] 위의 예제에서, 다른 방법으로 update_hand를 호출한 뒤에, hand의 값은 딕셔너리{'a':0, 'q':0, 'l':1, 'm':1, 'u':0, 'i':0}를 반환할 수도 있습니다. 정확한 값은 여러분이 어떻게 구현하는지에 따라 달라집니다. 하지만 display_hand()의 출력은 어떠한 경우든지 동일합니다.

문제 3

update_hand 함수를 구현하세요. 이 함수가 부작용이 없도록 주의하세요. 예를 들어 입력 받은 hand를 바꿔서는 안됩니다.

```
def update_hand(hand, word):
    """
    Assumes that 'hand' has all the letters in word.
    In other words, this assumes that however many times
    a letter appears in 'word', 'hand' has at least as
    many of that letter in it.

    Updates the hand: uses up the letters in the given word
    and returns the new hand, without those letters in it.

    Has no side effects: does not mutate hand.

    word: string
    hand: dictionary (string -> int)
    returns: dictionary (string -> int)
    """
    # TO DO ...
```

테스팅: test_update_hand() 테스트가 통과하는지 확인하세요. 또 다른 적절한 입력 값들로 update_hand를 테스트해도 좋습니다.

Ⅲ. 유효 단어

지금까지 우리는 무작위로 hand를 만들고, 그 hand를 사용자에게 보여주었습니다. 또한 우리는 사용자에게 하나의 단어를 입력 받아(파이썬의 input 혹은 raw_input) 그 단어의 점수(get_

word_score를 사용해서)를 매길 수 있습니다. 그러나 아직 우리는 플레이어가 입력한 단어가 게임의 규칙에 맞는지 확인하는 코드를 작성하지 않았습니다.

유효한 단어는 word_list에 있어야 하며, 현재 hand에 있는 문자들을 조합해서 만들 수 있어야 합니다.

문제 4

is_valid_word 함수를 구현하세요.

```
def is_valid_word(word, hand, word_list):
    """
    Returns True if word is in the word_list and is entirely
    composed of letters in the hand. Otherwise, returns False.
    Does not mutate hand or word_list.
    word: string
    hand: dictionary (string -> int)
    word_list: list of lowercase strings
    """
    # TO DO ...
```

테스팅: test_is_valid_word가 통과하는지 확인하세요. 특히 동일한 hand에 여러 번 함수를 호출하며 확인해봐야 할 것입니다.

Ⅳ. hand를 플레이하기

우리는 이제 플레이어와 상호작용하는 코드를 작성하기를 시작할 준비가 되었습니다.

문제 5

함수 play_hand를 구현하세요. 이 함수는 사용자가 hand 하나로 게임하는 것을 허락합니다.

```
def play_hand(hand, word_list):
    """
    Allows the user to play the given hand, as follows:
    * The hand is displayed.
    * The user may input a word.
    * An invalid word is rejected, and a message is displayed asking
```

```
    the user to choose another word.
    * When a valid word is entered, it uses up letters from the hand.
    * After every valid word: the score for that word and the total
      score so far are displayed, the remaining letters in the hand
      are displayed, and the user is asked to input another word.
    * The sum of the word scores is displayed when the hand finishes.
    * The hand finishes when there are no more unused letters.
      The user can also finish playing the hand by inputing a single
      period (the string '.') instead of a word.
    * The final score is displayed.
hand: dictionary (string -> int)
word_list: list of lowercase strings
    """
    # TO DO ...
```

테스팅: 게임을 하고 있다고 가정하고 여러분의 구현한 것을 한 번 테스트 해보세요.

[주의] hand에 문자가 항상 7개만 있다고 가정하지 마세요! 전역 변수 HAND_SIZE는 이 값을 의미합니다. 여기서, play_hand의 출력물 예제를 제공합니다(어떤 메시지를 출력하는 지에 따라 결과값은 상이할 것입니다).

```
Ex1:
   Current Hand:  a c i h m m z
   Enter word, or a . to indicate that you are finished: him
   him earned 8 points. Total: 8 points
   Current Hand:  a c m z
   Enter word, or a . to indicate that you are finished: cam
   cam earned 7 points. Total: 15 points
   Current Hand:  z
   Enter word, or a . to indicate that you are finished: .
   Total score: 15 points.
Ex2:
   Current Hand:  a s t t w f o
   Enter word, or a . to indicate that you are finished: tow
   tow earned 6 points. Total: 6 points
   Current Hand:  a s t f
   Enter word, or a . to indicate that you are finished: tasf
   Invalid word, please try again.
   Current Hand:  a s t f
   Enter word, or a . to indicate that you are finished: fast
   fast earned 7 points. Total: 13 points
   Total score: 13 points.
```

V. 게임하기

게임은 여러 hands를 가지고 하는 것으로 이루어져 있습니다. 워드 게임 프로그램을 완성하기

위해 마지막으로 함수 하나를 더 구현하면 됩니다.

문제 6

play_game 함수의 코드에 주석 처리된 부분을 실행할 수 있도록 하세요. play_game 안에 현재 주석 처리된 부분을 지우면 됩니다. 충분히 읽어보고 코드가 하는 역할과 어떻게 동작하는지 이해하세요.

이번 문제에서는 코딩이 필요하지 않습니다. 여러분이 유일하게 해야 할 일은 단지 주석 몇 줄을 지우고, 코드 몇 줄을 지우기만 하면 됩니다.

게임을 위해서, HAND_SIZE를 사용해서 hand에 있는 카드의 수를 결정을 해야 합니다. 원한다면 프로그램에서 HAND_SIZE의 값을 변화시켜가며 게임을 해보세요.

```
def play_game(word_list):
    """
    Allow the user to play an arbitrary number of hands.
    * Asks the user to input 'n' or 'r' or 'e'.
    * If the user inputs 'n', let the user play a new (random) hand.
      When done playing the hand, ask the 'n' or 'e' question again.
    * If the user inputs 'r', let the user play the last hand again.
    * If the user inputs 'e', exit the game.
    * If the user inputs anything else, ask them again.
    """
    # TO DO ...
```

테스팅: 여러분이 구현한 것으로 게임을 해보면서 테스트를 해보세요.

VI. Ghost: 다른 단어 게임

Ghost는 두 명이 하는 매우 유명한 단어 게임입니다. 이 문제에서 우리의 목적은 두 명이 서로 Ghost 게임을 할 수 있도록 상호적인 파이썬 프로그램을 구현하는 것입니다. 여러분이 규칙에 익숙하지 않다면, 위키피디아를 참조하세요. https://en.wikipedia.org/wiki/Ghost_(game) 이 문제 세트에서는 다음 규칙을 따라주세요.

- **Ghost의 규칙**

플레이어는 번갈아 가며 하나의 문자를 말하고, 그 문자를 단어(조각) 끝에 계속 더해가며 단어를 만들어 갑니다. Ghost에서 다음 두 방법을 하게 되면 지게됩니다.

- 3 문자보다 긴 단어를 만드는 것("PEA"는 괜찮지만, "PEAR"를 만들면 집니다)
- 더 이상의 문자들을 더해도 한 단어를 만들 수 없도록 하는 조합을 만드는 것(예를 들어 "QZ")

Ghost에서 이기길 원한다면, 지지 않으면 됩니다! 따라서, 예를 들어 게임은 다음과 같이 진행됩니다.

- 플레이어 1: 'P'를 말합니다.
- 플레이어 2: 'E'를 말합니다.
- 플레이어 1: 'A'를 말합니다. 플레이어 1은 PEA라는 단어를 만들었고, 3문자 이내이므로 괜찮습니다.
- 플레이어 2: 'F'를 말합니다(역주: peaf라는 단어는 없으며, peafowl(공작)이란 단어가 있습니다).
- 플레이어 1: 'O'를 말해야 합니다. PEAF로 시작하는 단어는 하나밖에 없기 때문입니다. 만약 플레이어 1이 'A'를 말하면 지게 됩니다. PEAFA로 시작하는 단어는 없기 때문이죠.
- 플레이어 2: 'W'를 말합니다. 다른 문자를 말할 선택의 여지가 없습니다.
- 플레이어 1: 'L'를 말합니다. PEAFOWL이 단어이기 때문에 플레이어 1이 졌습니다.

문제 7

두 사람이 서로 Ghost 게임을 시작할 수 있도록 `ghost()` 함수를 구현하세요.

시작하기

ps5_ghost.py를 이 문제 세트와 같은 경로에 있는 폴더에 다운로드하고 저장하세요. 이 파일은 단어 목록을 가져오는 함수를 제공합니다.

파일을 수정하기 전에 정상적으로 작동하는지 확인하세요. ps5_ghost.py를 수정하지 않았다면, 다음과 같은 출력물이 나와야 합니다.

```
1  Loading word list from file...
2  83667 words loaded.
```

이 파일 안에서 모든 코딩을 해야 할 것입니다. 파일 안에 정의하는 함수들을 써야 하기 때문이죠.

필요조건

게임을 위한 필요조건은 다음과 같습니다.

- 게임은 상호적이어야 합니다. 각 단계에서 현재 플레이어가 누구인지 그리고 현재 단어 조각이 무엇인지 말해 주어야 합니다.
- 플레이어에게 문자를 입력하라고 요청해야 합니다. 프로그램은 입력 값이 유효한지 확인해야 합니다(알파벳 한 문자이어야 하며, 대소문자 모두 가능합니다).
- 문자는 단어 조각에 더해지며 덧붙여진 단어 조각을 보여주어야 합니다.
- 플레이어가 단어를 만들었거나(3문자보다 긴 단어) 뒤이어 단어가 더 만들어 질 수 없다면 게임이 끝나야 합니다.

`ghost()`를 비롯하여 여러분이 만드는 모든 코드가 함수들로 구성되어야 합니다.

힌트: 문자가 알파벳인지 확인하기 위해 해당 문자가 문자열 `ascii_letters` 안에 있는지 확인하면 됩니다(import 한 string 모듈의 한 부분입니다).

```
>>> import string
>>> 'a' in string.ascii_letters
True
>>> 'F' in string.ascii_letters
True
>>> '2' in string.ascii_letters
False
```

추가 힌트: 함수가 대소문자를 모두 받아야 합니다. 출력할 때는 모두 대문자 혹은 소문자로 일치시키세요(두 가지 중 한가지 방법을 고르세요). 함수 `string.upper()` 혹은 `string.lower()`을 유용하게 사용할 수 있습니다. 자세한 정보는 파이썬 문서를 참조하세요.
https://docs.python.org/3.4/library/index.html

예제 게임은 다음과 같습니다.

```
Welcome to Ghost!
Player 1 goes first.
```

```
Current word fragment: ''
Player 1's turn.
Player 1 says letter: p
Current word fragment: 'p'
Player 2's turn.
Player 2 says letter: y
Current word fragment: 'py'
Player 1's turn.
Player 1 says letter: t
Current word fragment: 'pyt'
Player 2's turn.
Player 2 says letter: h
Current word fragment: 'pyth'
Player 1's turn.
Player 1 says letter: o
Current word fragment: 'pytho'
Player 2's turn.
Player 2 says letter: n
Current word fragment: 'python'
Player 2 loses because 'python' is a word!
Player 1 wins!
```

다른 예제를 하나 더 추가합니다. 유효하지 않은 단어 조각을 만들 때 플레이어가 지는 것을 보여줍니다.

```
Welcome to Ghost!
Player 1 goes first.
Current word fragment: ''
Player 1's turn.
Player 1 says letter: p
Current word fragment: 'p'
Player 2's turn.
Player 2 says letter: y
Current word fragment: 'py'
Player 1's turn.
Player 1 says letter: n
Current word fragment: 'pyn'
Player 1 loses because no word begins with 'pyn'!
Player 2 wins!
```

이 문제에서 겁먹지 마세요! 보기보다 쉽습니다. 문제를 논리적으로 작은 문제들로 나누세요. 이 게임을 실행하는데 어떤 함수들이 필요한가요?

제출 과정

저장. 워드 게임 파일은 ps5.py 안에 작성하세요. 마찬가지로 ghost 파일은 ps5_ghost.py 안에 작성하세요. 파일 이름을 바꾸지 마세요.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 파일 이름과 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적어주세요. 또한 공동 작업을 한 학우(들)이 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# ps5.py
# Problem Set 5
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

```
# ps5.py
#
# Problem Set 5: 6.00 Word Game
# Name:
# Collaborators:
# Time:
#

import random
import string

VOWELS = 'aeiou'
CONSONANTS = 'bcdfghjklmnpqrstvwxyz'
HAND_SIZE = 7

SCRABBLE_LETTER_VALUES = {
    'a': 1, 'b': 3, 'c': 3, 'd': 2, 'e': 1, 'f': 4, 'g': 2, 'h': 4, 'i': 1, 'j': 8,
    'k': 5, 'l': 1, 'm': 3, 'n': 1, 'o': 1, 'p': 3, 'q': 10, 'r': 1, 's': 1, 't': 1,
    'u': 1, 'v': 4, 'w': 4, 'x': 8, 'y': 4, 'z': 10
}
# -----------------------------------
# Helper code
# (you don't need to understand this helper code)

WORDLIST_FILENAME = "words.txt"

def load_words():
   """
   Returns a list of valid words. Words are strings of lowercase letters.

```

```
    Depending on the size of the word list, this function may
    take a while to finish.
    """
    print("Loading word list from file...")
    # inFile: file
    inFile = open(WORDLIST_FILENAME, 'r')
    # wordlist: list of strings
    wordlist = []
    for line in inFile:
    wordlist.append(line.strip().lower())
    print("  ", len(wordlist), "words loaded.")
    return wordlist

def get_frequency_dict(sequence):
    """
    Returns a dictionary where the keys are elements of the sequence
    and the values are integer counts, for the number of times that
    an element is repeated in the sequence.

    sequence: string or list
    return: dictionary
    """
    # freqs: dictionary (element_type -> int)
    freq = {}
    for x in sequence:
    freq[x] = freq.get(x,0) + 1
    return freq

# (end of helper code)
# -----------------------------------

#
# Problem #1: Scoring a word
#
def get_word_score(word, n):
    """
    Returns the score for a word. Assumes the word is a
    valid word.

    The score for a word is the sum of the points for letters
    in the word, plus 50 points if all n letters are used on
    the first go.

    Letters are scored as in Scrabble; A is worth 1, B is
    worth 3, C is worth 3, D is worth 2, E is worth 1, and so on.

    word: string (lowercase letters)
    returns: int >= 0
    """
    # TO DO ...

```

```
#
# Make sure you understand how this function works and what it does!
#
def display_hand(hand):
    """
    Displays the letters currently in the hand.

    For example:
      display_hand({'a':1, 'x':2, 'l':3, 'e':1})
    Should print(out something like:)
      a x x l l l e
    The order of the letters is unimportant.

    hand: dictionary (string -> int)
    """
    # TO DO ...
#
# Make sure you understand how this function works and what it does!
#
def deal_hand(n):
    """
    Returns a random hand containing n lowercase letters.
    At least n/3 the letters in the hand should be VOWELS.

    Hands are represented as dictionaries. The keys are
    letters and the values are the number of times the
    particular letter is repeated in that hand.

    n: int >= 0
    returns: dictionary (string -> int)
    """
    hand={}
    num_vowels = n // 3

    for i in range(num_vowels):
       x = VOWELS[random.randrange(0,len(VOWELS))]
       hand[x] = hand.get(x, 0) + 1

    for i in range(num_vowels, n):
       x = CONSONANTS[random.randrange(0,len(CONSONANTS))]
       hand[x] = hand.get(x, 0) + 1

    return hand

#
# Problem #2: Update a hand by removing letters
#
def update_hand(hand, word):
    """
    Assumes that 'hand' has all the letters in word.
    In other words, this assumes that however many times
```

```
    a letter appears in 'word', 'hand' has at least as
    many of that letter in it.
    Updates the hand: uses up the letters in the given word
    and returns the new hand, without those letters in it.

    Has no side effects: does not mutate hand.

    word: string
    hand: dictionary (string -> int)
    returns: dictionary (string -> int)
    """
    # TO DO ...

#
# Problem #3: Test word validity
#
def is_valid_word(word, hand, word_list):
    """
    Returns True if word is in the word_list and is entirely
    composed of letters in the hand. Otherwise, returns False.
    Does not mutate hand or word_list.

    word: string
    hand: dictionary (string -> int)
    word_list: list of lowercase strings
    """
    if word not in word_list:
       return False
    updated_hand = hand.copy()
    for char in word:
       num = updated_hand.get(char, 0)
       if not num:
          return False
       updated_hand[char] -= 1
    return True

#
# Problem #4: Playing a hand
#
def play_hand(hand, word_list):
    """
    Allows the user to play the given hand, as follows:
    * The hand is displayed.
    * The user may input a word.
    * An invalid word is rejected, and a message is displayed asking
      the user to choose another word.
    * When a valid word is entered, it uses up letters from the hand.
    * After every valid word: the score for that word and the total
      score so far are displayed, the remaining letters in the hand
      are displayed, and the user is asked to input another word.
    * The sum of the word scores is displayed when the hand finishes.
```

```
    * The hand finishes when there are no more unused letters.
      The user can also finish playing the hand by inputing a single
      period (the string '.') instead of a word.
    * The final score is displayed.
      hand: dictionary (string -> int)
      word_list: list of lowercase strings
    """
    # TO DO ...

#
# Problem #5: Playing a game
# Make sure you understand how this code works!
#
def play_game(word_list):
    """
    Allow the user to play an arbitrary number of hands.
    * Asks the user to input 'n' or 'r' or 'e'.
    * If the user inputs 'n', let the user play a new (random) hand.
      When done playing the hand, ask the 'n' or 'e' question again.
    * If the user inputs 'r', let the user play the last hand again.
    * If the user inputs 'e', exit the game.
    * If the user inputs anything else, ask them again.
    """
   # TO DO ...
    hand = deal_hand(HAND_SIZE) # random init
    while True:
      cmd = input('Enter n to deal a new hand, r to replay the last hand,
            or e to end game: ')
     if cmd == 'n':
        hand = deal_hand(HAND_SIZE)
        play_hand(hand.copy(), word_list)
        print()
     elif cmd == 'r':
        play_hand(hand.copy(), word_list)
        print()
     elif cmd == 'e':
        break
     else:
        print("Invalid command.")

#
# Build data structures used for entire session and play game
#
if __name__ == '__main__':
word_list = load_words()
play_game(word_list)
```

문제 세트 6
: 단어 게임 II

서문

이 문제 세트에서는 6.00 단어 게임을 스스로 할 수 있는 프로그램을 만들 것입니다. 한 명이 할 수 있는 단어 게임을 만들었던 문제 세트 5의 연장선입니다.

- **과제량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상했던 것보다 더 많은 시간이 걸리는 문제를 내지 않도록 주의할 것입니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 각각의 학생은 개별적으로 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

문제 1 얼마나 오래 걸리나요?

여러분의 친구 중 여러분과 단어 게임을 할 때마다 이기는 친구가 있습니다. 친구가 이기는 이유는 굉장히 오랫동안 생각하기 때문입니다. 친구가 그런 전략을 사용하지 못하도록 게임 규칙을 바꾸기로 했습니다. 점수는 전과 똑같이 받게 되지만 한 단어를 찾았을 때 원래 점수에서 그 단어를 찾기 위해 걸리는 시간으로 나눈 점수를 받게 됩니다. 단어 당 받게 되는 점수는 소수점 두 자리까지 보여주어야 합니다.

우선, ps5.py을 복사해 ps6.py라는 이름으로 저장하세요(단어 목록이 있는 words.txt는 같은 경로에 있는 폴더에 있도록 하세요).

play_hand 함수를 수정하여 게임이 다음과 같이 되도록 하세요(굵은 글씨로 표시된 부분에 주의하세요).

```
Current Hand: a c I h m m z
Enter word, or a . to indicate that you are finished: him
It took 6.47 seconds to provide an answer.
him earned 1.24 points. Total: 1.24 points
Current Hand: a c m z
Enter word, or a .  to indicate that you are finished: cam
It took 3.25 seconds to provide an answer.
cam earned 2.15 points. Total: 3.39 points
Current Hand: z
Enter word, or a . to indicate that you are finished: .
Total score: 3.39 points.
```

[주의] 만약 누군가가 단어를 굉장히 빠르게 입력한다면 어떻게 될까요? 시간은 최소의 값을 가진 단위 discrete 시간으로 계산됩니다. 그래서 단어를 매우 빠르게 입력하면 반올림하여 0초가 되기 때문에 0으로 나누면 에러가 발생할 수도 있습니다. 이러한 경우에 여러분이 어떻게 하는 것이 합리적일지 결정한 다음, 직접 해보세요. 여러분이 어떻게 바꾸었는지 기록하는 것을 잊지 마세요.

힌트: 다음 코드는 여러분의 이름을 입력하는 데 얼만큼의 시간이 소요되는지 알려줍니다.

```
import time

start_time = time.time()
name = input('What is your name?: ')#use raw_input() in python 2.x
end_time = time.time()
total_time = end_time - start_time
print 'It took %0.2f to enter your name' % total_time
```

문제 2 시간 제한

6.00 단어 게임을 할 때 여러분은 아직도 여러분의 친구가 너무 오래 시간을 쓰는 것이 지루하게 느껴질 것입니다. 여러분은 게임에 "체스 시계"를 더하기로 결정합니다. "체스 시계"는 한 사람이 자기 차례의 게임을 할 때 걸리는 전체 시간을 초 단위로 제한합니다.

다음과 같이 결과가 나오도록 게임을 수정하세요. 여러분의 프로그램은 시간 제한을 두어 일정 시간이 지나고 난 다음에 입력된 단어는 점수에 더하지 않도록 해야 합니다. 또한, 각각 입력한 후에 남은 시간(>=0이라면)을 출력해야 합니다.

여러분은 사용자의 입력 중 유효한 입력과 유효하지 않은 입력 모두에 대해서 그 시간을 측정해야 합니다. 측정된 시간을 총 시간에 포함하세요. 또한, 사용자가 입력하는 시간만을 측정해야 하며, 컴퓨터가 사용한 과정의 시간은 측정하지 않아야 합니다.

```
Enter time limit, in seconds, for players: 8
Current Hand:  a c i h m m z
Enter word, or a . to indicate that you are finished: him
It took 6.47 seconds to provide an answer.
You have 1.53 seconds remaining.
him earned 1.24 points. Total: 1.24 points
Current Hand:  a c m z
Enter word, or a . to indicate that you are finished: cam
It took 3.25 seconds to provide an answer.
Total time exceeds 8 seconds. You scored 1.24 points.
```

문제 3 컴퓨터 플레이어

여러분은 6.00 단어 게임을 만들기 위해 많은 시간을 썼지만 불행히도 여러분과 단어 게임을 함께 할 친구가 더 이상 없습니다. 그래서 여러분은 컴퓨터 플레이어의 도움을 받기로 했습니다.

사용자가 단어를 입력하는 대신에 다음과 같은 함수를 호출하여 input(play_hand 안에서)을 대체할 것입니다.

```
def pick_best_word(hand, points_dict):
    """
    Return the highest scoring word from points_dict that can be made with the
    given hand.
    Return '.' if no words can be made with the given hand.
    """
    ### implement me! ###
```

(여러분 중 누군가는 이 함수의 이름이 마음에 들지 않을 것입니다. 한 번의 순서 동안 가장 높은 점수의 단어를 입력하는 것은 총 점수를 최대화시키는 데 가장 좋은 방법이 아닐 수 있습니다. 이 문제에 대해서는 나중에 이야기 합시다.)

이 문제의 첫 번째 과정과 마찬가지로 다음과 같은 예비 실행 함수를 만드세요:

```
def get_words_to_points(word_list):
    """
    Return a dict that maps every word in word_list to its point value.
    """
```

이 함수가 몇 번 호출되어야 하는지 생각해 보세요 - 게임 당 한 번 호출될까요, hand 마다 한 번 호출될까요, 아니면 hand 안에서 각 순서마다 호출될까요? 이것을 구현하려면, 여러분은 전역변수 `points_dict`을 생성하여 결과로 나오는 딕셔너리를 저장해야 합니다.

다음으로, `word_list` 아니라고 여러분이 위에서 만든 표현(딕셔너리)을 인자를 받기 위하여 is_valid_word를 변경하세요. 어떤 표현의 선택하느냐는 실행에 큰 영향을 줄 것입니다. 이렇게 수정하는 것은 `is_valid_word`의 복잡성을 `O(len(word_list))`에서 `O(1)`로 향상시킬 것입니다.

마지막으로, `pick_best_word`를 만드세요(최적화하기 위해 너무 걱정하지 않아도 됩니다. 여러분이 쉽게 디버깅 할 수 있도록 직접적으로 구현하세요). `play_hand`를 수정하여 이 함수를 호출하세요.

컴퓨터 플레이어에게 사람 플레이어와 같은 시간을 주는 것은 공평하지 않습니다. 다음과 같은 함수를 써서 컴퓨터의 시간 제한을 두세요. 어떤 컴퓨터들은 다른 컴퓨터보다 상당히 빠르다는 사실을 잘 고려하기 위하여 지금 이런 작업을 하고 있다는 것을 기억해 두세요. k=1와 다른 값을 사용해 여러분이 구현한 것을 시험해보세요.

```
import time
def get_time_limit(points_dict, k):
    """
    Return the time limit for the computer player as a function of the multiplier k.
    points_dict should be the same dictionary that is created by
    get_words_to_points.
    """
    start_time = time.time()
    # Do some computation. The only purpose of the computation is so we can
    # figure out how long your computer takes to perform a known task.
    for word in points_dict:
    get_frequency_dict(word)
    get_word_score(word, HAND_SIZE)
    end_time = time.time()
    return (end_time - start_time) * k
```

마찬가지로 이 함수가 얼마나 호출될 것인지 생각해보세요. 이 함수의 결과값을 저장하도록 전역변수 time_limit을 정의하세요.

다음과 같이 동작하도록 구현하면 됩니다. 'zig', 'lo', '.' 은 사용자가 입력하는 값이 아니고 컴퓨터가 자동으로 가장 높은 점수를 갖는 단어를 찾아주는 값입니다. 즉 게임을 실행하면, 자동으로 다음과 같이 게임을 하게 되는 것입니다.

```
Enter n to deal a new hand, r to replay the last hand, or e to end game: n
Current Hand: g q i l o z z
Enter word, or a . to indicate that you are finished: zig
It took 0.10 seconds to provide an answer.
You have 0.68 seconds remaining.
zig earned 135.41 points. Total: 135.41 points
Current Hand: q l o z
Enter word, or a . to indicate that you are finished: lo
It took 0.14 seconds to provide an answer.
You have 0.55 seconds remaining.
lo earned 14.49 points. Total: 149.90 points
Current Hand: q z
Enter word, or a . to indicate that you are finished: .
It took 0.12 seconds to provide an answer.
You have 0.43 seconds remaining.
Total score: 149.90 points.
```

문제 4 알고리즘 분석

위 문제에서 구현한 pick_best_word의 시간 복잡도를 설명하세요.

ps6.py에 아래와 같이 주석으로 답을 하시면 됩니다.

```
## Problem 4 ##
# your response here.
# as many lines as you want.
```

제출 과정

저장. 문제를 ps6.py 안에 작성하세요. 다음 함수들에 대해 완성하고 테스트 과정을 마친 후에 제출하세요(get_words_to_points, get_word_rearrangements, pick_best_word, pick_best_word_faster). 최종 제출에 있어서 함수 play_game과 play_hand

는 다음 두 함수 pick_best_word 혹은 pick_best_word_faster 중에 선택해서 6.00 단어 게임을 실행해야 합니다.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 파일 이름과 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적어주세요. 또한 공동 작업을 한 학우(들)이 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# ps6.py
# Problem Set 6
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

```
# ps6.py
#
# Problem Set 6: 6.00 Word Game
# Name:
# Collaborators:
# Time:
#

import random
import string
import time

VOWELS = 'aeiou'
CONSONANTS = 'bcdfghjklmnpqrstvwxyz'
HAND_SIZE = 7

SCRABBLE_LETTER_VALUES = {
    'a': 1, 'b': 3, 'c': 3, 'd': 2, 'e': 1, 'f': 4, 'g': 2, 'h': 4, 'i': 1, 'j': 8,
    'k': 5, 'l': 1, 'm': 3, 'n': 1, 'o': 1, 'p': 3, 'q': 10, 'r': 1, 's': 1, 't': 1,
    'u': 1, 'v': 4, 'w': 4, 'x': 8, 'y': 4, 'z': 10
}

# -----------------------------------
# Helper code
# (you don't need to understand this helper code)

WORDLIST_FILENAME = "words.txt"

def load_words():
    """
    Returns a list of valid words. Words are strings of lowercase letters.

    Depending on the size of the word list, this function may
```

```
    take a while to finish.
    """
    print("Loading word list from file...")
    # inFile: file
    inFile = open(WORDLIST_FILENAME, 'r')
    # wordlist: list of strings
    wordlist = []
    for line in inFile:
        wordlist.append(line.strip().lower())
    print("  ", len(wordlist), "words loaded.")
    return wordlist

def get_frequency_dict(sequence):
    """
    Returns a dictionary where the keys are elements of the sequence
    and the values are integer counts, for the number of times that
    an element is repeated in the sequence.

    sequence: string or list
    return: dictionary
    """
    # freqs: dictionary (element_type -> int)
    freq = {}
    for x in sequence:
        freq[x] = freq.get(x,0) + 1
    return freq

# (end of helper code)
# -----------------------------------
#
# Problem #5: Scoring a word
#
def get_word_score(word, n):
    """
    Returns the score for a word. Assumes the word is a
    valid word.

    The score for a word is the sum of the points for letters
    in the word, plus 50 points if all n letters are used on
    the first go.

    Letters are scored as in Scrabble; A is worth 1, B is
    worth 3, C is worth 3, D is worth 2, E is worth 1, and so on.

    word: string (lowercase letters)
    returns: int >= 0
    """
    # To Do ...

#
# Make sure you understand how this function works and what it does!
```

```
#
def display_hand(hand):
    """
    Displays the letters currently in the hand.

    For example:
      display_hand({'a':1, 'x':2, 'l':3, 'e':1})
    Should print(out something like:)
      a x x l l l e
    The order of the letters is unimportant.

    hand: dictionary (string -> int)
    """
    # To Do ...

#
# Make sure you understand how this function works and what it does!
#
def deal_hand(n):
    """
    Returns a random hand containing n lowercase letters.
    At least n/3 the letters in the hand should be VOWELS.

    Hands are represented as dictionaries. The keys are
    letters and the values are the number of times the
    particular letter is repeated in that hand.

    n: int >= 0
    returns: dictionary (string -> int)
    """
    hand={}
    num_vowels = n // 3

    for i in range(num_vowels):
        x = VOWELS[random.randrange(0,len(VOWELS))]
        hand[x] = hand.get(x, 0) + 1

    for i in range(num_vowels, n):
        x = CONSONANTS[random.randrange(0,len(CONSONANTS))]
        hand[x] = hand.get(x, 0) + 1

    return hand

#
# Problem #6_3: Update a hand by removing letters
#
def update_hand(hand, word):
    """
    Assumes that 'hand' has all the letters in word.
    In other words, this assumes that however many times
    a letter appears in 'word', 'hand' has at least as
```

```
    many of that letter in it.

    Updates the hand: uses up the letters in the given word
    and returns the new hand, without those letters in it.

    Has no side effects: does not mutate hand.

    word: string
    hand: dictionary (string -> int)
    returns: dictionary (string -> int)
    """
    # To do ...

#
# Problem #6_3: Test word validity
#
def is_valid_word(word, hand):
    """
    Returns True if word is entirely
    composed of letters in the hand. Otherwise, returns False.
    Does not mutate hand.

    word: string
    hand: dictionary (string -> int)
    """
    # To Do ...

#
# Problem #6_3: Playing a hand
#
def play_hand(hand, word_list):
    """
    Allows the user to play the given hand, as follows:
    * The hand is displayed.
    * Computer inputs the best word.
    * When a valid word is entered, it uses up letters from the hand.
    * After every valid word: the score for that word and the total
      score so far are displayed, the remaining letters in the hand
      are displayed, and the computer is asked to input another word.
    * The sum of the word scores is displayed when the hand finishes.
    * The hand finishes when there are no more unused letters.
      The user can also finish playing the hand by inputing a single
      period (the string '.') instead of a word.
    * The final score is displayed.

    hand: dictionary (string -> int)
    word_list: list of lowercase strings
    """
    # To Do ...

#
```

```
# Problem #6_3
#
def pick_best_word(hand, points_dict):
    """
    Return the highest scoring word from points_dict that can be made with the
    given hand.

    Return '.' if no words can be made with the given hand.
    """
    # To Do ...

#
# Problem #6_3
#
def get_words_to_points(word_list):
    """
    Return a dict that maps every word in word_list to its point value.
    """
    # To Do ...

def get_time_limit(points_dict, k):
    """
    Return the time limit for the computer player as a function of the
    multiplier k.
    points_dict should be the same dictionary that is created by
    get_words_to_points.
    """
    start_time = time.time()
    # Do some computation. The only purpose of the computation is so we can
    # figure out how long your computer takes to perform a known task.
    for word in points_dict:
        get_frequency_dict(word)
        get_word_score(word, HAND_SIZE)
    end_time = time.time()
    return (end_time - start_time) * k
#
# Make sure you understand how this code works!
#
def play_game(word_list):
    """
    Allow the user to play an arbitrary number of hands.
    * Asks the user to input 'n' or 'r' or 'e'.
    * If the user inputs 'n', let the user play a new (random) hand.
      When done playing the hand, ask the 'n' or 'e' question again.
    * If the user inputs 'r', let the user play the last hand again.
    * If the user inputs 'e', exit the game.
    * If the user inputs anything else, ask them again.
    """
    # TO DO ...
    hand = deal_hand(HAND_SIZE) # random init
    while True:
```

```
        cmd = input('Enter n to deal a new hand, r to replay the last hand, or e to end game: ')
        if cmd == 'n':
            hand = deal_hand(HAND_SIZE)
            play_hand(hand.copy(), word_list)
            print()
        elif cmd == 'r':
            play_hand(hand.copy(), word_list)
            print()
        elif cmd == 'e':
            break
        else:
            print("Invalid command.")

#
# Build data structures used for entire session and play game
#
if __name__ == '__main__':
    word_list = load_words()
    play_game(word_list)
```

문제 세트 7
: 간단한 연습 문제들

서문

이 문제 세트는, 강의에서는 다루었지만 프로그래밍 문제들에서는 강조하지 않는 주제들을 점검함으로 여러분의 이해를 확실히 하기 위하여 만들었습니다. 여러분 자신이 직접 해보기를 적극 권장하지만, 제출할 필요는 없습니다.

문제 1

`fact0`의 계산복잡도는 어떻게 되나요? 답을 설명하세요.

```
1  def fact0(i):
2      assert type(i) == int and i >= 0
3      if i == 0 or i == 1:
4          return 1
5      return i * fact0(i-1)
```

문제 2

`fact1`의 계산복잡도는 어떻게 되나요? 답을 설명하세요.

```
1  def fact1(i):
2      assert type(i) == int and i >= 0
3      res = 1
4      while i > 1:
5          res = res * i
6          i -= 1
7      return res
```

문제 3

makeSet의 계산복잡도는 어떻게 되나요? 답을 설명하세요.

```
def makeSet(s):
    assert type(s) == str
    res = ''
    for c in s:
        if not c in res:
            res = res + c
    return res
```

문제 4

intersect의 계산복잡도는 어떻게 되나요? 답을 설명하세요.

```
def intersect(s1, s2):
    assert type(s1) == str and type(s2) == str
    s1 = makeSet(s1)
    s2 = makeSet(s2)
    res = ''
    for e in s1:
        if e in s2:
            res = res + e
     return res
```

문제 5

아래의 코드를 손으로 직접 시뮬레이션 해보세요. 각 연산의 과정마다 해당 변수의 값이 어떻게 변하는지 설명하세요. s1과 s2는 하나 이상의 범위 안에 존재한다는 사실에 주목하세요.

```
def swap0(s1, s2):
    assert type(s1) == list and type(s2) == list
    tmp = s1[:]
    s1 = s2[:]
    s2 = tmp
    return
```

```
s1 = [1]
s2 = [2]
swap0(s1, s2)
print(s1, s2)
```

문제 6

아래의 코드를 손으로 직접 시뮬레이션 해보세요.

```
def swap1(s1, s2):
    assert type(s1) == list and type(s2) == list
    return s2, s1
s1 = [1]
s2 = [2]
s1, s2 = swap1(s1, s2)
print(s1, s2)
```

문제 7

아래의 코드를 손으로 직접 시뮬레이션 해보세요.

```
def rev(s):
    assert type(s) == list
    for i in range(len(s)/2):
        tmp = s[i]
        s[i] = s[-(i+1)]
        s[-(i+1)] = tmp
s = [1,2,3]
rev(s)
print(s)
```

문제 세트 8
: 수강신청 도우미

서문

이름을 밝힐 수 없는 어느 한 고등교육기관에서 조언자들은 학생들의 목적에 맞게 각 학생에게 적합한 수강 과목 목록을 정할 수 있도록 도움을 주어왔습니다. 그러나, 재정적인 문제 때문에, 그 교육 기관은 조언자들을 소프트웨어로 대체하기로 결정했습니다. 학생들이 하고 싶은 공부의 양이 주어지면, 프로그램은 그 가치를 최대화시키는 수강과목 목록을 반환합니다.

이 문제 세트의 목적은 동적 프로그래밍 알고리즘을 구현하고, **함수**를 매개 변수로 어떻게 넘기는 지에 대해 배우는 것입니다.

- **과제량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상했던 것보다 더 많은 시간이 걸리는 문제를 내지 않도록 주의할 것입니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 각각의 학생은 개별적으로 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

시작하기

다음 파일들을 같은 폴더에 다운로드하고 저장하세요.

- ps8.py: 문제 1-5를 작성할 골격skeleton(스켈레톤)입니다.
- subjects.txt: value와 work에 대한 정보를 담고 있는 과목 리스트입니다.

문제 1 과목 딕셔너리 만들기

첫 단계는 ps8.py에 있는 loadSubjects를 구현하는 것입니다. 이 함수는 파일에서 과목의 목록을 가져옵니다. 파일의 각 줄에는 "name, value, work"의 형태의 문자열이 있습니다. 여기서 name은 문자열을 의미하며, value는 학생이 과목을 선택함으로써 배우는 정도의 양을 정수로 나타내고, work는 학생이 과목을 통과하기 위해 사용해야 하는 시간을 정수로 나타냅니다. loadSubjects는 과목 이름을 튜플로 매핑하는 딕셔너리를 반환합니다. 여기서 튜플은 정수 쌍(배우는 양, 공부하는 시간)으로 구성되어 있습니다.

```
def loadSubjects(filename):
    """
    Returns a dictionary mapping subject name to (value, work), where the
    name is a string and the value and work are integers. The subject
    information is read from the file named by the string filename. Each line of
    the file contains a string of the form "name,value,work".

    returns: dictionary mapping subject name to (value, work)
    """

    # The following sample code reads lines from the specified file and prints
    # each one.
    inputFile = open(filename)
    for line in inputFile:
        print(line)

    # TODO: Instead of printing each line, modify the above to parse the name,
    # value, and work of each subject and create a dictionary mapping the name
    # to the (value, work).
```

힌트: 이것을 구현하기 위해 문자열 메소드인 `split()`와 `strip()`을 유용하게 사용할 수 있습니다. 온라인 Python Library Reference[1] 문서를 확인하세요. 특히 `split()`을 사용해서 쉼표를 나누고 `strip()`을 사용해 각 줄의 끝에 있는 줄 넘김line break 문자를 제거할 수 있습니다.

loadSubjects에서 반환한 딕셔너리가 전체 과목 목록을 표현한 것으로 생각할 수 있습니다. 앞으로 계속 나오는 문제들에서도 우리는 이와 같은 형태의 딕셔터리를 사용하여 선택한 과목들을 표현할 것입니다(전체 과목 목록의 부분집합이죠).

1. https://docs.python.org/3/library/stdtypes.html#string-methods

이러한 딕셔너리의 내용을 깔끔하게 보여주기 위해서 우리는 printSubjects라는 함수를 제공했습니다. 여기 출력물 예제가 있습니다.

```
>>> subjects = loadSubjects(SUBJECT_FILENAME)
>>> printSubjects(subjects)
Course  Value  Work
======  ====  =====
10.00  1  20
10.01  2  20
10.02  1  16
10.03  6  19
10.04  3  13
10.15  8  13
[... TRUNCATED ...]
9.16  9  10
9.17  9  11
9.18  7  16
9.19  9  9

Total Value:  1804
Total Work: 3442
```

subjects.txt 파일이 너무 크기 때문에, 여러분 나름대로 작은 크기의 subjects.txt를 만들어, 다룰 수 있을 정도의 적은 과목들로 함수를 테스트 해 보아도 좋습니다.

문제 2 그리디 최적화를 이용한 과목 선택

해당 고등교육기관이 프로그램을 구현하기 위해 여러분을 고용했다고 가정합시다. 그들이 여러분에게 그리디 방법으로greedyAdvisor 각 학생들의 제한 조건(학생이 하고자 하는 공부량)을 만족하는 과목들의 리스트를 구현하라고 했습니다.

알고리즘은 "최고의best" 과목부터 선택해야 합니다. "최고의" 라는 의미는 비교함수comparator를 사용해 결정할 수 있습니다. 비교함수는 인자 두 개를 받는 함수입니다. 각각은 (value, work) 튜플이며, 처음 인자가 두 번째 인자보다 "더 나은지better" 확인하고 불리언(True 혹은 False)을 반환합니다. 여기서 "더 나은지" 의 정의는 서로 다른 비교함수를 넘겨받아 비교한다면, 그 결과는 달라질 수 있습니다.

여러분이 인자로 넘길 수 있는 세 가지 비교함수들이 제공됩니다.

- `cmpValue`: 과목의 value를 비교합니다.
- `cmpWork`: 과목들의 workload를 비교합니다.
- `cmpRatio`: 과목들의 value/workload를 비교합니다.

예를 들어, 만약 우리에게 다음의 과목 딕셔너리smallCatalog와 최대 15시간 공부할 수 있는 시간work이 주어진다고 가정합시다.

```
1  # name  value  work
2  {'6.00':  (16,  8),
3  '1.00':  (7,  7),
4  '6.01':  (5,  3),
5  '15.01':  (9,  6)}
```

우리가 `greedyAdvisor`를 value 비교함수와 함께 사용한다면(함수의 매개변수가 어떻게 되는지 아래를 통해 확인하세요),

```
>>> greedyAdvisor(smallCatalog, 15, cmpValue)
```

위 함수는 비교함수comparator `cmpValue`를 사용하며, 6.00을 우선으로 고르고, 15.01을 다음으로 고른 후에, 다음 딕셔너리를 반환합니다.

```
{'6.00': (16, 8), '15.01': (9, 6)}
```

다른 과목들은 포함되지 않았습니다. 만약 포함한다면, 총 `workload`가 `maxWork` 한계를 넘어서기 때문입니다.

주목 : 과목 이름은 출력하기 전에 문자열로 정렬되어 있습니다. 따라서 순서는 위와 같으며, 또한 그 순서가 항상 보장됩니다.

우리가 `work` 비교함수를 사용한다면,

```
>>> greedyAdvisor(smallCatalog, 15, cmpWork)
```

위 함수는 6.01과 15.01을 순서대로 선택하고 다음을 반환합니다.

```
{'6.01': (5, 3), '15.01': (9, 6)}
```

우리가 `ratio` 비교함수를 사용한다면,

```
>>> greedyAdvisor(smallCatalog, 15, cmpRatio)
```

위 함수는 6.00과 6.01을 순서대로 선택하고 다음을 반환합니다.

```
{'6.00': (16, 8), '6.01': (5, 3)}
```

다시, greedyAdvisor를 통해 반환된 딕셔너리를 printSubjects를 사용해서 보기 좋게 만들 수 있습니다.

```
>>> selected = greedyAdvisor(smallCatalog, 15, cmpRatio)
>>> printSubjects(selected)
Course  Value  Work
======  ====  =====
6.00  16  8
6.01  5  3
Total Value:  21
Total Work:  11
```

ps8.py 파일에 greedyAdvisor를 구현하세요.

```
def greedyAdvisor(subjects, maxWork, comparator):
    """
    Returns a dictionary mapping subject name to (value, work) which includes
    subjects selected by the algorithm, such that the total work of subjects
    in the dictionary is not greater than maxWork.  The subjects are chosen
    using a greedy algorithm.  The subjects dictionary should not be mutated.

    subjects: dictionary mapping subject name to (value, work)
    maxWork: int >= 0
    comparator: function taking two tuples and returning a bool
    returns: dictionary mapping subject name to (value, work)
    """
    # TODO...
```

[주의] 주어진 입력에 대해 항상 한 가지의 정답이 있지는 않습니다. 예를 들어 여러 수업이 동일한 value 혹은 work 값을 가질 수 있습니다. 또한, 그리디 알고리즘을 subjects.txt 파일에 있는 전체 과목 데이터로 확인하세요.

문제 3 브루트포스(brute-force, 무차별) 알고리즘을 이용한 과목 선택

강의에서 본 것 같이, 그리디 알고리즘이 항상 최적의 해를 계산해주지는 않습니다. 전역적으로 최적의 해를 찾는 한 가지 방법은 브루트포스 알고리즘을 사용하는 것입니다. 모든 가능한 해답

을 전부 나열하고, 주어진 제한조건 하에서 최고의 값을 찾는 방법입니다.

ps8.py에 브루트포스 알고리즘을 구현한 함수 bruteForceAdvisor를 제공했습니다. 이 함수는 과목 이름을 (value, work)로 매핑하는 딕셔너리를 반환하는데, 이것이 바로 브루트포스 알고리즘을 사용하여 전역적으로 최적의 값을 갖는 과목을 나타내는 것이죠.

```
def bruteForceTime():
    """
    Runs tests on bruteForceAdvisor and measures the time required to compute
    an answer.
    """
```

문제 세트 6에서 소개한 time 함수를 사용하여, 서로 다른 maxWork값에 따라 연산 시간이 얼마나 걸리는지 확인해보세요. 작성한 코드를 bruteForceTime 함수에 넣으세요. 브루트포스 알고리즘으로 연산하는데 비합리적으로 많은 시간이 걸리기 바로 전까지의 최대 maxWork값은 어떻게 되나요? (합리적인 시간에 대해서는 스스로 결정해 보세요.) 그리고, 여러분이 작성한 bruteForceTime 코드 다음에 주석으로 여러분이 관찰한 결과를 기록하세요.

문제 4 동적 프로그래밍을 이용한 과목 선택

강의에서 동적 프로그래밍을 이용하면, 지수적 시간이 걸리는 최적화 알고리즘을 의사-다항-시간pseudo-polynomial-time 알고리즘으로 줄일 수 있는 것을 확인했습니다. 기존의 문제를 여러 하위문제sub-problem(부분문제)들로 나누고, 하위문제들에서 나오는 중간 해를 메모이제이션memoizing 함으로써 얻을 수 있습니다. 이것은 최적 해를 더욱 빠르게 찾을 수 있도록 도와줍니다. 그 이유는 하위문제에서 나오는 답이 여러 번 사용되고, 메모이제이션은 이러한 해답을 얻기 위해 다시 계산하는 과정을 없애주기 때문입니다.

이 문제에서는 과목을 선택함에 있어 전역적으로 최적 해를 찾기 위해 동적 프로그래밍을 사용합니다.

ps8.py에 dpAdvisor를 구현하세요. 이 함수는 동적 프로그래밍 방법을 사용해서 과목 이름을 (value, work)로 매핑하는 딕셔너리를 반환합니다.

```
def dpAdvisor(subjects, maxWork):
    """
```

```
    Returns a dictionary mapping subject name to (value, work) that contains a
    set of subjects that provides the maximum value without exceeding
    maxWork.

    subjects: dictionary mapping subject name to (value, work)
    maxWork: int >= 0
    returns: dictionary mapping subject name to (value, work)
    """
    # TODO...
```

힌트: 동적 프로그래밍으로 해결할 수 있도록 이 문제를 다른 문제로 줄일 수 있나요?

다음은 문제 1에서 가져온 과목 항목을 사용한 출력물입니다.

```
>>> printSubjects(dpAdvisor(subjects, 30))
Course  Value  Work
======  ====  =====
12.04  7  1
12.09  8  2
14.02  10  2
15.01  7  1
18.08  10  3
2.03  6  1
22.01  6  2
22.03  10  2
22.05  6  2
22.06  10  3
24.12  6  1
6.00  10  1
7.00  7  1
7.05  8  2
7.06  4  1
7.16  7  1
7.17  10  1
7.18  10  2
8.08  4  1
Total Value:  146
Total Work:  30
```

문제 5 실행 시간 비교

문제 3에서 사용한 dpAdvisor의 실행 시간을 측정하세요. 브루트포스 알고리즘과 비교해서 동적 프로그래밍 알고리즘의 실행 시간이 어떠한가요? 시간을 측정하는 코드를 dpTime에 넣고, dpTime 아래에 주석으로 여러분의 관측 결과를 기록하세요.

```
def dpTime():
    """
    Runs tests on dpAdvisor and measures the time required to compute an
    answer.
    """
    # TODO...
```

제출 과정

저장. ps8.py 안에 수정한 코드를 작성하세요.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적으세요. 또한 공동 작업한 학우가 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# Problem Set 8
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

다시 한 번 확인하기. 문제 세트를 마친 후에 다시 한 번 확인하세요.

- ps8.py를 실행하고, 에러 없이 동작하는지 확인하세요.
- loadSubjects를 확인하고, 제공된 subjects.txt를 읽어올 수 있는지 확인하세요.
- 문제 2와 4에서 구현한 함수를 실행하고, 예상한 값을 반환하는지 확인하세요.

```
# ps8.py
#
# 6.00 Problem Set 8
#
# Intelligent Course Advisor
#
# Name:
# Collaborators:
# Time:
#

    import time

    SUBJECT_FILENAME = "subjects.txt"
    VALUE, WORK = 0, 1

    #
    # Problem 1: Building A Subject Dictionary
#
def loadSubjects(filename):
    """
    Returns a dictionary mapping subject name to (value, work), where the name
    is a string and the value and work are integers. The subject information is
    read from the file named by the string filename. Each line of the file
    contains a string of the form "name,value,work".

    returns: dictionary mapping subject name to (value, work)
    """

    # The following sample code reads lines from the specified file and prints
    # each one.
    inputFile = open(filename)
    subject_dic = {}
    for line in inputFile:
        print(line)

# TODO: Instead of printing each line, modify the above to parse the name,
# value, and work of each subject and create a dictionary mapping the name
# to the (value, work).

def printSubjects(subjects):
    """
    Prints a string containing name, value, and work of each subject in
    the dictionary of subjects and total value and work of all subjects
    """
       totalVal, totalWork = 0,0
    if len(subjects) == 0:
        return 'Empty SubjectList'
    res = 'Course\tValue\tWork\n======\t====\t=====\n'
    subNames = list(subjects.keys())
    subNames.sort()
```

```
    for s in subNames:
        val = subjects[s][VALUE]
        work = subjects[s][WORK]
        res = res + s + '\t' + str(val) + '\t' + str(work) + '\n'
        totalVal += val
        totalWork += work
    res = res + '\nTotal Value:\t' + str(totalVal) +'\n'
    res = res + 'Total Work:\t' + str(totalWork) + '\n'
    print(res)

def cmpValue(subInfo1, subInfo2):
    """
    Returns True if value in (value, work) tuple subInfo1 is GREATER than
    value in (value, work) tuple in subInfo2
    """
    val1 = subInfo1[VALUE]
    val2 = subInfo2[VALUE]
    return  val1 > val2

def cmpWork(subInfo1, subInfo2):
    """
    Returns True if work in (value, work) tuple subInfo1 is LESS than than work
    in (value, work) tuple in subInfo2
    """
    work1 = subInfo1[WORK]
    work2 = subInfo2[WORK]
    return  work1 < work2

def cmpRatio(subInfo1, subInfo2):
    """
    Returns True if value/work in (value, work) tuple subInfo1 is
    GREATER than value/work in (value, work) tuple in subInfo2
    """
    val1 = subInfo1[VALUE]
    val2 = subInfo2[VALUE]
    work1 = subInfo1[WORK]
    work2 = subInfo2[WORK]
    return float(val1) / work1 > float(val2) / work2

#
# Problem 2: Subject Selection By Greedy Optimization
#
def greedyAdvisor(subjects, maxWork, comparator):
    """
    Returns a dictionary mapping subject name to (value, work) which includes
    subjects selected by the algorithm, such that the total work of subjects in
    the dictionary is not greater than maxWork. The subjects are chosen using
    a greedy algorithm. The subjects dictionary should not be mutated.

    subjects: dictionary mapping subject name to (value, work)
    maxWork: int >= 0
```

```
    comparator: function taking two tuples and returning a bool
    returns: dictionary mapping subject name to (value, work)
    """
    # TODO…

def bruteForceAdvisor(subjects, maxWork):
    """
    Returns a dictionary mapping subject name to (value, work), which
    represents the globally optimal selection of subjects using a brute force
    algorithm.

    subjects: dictionary mapping subject name to (value, work)
    maxWork: int >= 0
    returns: dictionary mapping subject name to (value, work)
    """
    nameList = list(subjects.keys())
    tupleList = list(subjects.values())
    bestSubset, bestSubsetValue = \
          bruteForceAdvisorHelper(tupleList, maxWork, 0, None, None, [], 0, 0)
    outputSubjects = {}
    for i in bestSubset:
    outputSubjects[nameList[i]] = tupleList[i]
    return outputSubjects
def bruteForceAdvisorHelper(subjects, maxWork, i, bestSubset, bestSubsetValue,
                        subset, subsetValue, subsetWork):
    # Hit the end of the list.
    if i >= len(subjects):
       if bestSubset == None or subsetValue > bestSubsetValue:
          # Found a new best.
          return subset[:], subsetValue
       else:
          # Keep the current best.
          return bestSubset, bestSubsetValue
    else:
       s = subjects[i]
       # Try including subjects[i] in the current working subset.
       if subsetWork + s[WORK] <= maxWork:
          subset.append(i)
          bestSubset, bestSubsetValue = bruteForceAdvisorHelper(subjects,
                maxWork, i+1, bestSubset, bestSubsetValue, subset,
                subsetValue + s[VALUE], subsetWork + s[WORK])
          subset.pop()
       bestSubset, bestSubsetValue = bruteForceAdvisorHelper(subjects,
             maxWork, i+1, bestSubset, bestSubsetValue, subset,
             subsetValue, subsetWork)
       return bestSubset, bestSubsetValue

#
# Problem 3: Subject Selection By Brute Force
#
```

```
153     def bruteForceTime():
154     """
155     Runs tests on bruteForceAdvisor and measures the time required to compute
156     an answer.
157     """
158     # TODO...
156
160 # Problem 3 Observations
161 # ======================
162 #
163 # TODO: write here your observations regarding bruteForceTime's performance
164
165 #
166 # Problem 4: Subject Selection By Dynamic Programming
167 #
168 def dpAdvisor(subjects, maxWork):
169     """
170     Returns a dictionary mapping subject name to (value, work) that contains a
171     set of subjects that provides the maximum value without exceeding maxWork.
172
173     subjects: dictionary mapping subject name to (value, work)
174     maxWork: int >= 0
175     returns: dictionary mapping subject name to (value, work)
176     """
177     # TODO...
178
179 #
180 # Problem 5: Performance Comparison
181 #
182 def dpTime():
183     """
184     Runs tests on dpAdvisor and measures the time required to compute an
185     answer.
186     """
187     # TODO...
188
189 # Problem 5 Observations
190 # ======================
191 #
192 # TODO: write here your observations regarding dpAdvisor's performance and
193 # how its performance compares to that of bruteForceAdvisor.
```

문제 세트 9
: 더 많은 클래스가 필요한 것처럼

서문

이 문제 세트에서는 클래스class와 메소드method를 생성하고 사용하는 것을 소개합니다. 그러면서 여러분은 메소드에 대한 상속과 오버라이딩overriding 등의 개념에 익숙해져야 합니다.

- **과제량**

각각의 문제를 푸는데 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상했던 것보다 더 많은 시간이 걸리는 문제를 내지 않도록 주의할 것입니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 각각의 학생은 개별적으로 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

시작하기

이 문제 세트에 대한 여러분의 코드를 템플릿 파일 ps9.py에 작성하세요:

문제 4에서는 샘플 입력 파일 shapes.txt을 사용하세요:

문제 1

이 문제 세트에서는 도형들을 캡슐화하기 위해 클래스를 사용합니다.

여러분을 위해 도형의 하위클래스subclasses 두 개, 즉 Square와 Circle 클래스가 구현되어 있습니다. 이 도형들 각각에 대해서 면적을 구하는 것(각각의 area 메소드가 어떻게 Shape라는 부모클

래스의 area 메소드를 오버라이드 할 것인지에 대해 주의하세요), 동일성equality을 확인하는 것, 그리고 객체를 문자열로 표현하는 것에 초점을 맞추겠습니다.

Shape를 상속받는 Triangle 클래스를 구현하세요. Triangle은 Shape의 다른 하위클래스들과 같은 메소드들을 포함해야 합니다. 그 외에 주의해야 할 것은

- Triangle은 base와 height를 초기화합니다.
- 밑변base이 3.0이고 높이height가 4.0인 삼각형에 대한 문자열 표현은 다음과 같아야 합니다:

```
Triangle with base 3.0 and height 4.0
```

[주의] 메소드 이름 앞뒤에 두 개의 밑줄표시(__)가 있는 것은 그 메소드가 다른 메소드처럼 사용되거나 오버라이드 될 수 있을지라도 파이썬에서 특별한 의미를 갖고 있다는 것을 기억하세요. 파이썬에서는 이러한 메소드를 호출하는 데에 특정한 관습이 있습니다. 예를 들어 `len(x)`은 파이썬의 `x.__len__()`을 호출합니다.

문제 2

도형들의 집합set을 다루어 보면 어떤 용도가 있을 것 같습니다. 이 문제에서 우리는 도형들의 집합을 다루는 ShapeSet이라는 클래스를 정의합니다. 한 집합set 안의 도형들은 어느 것도 같을 수 없다(연산자 ==로 비교할 때)는 조건을 만족하도록 도형을 집합에 추가하기 위한 메소드 addShape를 생성하세요. `__iter__` 메소드는 집합 안의 도형들을 하나씩 반복하는 반복문을 반환해야 합니다. ShapeSet은 어떠한 반복문이 실행되고 있을 때 변형되지 않는다고 가정할 수 있습니다.

위의 사양들을 만족하도록 다음의 스켈레톤 코드를 완성하세요.

```
1  class ShapeSet:
2      def __init__(self):
3          """
4          Initialize any needed variables
5          """
6          ## TO DO
7      def addShape(self, sh):
8          """
9          Add shape sh to the set; no two shapes in the set may be
10         identical
11         sh: shape to be added
```

```
        """
        ## TO DO
    def __iter__(self):
        """
        Return an iterator that allows you to iterate over the set of
        shapes, one shape at a time
        """
        ## TO DO
    def __str__(self):
        """
        Return the string representation for a set, which consists of
        the string representation of each shape, categorized by type
        (circles, then squares, then triangles)
        """
        ## TO DO
```

ShapeSet의 문자열 표현은 각 도형의 문자열 표현을 포함해야 합니다(출력된다면 각 도형의 문자열 표현은 자체적으로 한 줄에 모두 출력되어야 합니다). 집합의 도형들을 타입으로 분류하세요.

만일 집합에 한 변의 길이기 4인 정사각형과 한 변의 길이가 1인 정사각형, 반지름이 2인 원, 그리고 밑변의 길이가 1이고 높이가 1인 삼각형이 있다면 이 ShapeSet의 문자열 표현은 다음과 같을 것입니다:

```
Circle with radius 2.0
Square with side 4.0
Square with side 1.0
Triangle with base 1.0 and height 1.0
```

문제 3

이제 우리는 각각의 도형들과 여러 도형들의 그룹들을 표현할 수 있기 때문에, 우리는 이것을 코드 외부에서도 사용할 수 있습니다. 예를 들어 우리가 ShapeSet에서 가장 큰 면적을 가진 한 도형이나 여러 도형들을 찾는다고 가정해 봅시다. 이 함수는 우리가 정의한 클래스들의 외부에 있을 것입니다.

ShapeSet에서 가장 큰 면적을 가진 모든 요소들을 포함하는 튜플을 반환하는 함수 findLargest을 만드세요.

```
def findLargest(shapes):
    """
    Returns a tuple containing the elements of ShapeSet with the
      largest area.
    shapes: ShapeSet
    """
    ## TO DO
```

여러분의 코드를 시험하기 위해 여러 개의 서로 다른 ShapeSet들에서 가장 큰 면적의 도형을 찾아보세요. 예를 들어

```
>>> ss = ShapeSet()
>>> ss.addShape(Triangle(1.2,2.5))
>>> ss.addShape(Circle(4))
>>> ss.addShape(Square(3.6))
>>> ss.addShape(Triangle(1.6,6.4))
>>> ss.addShape(Circle(2.2))
>>> largest = findLargest(ss)
>>> largest
(<__main__.Circle object at xxxxx>,)
>>> for e in largest: print(e)
Circle with radius 4.0
```

argest은 집합 안에서 가장 면적이 큰 도형이기 때문에 Circle의 인스턴스를 포함하는 튜플이라는 것에 주의하세요.

어떤 경우에는 가장 큰 면적을 가진 도형이 여러 개일 수 있습니다. 예를 들어 밑변이 4이고 높이가 6인 삼각형은 밑변이 3이고 높이가 8인 삼각형과 면적이 같습니다. 이 경우에는 튜플이 두 도형 모두를 포함해야 합니다.

```
>>> ss = ShapeSet()
>>> ss.addShape(Triangle(3,8))
>>> ss.addShape(Circle(1))
>>> ss.addShape(Triangle(4,6))
>>> largest = findLargest(ss)
>>> largest
(<__main__.Triangle object at xxxxx>, <__main__.Triangle object at xxxxx>)
```

```
>>> for e in largest: print(e)
Triangle with base 3.0 and height 8.0
Triangle with base 4.0 and height 6.0
```

여러분은 튜플 안의 도형은 사실상 ShapeSet에 처음에 추가한 도형이라는 것을 `is` 키워드를 사용하여 확인할 수 있는데, 이것을 객체의 동일성object equality이라고 합니다:

```
>>> t = Triangle(6,6)
>>> c = Circle(1)
>>> ss = ShapeSet()
>>> ss.addShape(t)
>>> ss.addShape(c)
>>> largest = findLargest(ss)
>>> largest
(<__main__.Triangle object at xxxxx>,)
>>> largest[0] is t
True
>>> largest[0] is c
False
```

문제 4

이제 우리는 새로운 ShapeSet을 생성하기 위해, 먼저 여러 개의 도형들을 각각 생성한 후에 하나씩 ShapeSet에 추가해야 합니다. 만약 우리가 굉장히 많은 도형들을 사용하고 싶거나 하나의 프로그램에서 사용한 도형들을 다음에 다시 사용하고 싶은데, 도형을 하나씩 코드로 생성하는 것은 지루하고 불필요합니다.

이 문제를 해결하기 위해 우리는 도형 정보를 하나의 파일에 저장하고 이 파일을 줄마다 읽어 주어진 정보로 ShapeSet을 생성할 수 있습니다.

주어진 파일에서 제공하는 정보에 기초해서 ShapeSet을 생성하고 반환하는 함수 `readShapesFromFile`을 만드세요.

```
def readShapesFromFile(filename):
    """
    Retrieves shape information from the given file.
    Creates and returns a ShapeSet with the shapes found.
    filename: string
    """
```

```
7       ## TO DO
```

파일의 정보는 다음과 같이 저장됩니다:

```
1  circle,4
2  square,3
3  circle,2
4  triangle,4,4
5  square,4
6  circle,4
```

샘플 도형 파일은 shape.txt를 참고하세요.

원과 정사각형은 하나의 인자(각각 변의 길이와 반지름)가 있고, 삼각형은 두 개의 인자(밑변과 높이)가 있는 것에 주의하세요. 여러분은 파일을 읽어 들이면서, 두 가지 경우를 다룰 수 있는 방법을 찾아야 합니다.

파일을 읽는 방법에 대해 더 많은 정보를 알고 싶다면 ps8의 코드를 살펴보세요.

여러분은 ShapeSet이 반환하는 것을 출력해서 코드를 시험해 볼 수 있습니다. 또한 ShapeSet과 다른 함수들을 시험할 수도 있습니다:

```
1  >>> ss = readShapesFromFile("shapes.txt")
2  >>> print ss
3  Circle with radius 4.3
4  Circle with radius 2.0
5  Circle with radius 4.5
6  Circle with radius 3.3
7  Circle with radius 6.0
8  Circle with radius 19.4
9  Square with side 3.0
10 Square with side 4.0
11 Square with side 16.1
12 Square with side 2.7
13 Square with side 10.9
14 Triangle with base 4.2 and height 4.0
15 Triangle with base 3.0 and height 6.6
16 Triangle with base 1.3 and height 4.0
17 Triangle with base 4.0 and height 1.0
18 Triangle with base 2.6 and height 1.2
```

제출 과정

저장. ps9.py 안에 수정한 코드를 작성하세요.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적으세요. 또한 공동 작업한 학우가 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# Problem Set 9
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

다시 한 번 확인하기. 문제 세트를 마친 후에 다시 한 번 확인하세요.

- ps9.py를 실행하고, 에러 없이 동작하는지 확인하세요.
- 문제 4에서 가져와 만들어진 ShapeSet을 확인하세요. 조건을 만족하나요?

```
# shapes.txt

circle,4.3
square,3
circle,2
triangle,4.2,4
square,4
circle,4.5
square,16.1
circle,3.3
triangle,3,6.6
triangle,1.3,4
square,2.7
triangle,4,1
circle,6.0
triangle,2.6,1.2
square,10.9
circle,19.4

# ps9.py
#
# 6.00 Problem Set 9
# Name:
# Collaborators:
# Time:
```

```
26
27 from string import *
28
29 class Shape(object):
30     def area(self):
31         raise AttributeException("Subclasses should override this method.")
32
33 class Square(Shape):
34     def __init__(self, h):
35         """
36         h: length of side of the square
37         """
38         self.side = float(h)
39     def area(self):
40         """
41         Returns area of the square
42         """
43         return self.side**2
44     def __str__(self):
45         return 'Square with side ' + str(self.side)
46     def __eq__(self, other):
47         """
48         Two squares are equal if they have the same dimension.
49         other: object to check for equality
50         """
51         return type(other) == Square and self.side == other.side
52
53 class Circle(Shape):
54     def __init__(self, radius):
55         """
56         radius: radius of the circle
57          """
58         self.radius = float(radius)
59     def area(self):
60         """
61         Returns approximate area of the circle
62         """
63         return 3.14159*(self.radius**2)
64     def __str__(self):
65         return 'Circle with radius ' + str(self.radius)
66     def __eq__(self, other):
67         """
68         Two circles are equal if they have the same radius.
69         other: object to check for equality
70         """
71         return type(other) == Circle and self.radius == other.radius
72 #
73 # Problem 1: Create the Triangle class
74 #
75 ## TO DO: Implement the 'Triangle' class, which also extends 'Shape'.
76
77 #
78 # Problem 2: Create the ShapeSet class
```

```
#
## TO DO: Fill in the following code skeleton according to the
##   specifications.

class ShapeSet:
    def __init__(self):
        """
        Initialize any needed variables
        """
        ## TO DO
    def addShape(self, sh):
        """
        Add shape sh to the set; no two shapes in the set may be
        identical
        sh: shape to be added
        """
        ## TO DO
    def __iter__(self):
        """
        Return an iterator that allows you to iterate over the set of
        shapes, one shape at a time
        """
## TO DO
    def __str__(self):
        """
        Return the string representation for a set, which consists of
        the string representation of each shape, categorized by type
        (circles, then squares, then triangles)
        """
        ## TO DO

#
# Problem 3: Find the largest shapes in a ShapeSet
#
def findLargest(shapes):
    """
    Returns a tuple containing the elements of ShapeSet with the
       largest area.
    shapes: ShapeSet
    """
     ## TO DO

#
# Problem 4: Read shapes from a file into a ShapeSet
#
def readShapesFromFile(filename):
    """
    Retrieves shape information from the given file.
    Creates and returns a ShapeSet with the shapes found.
    filename: string
    """
    ## TO DO
```

문제 세트 10 : OOP로 단어 게임을 다시 했습니다

서문

문제 세트 5와 6에서 했던 단어 게임이 재미있었기를 바랍니다.^^ 그래서 다시 단어 게임으로 돌아왔습니다! 이 문제 세트에서는 6.00 단어 게임의 시각적인 버전에 해당하는 코드를 작성할 것입니다. 혼자 하기도 하고, 두 명이 하기도 하고, 또한 컴퓨터를 상대로 방식을 포함한 코드를 작성해야 합니다.

코드를 작성하면서, 여러분은 단어 게임에 대한 자료와 함수들을 캡슐화encapsulate하고 관리할 클래스를 구현할 것입니다. 또한 이러한 클래스의 인스턴스를 메소드들을 사용하여 처리하는 연습을 할 것입니다. 마지막으로 6.00 단어 게임을 구현하는 클래스들이 그래픽 사용자 인터페이스GUI 모듈과 함께 상호작용 하는 것을 통해 클래스들이 동작하는 것을 확인할 것입니다.

- **과제량**

각각의 문제를 푸는데 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상했던 것보다 더 많은 시간이 걸리는 문제를 내지 않도록 주의할 것입니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 각각의 학생은 개별적으로 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

시작하기

다음의 파일을 동일한 폴더에 다운로드[1] 후 저장하세요.

- ps10.py: 단어 게임을 위한 코드 견본template.
- ps10_test.py: ps10.py에서 사용할 클래스를 위한 스켈레톤 테스트. 이 파일에 테스트를 추가하세요.
- ps10_gui.py: 그래픽 사용자 인터페이스 단어 게임을 위한 코드. 이 파일을 수정하지 마세요.
- words.txt: 문제 세트 5와 6에서 사용한 단어 목록.

문제 1 wxPython 설치

이 문제 세트를 시작하기에 앞서, 파이썬으로 그래픽 사용자 인터페이스GUI를 만들기 위한 툴키트toolkit가 필요합니다. 이 목적에 맞게 우리는 wxPython 툴키트를 사용하겠습니다.

우선, 파이썬이 설치되어있는지 확인하세요. IDLE의 파이썬 쉘 창에서 "Help" 메뉴로 가서 "About IDLE" 을 클릭하면 파이썬 버전을 확인할 수 있습니다.

파이썬이 설치되어 있다면, wxPython을 다운로드하고 설치할 수 있습니다.

• Python 2.x에서 설치

wxPython은 Python 2.x 기반으로 만들어진 툴키트입니다. wxPython 다운로드 페이지에서 해당 버전에 맞는 툴키트를 아래 링크에서 다운로드 받고 설치하면 됩니다.
http://www.wxpython.org/download.php#msw

• Python 3.x에서 설치

Python 3.x에서 wxPython을 설치하는 데에는 여러 방법이 있지만, 가장 쉬운 방법은 pip을 사용하는 것입니다. 우선 pip을 최신 버전으로 업그레이드 합니다. 다음의 코드를 cmd창이나 쉘 환경에서 실행하면 됩니다.

```
python -m pip install --upgrade pip
```

다음으로 pip을 이용해서 wxPython을 설치합니다. 다음의 코드를 cmd창이나 쉘 환경에서 실행하면 됩니다.

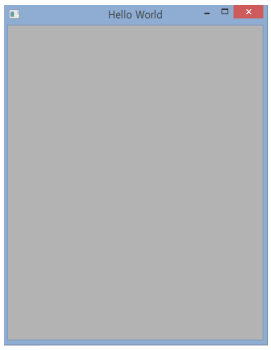
1. [역자 주] 참고문헌: 모든 문제 세트에 필요한 파일은 MIT OCW에서 다운로드 받을 수 있습니다.
http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/assignments/

```
pip install -U --pre -f --trusted-host http://wxpython.org/Phoenix/
snapshot-builds/wxPython_Phoenix-.0.3.dev1820+49a8884-cp34-none-win32.whl
```

이제 wxPython가 정상적으로 설치되었는지 확인하기 위해 "Hello World" 라는 프레임을 만들어 보겠습니다. 파이썬을 실행한 후에 아래 코드를 파이썬 셀에서 입력하세요.

[주의] cmd창에서 파이썬을 실행하기 위해서는 python.exe가 있는 디렉터리를 환경 변수에 추가해야 합니다.

```
>>> import wx
>>> app = wx.App()
>>> frame = wx.Frame(None, -1, "Hello World")
>>> frame.Show()
True
>>> app.MainLoop()
```



위의 코드를 파이썬 쉘에서실행하면 왼쪽과 같은 Hello World 프레임이 나타납니다. 이 과정까지 에러 없이 진행된다면, wxPython 툴키트를 성공적으로 설치한 것입니다.

이 문제에 대해서 제출할 내용은 없습니다. 단지 wxPython 툴키트를 설치하면 됩니다. wxPython의 자세한 내용을 확인하고 싶다면 다음 문서를 참고하세요.

http://wxpython.org/Phoenix/docs/html/index.html

문제 2 Hand를 표현하기

문제 세트 5에서 플레이어의 hand와 관련된 여러 자료 구조와 함수들을 관리했습니다. hand에 쓰이는 자료는 각 문자를 hand안에 있는 그 문자의 빈도수로 매핑하여 딕셔너리로 저장됩니다.

Hand를 관리하기 위해 다양한 함수들을 사용했습니다. `deal_hand()`를 사용하여 새로운 hand를 초기화 하며, `display_hand()`를 사용하여 화면에 hand를 읽을 수 있는 형태로 출력하고, `update_hand()`를 사용하여 hand로부터 word에 있는 글자를 제거했으며 `is_valid_word()`를 사용하여 현재 hand에 있는 글자로 word를 만들 수 있는지 확인했습니다.

이제 우리는 Hand라는 클래스 안에 hand와 관련된 함수들과 자료를 캡슐화 할 것입니다. 우리는 hand의 초기 크기를 인자로 받는(매개변수 initialSize) 생성자 메소드 `__init__()`을 제공

했습니다.

생성자는 두 번째 인자(매개변수 initialHandDict)를 선택적으로 받습니다. 이것은 딕셔너리 표현방법으로 hand를 특정한 글자 세트로 초기화합니다. 이 매개 변수는 테스팅하는 목적으로 사용합니다. 두 번째 인자가 주어지지 않으면, 생성자는 hand를 임의의 글자로 초기화합니다.

클래스는 초기의 hand 크기를 속성 self.initialSize로 저장하며 hand의 딕셔너리 표현방법을 속성 self.handDict로 저장합니다.

이제 hand의 클래스 표현방법과 동일하게 함수의 남은 부분들을 채워 넣으면 됩니다.

조건에 맞게 Hand 클래스 안에 있는 메소드 `update()`, `containsLetters()`, `isEmpty()`, `__eq__()`를 작성하세요. 아래의 메소드 구현은 상대적으로 간결해야 합니다(각 메소드마다 10-20줄 혹은 그보다 간결하게 작성하세요).

문제 세트 5와 6에서 사용한 답을 사용해서 코드를 작성해도 됩니다. 우리는 ps10_test.py 안에 스켈레톤 테스트를 제공했습니다. Hand 클래스를 테스트하고 싶다면 `testHand()`를 사용하세요. Hand 클래스를 적절하게 테스트 하기 위하여 `testHand()`에 추가적인 테스트들을 더해야 합니다.

```
class Hand(object):
    …
    def update(self, word):
        """
        Remove letters in word from this hand.

        word: The word (a string) to remove from the hand
        postcondition: Letters in word are removed from this hand
        """
        # TODO
    def containsLetters(self, letters):
        """
        Test if this hand contains the characters required to make the input
        string (letters)

        returns: True if the hand contains the characters to make up letters,
        False otherwise
        """
        # TODO
    def isEmpty(self):
        """
```

```
        Test if there are any more letters left in this hand.
        returns: True if there are no letters remaining, False otherwise.
        """
        # TODO
    def __eq__(self, other):
        """
        Equality test, for testing purposes

        returns: True if this Hand contains the same number of each letter as
        the other Hand, False otherwise
        """
        # TODO
        …
```

문제 3 플레이어 표현하기

문제 세트 5와 6에서 여러분이 플레이어의 상태를 초기화하고 관리했습니다. 여러분이 플레이어의 점수, 그들에게 남은 시간, 현재 그들의 hand를 추적했습니다.

이 문제 세트에서는 Player 클래스에 이러한 정보를 캡슐화 할 것입니다. 다만 시간은 더 이상 추적하지 않습니다. 이러한 정보에 접근하고 수정하기 위해 메소드 세트를 만들 것입니다. Player 클래스를 위해서 생성자 메소드 `__init__()`을 제공했습니다. 생성자는 매개변수로 ID 숫자idNum(플레이어 1에는 1, 플레이어 2에는 2)와 플레이어의 hand를 표현하기 위한 Hand 객체hand를 받습니다. 그리고 각각을 클래스 속성 self.idNum과 self.hand로 저장합니다.

이 문제에서는 Player의 클래스 표현 조건에 부합하도록 Player 클래스의 남은 메소드들을 작성하면 됩니다.

Player 클래스의 메소드인 `getHand()`, `addPoints()`, `getPoints()`, `getIdNum()`, `__cmp__()`을 작성하세요. 다시 말하지만, 이 메소드들을 매우 간결하게 구현해야 합니다.

ps10_test.py에 스켈레톤 테스트를 제공했습니다. Player 클래스를 테스트하기 위해 `testPlayer()`를 사용하세요. Player 클래스를 정확하게 테스트 하고 싶다면 `testPlayer()` 내부에 테스트들을 추가하세요.

```
1  class Player(object):
2      """
3      General class describing a player.
```

```
    Stores the player's ID number, hand, and score.
    """
        …
    def getHand(self):
        """
        Return this player's hand.

        returns: the Hand object associated with this player.
        """
        # TODO
    def addPoints(self, points):
        """
        Add points to this player's total score.

        points: the number of points to add to this player's score

        postcondition: this player's total score is increased by points
        """
        # TODO
     def getPoints(self):
       """
        Return this player's total score.

        returns: A float specifying this player's score
        """
        # TODO
    def getIdNum(self):
        """
        Return this player's ID number (either 1 for player 1 or
        2 for player 2).

        returns: An integer specifying this player's ID number.
        """
        # TODO
    def __cmp__(self, other):
        """
        Compare players by their scores.

        returns: 1 if this player's score is greater than other player's
        score, -1 if this player's score is less than other player's score,
        and 0 if they're equal.
        """
        # TODO
        …
```

문제 4 컴퓨터 플레이어 표현하기

컴퓨터 플레이어는 Player 클래스의 특수한 한 경우라고 생각할 수 있습니다. 전혀 새로운 컴퓨터 플레이어 클래스를 구현하는 것보다는, 우리는 Player 클래스를 상속받아 ComputerPlayer 클래스를 생성하고 `pickBestWord()` 메소드를 추가할 것입니다. 이 메소드는 hand와 단어 목록(이 메소드의 매개변수들)이 주어졌을 때, 최고의 단어를 선택하는 컴퓨터 플레이어의 알고리즘을 구현합니다.

문제 세트 6에서 구현한 느린 그리디 알고리즘(`pick_best_word`)을 사용해야 합니다. 매개변수 wordlist는 클래스 Wordlist의 인스턴스입니다. 이미 구현되어 여러분에게 제공된 Wordlist 클래스를 이해해보세요. Wordlist 인스턴스에 적용할 수 있는 메소드들을 확실히 이해한다면 코드를 구현하기에 매우 유용하리라 생각됩니다.

ComputerPlayer 클래스의 `pickBestWord()` 메소드를 구현하세요. 이 알고리즘을 다시 구현하면서 이 전에 문제 세트에서 구현한 코드를 가져와 적용하려면, 이 문제 세트의 클래스와 함께 사용할 수 있는 코드로 수정해야 합니다. 컴퓨터의 hand와 단어 목록에서 사용 가능한 단어들에 관하여 무엇인가 하려면 다른 클래스들을 활용해야 합니다.

ps10_test.py는 스켈레톤 테스트를 제공하고 있습니다. ComputerPlayer 클래스를 테스트하기 위해 `testComputerPlayer()`를 사용하세요. ComputerPlayer 클래스를 적절하게 테스트하려면 `testComputerPlayer()` 내부에 테스트들을 추가해야 합니다.

```
class ComputerPlayer(Player):
    """
    A computer player class.
    Does everything a Player does, but can also pick a word using the
    PickBestWord method.
    """
    def pickBestWord(self, wordlist):
        """
        Pick the best word available to the computer player.

        returns: The best word (a string), given the computer player's hand and
        the wordlist
        """
        # TODO
```

문제 5 그래픽 사용자 인터페이스(GUI: Graphical User Interface)

Hand, Player, ComputerPlayer를 구현했다면, 이제 그래픽 사용자 인터페이스GUI를 사용해서 게임을 실행할 수 있어야 합니다.

GUI를 시작할 때, 세 가지 서로 다른 게임 방법들을 선택 사항으로 보여줍니다. 그 중 한 가지를 선택하면 메인 게임 실행 창이 뜰 것이며 즉각적으로 게임의 시작을 도울 것입니다. Solo 방식은 hand를 사용해 당신이 혼자서 게임을 합니다. Vs. Computer 방식은 당신이 먼저 시작하고, 뒤이어 컴퓨터가 시작합니다. Vs. Human 방식은 당신이 먼저 시작하고 뒤이어 친구가 시작합니다. Vs. 방식은 두 플레이어가 같은 hand를 받기 때문에 당신이 게임을 하는 동안 당신의 친구가 보고 있지 않은지 확인하세요!

게임 창에서 텍스트 상자에 단어를 입력하고, Enter 혹은 Submit 버튼을 누르면 됩니다. 창 아래 부분에 있는 상태막대status bar에는 당신의 단어가 채택 되었는지를 보여주며, 채택된 단어는 이력목록상자history list box에 추가됩니다. 왼쪽과 오른쪽에는 각각 플레이어1과 플레이어2의 상태를 볼 수 있습니다. hand를 끝내기 위해서는 모든 글자를 사용하거나 `'.'`를 입력하면 됩니다. 게임을 중지하기 위해서 언제든지 창을 닫아도 됩니다.

컴퓨터와 게임을 할 때에는 메인 창으로 상호 교류할 수 없습니다. 하지만, 컴퓨터 플레이어가 게임을 하며 선택하는 단어를 볼 수는 있습니다.

> `ps10_gui.py`를 열어서 실행하세요. 서로 다른 게임 방식을 테스트해보고 작성한 코드가 모든 방식에 있어서 동작하는지 확인하세요. 이 문제에 대해서는 제출할 내용이 없습니다. 철저하게 코드를 테스트하는 것은 잊지 마세요.

제출 과정

저장. ps10.py 안에 수정한 코드를 작성하세요.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적으세요. 또한 공동 작업한 학우가 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# Problem Set 10
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

다시 한 번 확인하기. 문제 세트를 마친 후에 다시 한 번 확인하세요.

- ps10_gui.py 및 ps10_test.py를 실행하고, 에러 없이 동작하는지 확인하세요.

```
# ps10.py
#
# Backend code for PS10

import random
import string

# Global Constants
VOWELS = 'aeiou'
CONSONANTS = 'bcdfghjklmnpqrstvwxyz'
HAND_SIZE = 30
SCRABBLE_LETTER_VALUES = {
    'a': 1, 'b': 3, 'c': 3, 'd': 2, 'e': 1, 'f': 4, 'g': 2, 'h': 4, 'i': 1,
    'j': 8, 'k': 5, 'l': 1, 'm': 3, 'n': 1, 'o': 1, 'p': 3, 'q': 10, 'r': 1,
    's': 1, 't': 1, 'u': 1, 'v': 4, 'w': 4, 'x': 8, 'y': 4, 'z': 10
}
HUMAN_SOLO = 0
HUMAN_VS_HUMAN = 1
HUMAN_VS_COMP = 2

WORDLIST_FILENAME = "words.txt"

def getFrequencyDict(sequence):
    """
    Given a sequence of letters, convert the sequence to a dictionary of
    letters -> frequencies. Used by containsLetters().

    returns: dictionary of letters -> frequencies
    """
    freq = {}
    for x in sequence:
        freq[x] = freq.get(x,0) + 1
    return freq

def getWordScore(word):
    """
    Computes the score of a word (no bingo bonus is added).

    word: The word to score (a string).
```

```
    returns: score of the word.
    """
   score = 0
   for ch in word:
      score += SCRABBLE_LETTER_VALUES[ch]
   if len(word) == HAND_SIZE:
      score += 50
      return score
#
# Problem 2: Representing a Hand
#

class Hand(object):
    def __init__(self, handSize, initialHandDict = None):
        """
        Initialize a hand.

        handSize: The size of the hand

        postcondition: initializes a hand with random set of initial letters.
        """
        num_vowels = handSize // 3
        if initialHandDict is None:
            initialHandDict = {}
            for i in range(num_vowels):
                x = VOWELS[random.randrange(0,len(VOWELS))]
                initialHandDict[x] = initialHandDict.get(x, 0) + 1
            for i in range(num_vowels, handSize):
                x = CONSONANTS[random.randrange(0,len(CONSONANTS))]
                initialHandDict[x] = initialHandDict.get(x, 0) + 1
        self.initialSize = handSize
        self.handDict = initialHandDict
    def update(self, word):
        """
        Remove letters in word from this hand.

        word: The word (a string) to remove from the hand
        postcondition: Letters in word are removed from this hand
        """
        # TODO
    def containsLetters(self, letters):
        """
        Test if this hand contains the characters required to make the input
        string (letters)

        returns: True if the hand contains the characters to make up letters,
        False otherwise
        """
        # TODO
    def isEmpty(self):
        """
```

```
    Test if there are any more letters left in this hand.

    returns: True if there are no letters remaining, False otherwise.
    """
        # TODO
    def __eq__(self, other):
        """
        Equality test, for testing purposes

        returns: True if this Hand contains the same number of each letter as
        the other Hand, False otherwise
        """
        # TODO
    def __str__(self):
        """
        Represent this hand as a string

        returns: a string representation of this hand
        """
        string = ''
        for letter in self.handDict.keys():
           for j in range(self.handDict[letter]):
              string = string + letter + ''
        return string

#
# Problem 3: Representing a Player
#

class Player(object):
    """
    General class describing a player.
    Stores the player's ID number, hand, and score.
    """
    def __init__(self, idNum, hand):
        """
        Initialize a player instance.

        idNum: integer: 1 for player 1, 2 for player 2.  Used in informational
        displays in the GUI.

        hand: An object of type Hand.

        postcondition: This player object is initialized
        """
        self.points = 0.
        self.idNum = idNum
        self.hand = hand
    def getHand(self):
        """
        Return this player's hand.
```

```
        returns: the Hand object associated with this player.
        """
        # TODO
    def addPoints(self, points):
        """
        Add points to this player's total score.

        points: the number of points to add to this player's score

        postcondition: this player's total score is increased by points
        """
        # TODO
    def getPoints(self):
        """
        Return this player's total score.

        returns: A float specifying this player's score
        """
        # TODO
    def getIdNum(self):
        """
        Return this player's ID number (either 1 for player 1 or
        2 for player 2).

        returns: An integer specifying this player's ID number.
        """
        # TODO
    def __cmp__(self, other):
        """
        Compare players by their scores.
        returns: 1 if this player's score is greater than other player's score,
        -1 if this player's score is less than other player's score, and 0 if
        they're equal.
        """
        # TODO
    def __str__(self):
        """
        Represent this player as a string

        returns: a string representation of this player
        """
        return 'Player %d\n\nScore: %.2f\n' % \
            (self.getIdNum(), self.getPoints())

#
# Problem 4: Representing a Computer Player
#

class ComputerPlayer(Player):
    """
    A computer player class.
```

```
193     Does everything a Player does, but can also pick a word using the
194     PickBestWord method.
195     """
196     def pickBestWord(self, wordlist):
197         """
198         Pick the best word available to the computer player.
199 
200         returns: The best word (a string), given the computer player's hand and
201         the wordlist
202         """
203         # TODO
204     def playHand(self, callback, wordlist):
205         """
206         Play a hand completely by passing chosen words to the callback
207         function.
208         """
209         while callback(self.pickBestWord(wordlist)): pass
210 
211 class HumanPlayer(Player):
212     """
213     A Human player class.
214     No methods are needed because everything is taken care of by the GUI.
215     """
216 
217 class Wordlist(object):
218     """
219     A word list.
220     """
221     def __init__(self):
222         """
223         Initializes a Wordlist object.
224 
225         postcondition: words are read in from a file (WORDLIST_FILENAME, a
226         global constant) and stored as a list.
227         """
228         inputFile = open(WORDLIST_FILENAME)
229         try:
230             self.wordlist = []
231             for line in inputFile:
232                 self.wordlist.append(line.strip().lower())
233         finally:
234             inputFile.close()
235     def containsWord(self, word):
236         """
237         Test whether this wordlist includes word
238 
239         word: The word to check (a string)
240 
241         returns: True if word is in this Wordlist, False if word is not in
242         Wordlist
243         """
```

```
        return word in self.wordlist
    def getList(self):
        return self.wordlist

class EndHand(Exception): pass

class Game(object):
    """
    Stores the state needed to play a round of the word game.
    """
    def __init__(self, mode, wordlist):
        """
        Initializes a game.

        mode: Can be one of three constant values - HUMAN_SOLO, HUMAN_VS_COMP,
        and HUMAN_VS_HUMAN
        postcondition: Initializes the players nd their hands.
        """
        hand = Hand(HAND_SIZE)
        hand2 = Hand(HAND_SIZE, hand.handDict.copy())
        if mode == HUMAN_SOLO:
            self.players = [HumanPlayer(1, hand)]
        elif mode == HUMAN_VS_COMP:
            self.players = [HumanPlayer(1, hand),
                        ComputerPlayer(2, hand2)]
        elif mode == HUMAN_VS_HUMAN:
            self.players = [HumanPlayer(1, hand),
                        HumanPlayer(2, hand2)]
        self.playerIndex = 0
        self.wordlist = wordlist
    def getCurrentPlayer(self):
        """
        Gets the Player object corresponding to the active player.

        returns: The active Player object.
        """
        return self.players[self.playerIndex]
    def nextPlayer(self):
        """
        Changes the game state so that the next player is the active player.

        postcondition: playerIndex is incremented
        """
        if self.playerIndex + 1 < len(self.players):
           self.playerIndex = self.playerIndex + 1
           return True
        else:
           return False
    def gameOver(self):
        """
        Determines if the game is over
```

```
295         returns: True if the playerIndex >= the number of players, False
296         otherwise
297         """
298         return self.playerIndex >= len(self.players)
299     def tryWord(self, word):
300         if word == '.':
301             raise EndHand()
302         player = self.getCurrentPlayer()
303         hand = player.getHand()
304         if self.wordlist.containsWord(word) and hand.containsLetters(word):
305             points = getWordScore(word)
306             player.addPoints(points)
307             hand.update(word)
308             if hand.isEmpty():
309                 raise EndHand()
310             return points
311         else:
312             return None
313     def getWinner(self):
314         return max(self.players)
315     def getNumPlayers(self):
316         return len(self.players)
317     def isTie(self):
318         return len(self.players) > 1 and \
319               self.players[0].getPoints() == self.players[1].getPoints()
320     def __str__(self):
321         """
322         Convert this game object to a string
323 
324         returns: the concatenation of the string representation of the players
325         """
326         string = ''
327         for player in self.players:
328             string = string + str(player)
329         return string
```

문제 세트 11 : 로봇 시뮬레이션

서문

이 문제 세트에서는 시뮬레이션을 디자인하고 클래스를 사용하는 프로그램을 구현하는 연습을 합니다.

- **과제량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상했던 것보다 더 많은 시간이 걸리는 문제를 내지 않도록 주의할 것입니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 각각의 학생은 개별적으로 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

시작하기

다음의 파일을 동일한 경로에 다운로드 받고 저장하세요.

- ps11.py: 코드를 작성할 스켈레톤
- ps11_visualize.py: 여러분에게 제공하는 훌륭한 라이브러리(뒤에 설명하겠습니다)

pylab 설치하기

이 문제 세트의 문제 4와 문제 6에서 그래프를 그려야 하기 때문에 몇 라이브러리들을 설치해야 합니다. 그런데 파이썬을 설치할 때 여러 회사들이 파이썬과 여러 패키지를 한 번에 묶어서 설치할 수 있도록 배포판을 많이 사용합니다. 대표적인 배포판에는 Anaconda, Canopy,

WinPython, Python(x, y)와 같은 것들이 있습니다. 이런 배포판으로 파이썬을 설치했다면 다음과 같은 과정이 필요 없을 것입니다. 이런 경우 다른 단락인 [시뮬레이션 개요]로 바로 가시기 바랍니다.

- **Windows에서 pylab 설치하기**

  1. 파이썬의 버전이 Python 3.x가 맞는지 확인하세요 IDLE에서 Help → About IDLE를 들어가면 확인할 수 있습니다.
  2. numpy를 다운로드 받고 설치하세요.
  3. matplotlib을 다운로드 받고 설치하세요.

- **Mac OS X에서 pylab 설치하기**

  1. 파이썬의 버전이 Python 3.x이 맞는지 확인하세요. IDLE에서 Help → About IDLE를 들어가면 볼 수 있습니다.
  2. python-dateutil, pytz, numpy를 다운로드 받고 설치하세요.
  3. matplotlib을 다운로드 받고 설치하세요.

- **Linux ubuntu에서 pylab 설치하기**

Linux의 다른 버전마다 약간의 차이가 있을 수 있는데, 그런 경우 여러분이 스스로 해결해보세요. 다음은 참고사항입니다. Linux에서 작업을 하고 있다면, 명령 프롬프트 창에 사용하고 있는 쉘shell은 bash 쉘이어야 합니다. echo $SHELL이라고 입력해서 확인할 수 있으며 bash를 포함하는 문자열을 볼 수 있습니다(특히 csh는 아니어야 합니다). 만약 쉘이 bash가 아니라면, bash를 실행하세요.

파이썬 2.x일 경우,

```
$ sudo apt-get install python-numpy
$ sudo apt-get install python-scipy
$ sudo apt-get install python-matplotlib
$ sudo apt-get install idle
```

파이썬 3.x일 경우,

```
$ sudo apt-get install python3-numpy
$ sudo apt-get install python3-scipy
```

```
3  $ sudo apt-get install python3-matplotlib
4  $ sudo apt-get install idle3
```

다음 명령어를 사용해서 IDLE를 파이썬 2.x 혹은 3.x에서 실행할 수 있습니다.

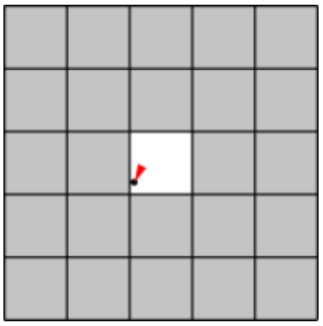
```
    $ idle &
혹은
    $ idle3 &
```

시뮬레이션 개요

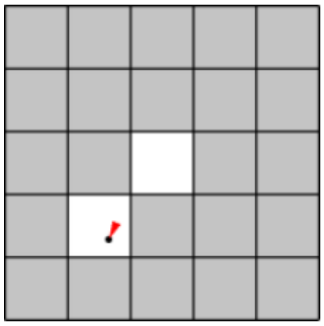
iRobot은 Roomba 로봇 진공청소기(다음 링크를 통해 현재 판매되고 있는 로봇청소기 모델 중 하나를 확인하세요. http://store.irobot.com/home/index.jsp)을 판매하는 MIT 졸업생과 교수들이 시작한 회사입니다. Roomba 로봇은 집안을 돌아다니며, 자기가 지나간 공간을 청소합니다. 여러분은 Roomba와 같은 로봇들이 방 하나와 마루를 청소하는 데 시간이 얼마나 걸릴 지 시뮬레이션을 설계할 것입니다.

다음은 5 X 5 크기의 방에서 청소하는 로봇의 움직임을 간소화한 모델인데, 우리가 시뮬레이션할 시스템에 대한 직관을 줄 수 있을 겁니다.

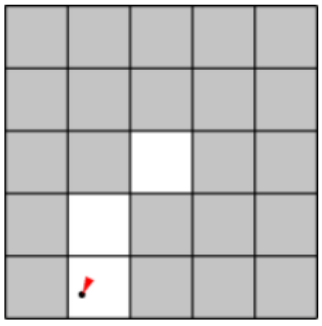
로봇은 방 안의 임의의 장소에서 청소를 시작하며 임의의 방향으로 움직이며 청소를 합니다. 아래의 그림에서 검은 점으로 로봇의 위치를 표시하며, 빨강 화살촉으로 로봇의 방향을 표시합니다.



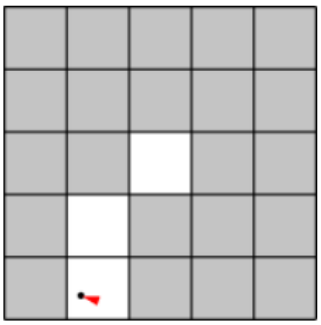
시간 t = 0: 로봇은 위치 (2.1, 2.2)에서 각도 205도("북쪽"을 기준으로 시계방향으로 측정합니다)로 시작합니다. 로봇이 현재 있는 타일은 청소되었습니다.



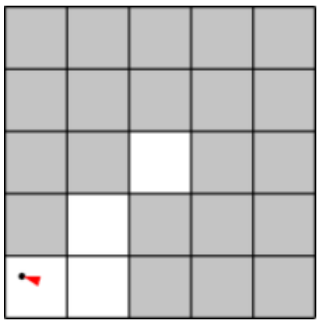
t = 1: 로봇은 마주하고 있는 방향으로 한 단위 움직입니다. 이제 위치 (1.7, 1.3)로 움직이며 다른 타일을 청소합니다.



t = 2: 로봇은 같은 방향으로 한 단위 움직입니다. (북쪽으로부터 205도의 각도로) 위치 (1.2, 0.4)에 도착하고 다른 타일을 청소합니다.



t = 3: 로봇 앞에 벽이 있기에 같은 방향으로 한 단위 움직일 수 없습니다. 따라서 임의의 방향을 바라봅니다. 여기서는 287도를 바라보는군요.



t = 4: 로봇은 위치 (0.3, 0.7)로 한 단위 이동하고 그 곳에 있는 타일을 청소합니다.

시뮬레이션 상세정보

다음은 시뮬레이션 모델에 대한 추가 상세정보입니다. 신중하게 읽어보세요.

- **여러 로봇** 일반적으로 방에 N 〉 0 로봇들이 있습니다. N은 주어지는 숫자입니다. 모델을 단순화하기 위해, 로봇은 점들이며 서로를 통과할 수 있고, 방해 없이 같은 장소를 점유할 수 있다고 합시다.

- **방** 방은 너비 w와 높이 h를 갖는 직사각형 형태입니다. 너비와 높이 모두 주어지는 숫자입니다. 시작할 때는 방 바닥 전체가 더럽습니다. 로봇은 방의 벽을 통과할 수 없습니다. 로봇은 방 밖으로 나갈 수 없습니다.

- **로봇 움직임의 규칙**
  - 각 로봇은 방 안에 위치가 있습니다. 우리는 그 지점들을 좌표 (x, y)로 표현할 것입니다. 각각은 실수이며 0 〈= x 〈 w, 0 〈= y 〈 h의 조건을 만족합니다. 우리의 프로그램에서는 Position 클래스의 인스턴스를 사용하여 이러한 좌표를 저장할 것입니다.
  - 로봇은 방향을 갖고 움직입니다. 우리는 이 방향을 정수 d로 표현할 것입니다. 0 〈= d 〈 360을 만족하며 북(12시)을 기준으로 시계방향 각도를 의미합니다.
  - 모든 로봇들은 같은 속도 s로 움직입니다. 속도는 주어지며 시뮬레이션을 하는 동안 일정합니다. 모든 시간-단계에서 로봇은 그 방향으로 s 단위들만큼 움직입니다.
  - 로봇이 벽에 부딪치게 되면 임의의 값으로 새 방향이 설정됩니다. 로봇은 그 방향으

로 계속해서 나아가며 다른 벽에 부딪칩니다.

- **타일** 로봇(들)에 의해서 청소된 바닥 부분들을 추적하며 기록해 둘 필요가 있습니다. 우리는 방을 1 X 1 크기의 타일로 분할할 것입니다. (w * h개의 타일이 있겠죠) 로봇이 타일의 어느 곳에 위치하면, 각 타일이 청소되었다고 가정합니다. (위의 그림과 동일합니다) 관례상 정수의 쌍으로 타일을 표시하겠습니다: (0, 0), (0, 1), … , (0, h-1), (1, 0), (1, 1), … , (w-1, h-1).

- **종료** 지정된 만큼의 타일이 청소되면 시뮬레이션은 끝납니다.

만약 위 시뮬레이션 사양들 중에 어떤 부분이 모호하게 느껴진다면, 여러분 스스로 여러분의 프로그램이나 모델이 어떻게 동작해야 할 지에 대한 합리적인 결정을 내리세요. 그리고, 코드를 작성할 때, 그 결정에 대한 것도 함께 작성하세요.

부분 I : RectangularRoom 그리고 BaseRobot 클래스

여러분은 각 로봇의 위치와 방향뿐만 아니라 방의 어느 부분이 청소되었는지 두 개의 클래스를 사용하여 추적하면서 저장할 필요가 있습니다.

ps11.py 파일 안에, 다음 두 클래스에 대한 스켈레톤 코드를 제공합니다. 문제 1을 해결할 때에 여기에 코드를 작성하면 됩니다.

- RectangularRoom: 청소 되어야 할 공간과 청소된 타일들을 추적합니다.
- BaseRobot: 로봇의 위치와 방향을 저장합니다.

또한 다음 클래스들을 완전히 구현하여 여러분에게 제공합니다.

- Position: 방에 있는 로봇의 x 축과 y 축에 대한 정보를 저장합니다.

시작하기 전에 ps11.py 를 신중하게 읽으세요. 그래서, 제공된 코드가 무엇을 할 수 있는지 완전히 이해하세요.

문제 1

주어진 사양에 따라 ps11.py 파일에 있는 메소드들을 구현해서 `RectangularRoom`과 `BaseRobot` 클래스들을 완성하세요.

RectangularRoom 클래스를 구현하기 위해 어떠한 필드들을 사용할 것인지 그리고 다음과 같은 연산들을 어떻게 구현할 것인지 결정해야 할 필요가 있을 것입니다.

- 객체를 초기화하기
- 로봇이 주어진 위치로 이동할 때에 그에 해당하는 타일이 청소된 것을 표시하기
- 주어진 타일이 청소되었는지 알아내기
- 방 안에 타일이 얼마나 있는지 알아내기
- 방 안에 얼마나 많은 타일이 청소되었는지 알아내기
- 방 안의 임의의 한 위치 구하기
- 주어진 위치가 방 안에 있는지 알아내기

BaseRobot 클래스를 구현하기 위해 어떠한 필드들을 사용할 것인지 그리고 다음과 같은 연산들을 어떻게 구현할 것인지 결정해야 할 필요가 있을 것입니다.

- 객체를 초기화하기
- 로봇의 위치에 접근하기
- 로봇의 방향에 접근하기
- 로봇의 위치를 설정하기
- 로봇의 방향을 설정하기

(이 문제가 많은 부분으로 구성되어 있을지라도, 자료를 어떻게 표현할지 결정했다면 그다지 오래 걸리지 않을 것입니다. 합리적인 표현방법으로, 대부분의 메소드들은 코드 한 줄이면 충분할 것입니다.)

- **부분 II : 시뮬레이터를 만들고 사용하기**

문제 2

각 로봇이 방에서 어떻게 움직여야 하는지 로봇에게 지시하는 코드가 필요합니다. 이 부분은 updatePositionAndClean 메소드에 들어갑니다.

대게 우리는 로봇의 모든 메소드들을 하나의 클래스에 넣는 것을 고려할 것입니다. 하지만 이 문제 세트 뒤 부분에서 우리는 서로 다른 움직임 전략들을 가지고 있는 여러 로봇들을

고려할 것인데, 이 부분은 같은 인터페이스를 사용하는 서로 다른 클래스들로 구현할 것입니다. 이러한 클래스들은 updatePositionAndClean에 대해 각자 나름대로 구현할 것이지만 대부분 기존의 로봇과 유사할 겁니다. 따라서 우리는 코드가 중복되는 것을 방지하기 위해 상속inheritance을 사용할 것입니다.

이미 우리는 로봇 코드를 두 개의 클래스로 나누어 재정의하였습니다. 위에서 완성한 BaseRobot 클래스(일반적인 로봇 코드를 포함하고 있음)와 그로부터 상속받은 Robot 클래스(자신만의 움직임 전략을 포함하고 있음)입니다.

로봇의 updatePositionAndClean 메소드를 완성하세요. **이는 단 하나의 시간-단계 뒤에 로봇의 움직임을 시뮬레이션 하기 위한 메소드입니다**(위에서 설명한 동적 시뮬레이션과 동일합니다).

```
class Robot(BaseRobot):
    """
    A Robot is a BaseRobot with the standard movement strategy.

    At each time-step, a Robot attempts to move in its current
    direction; when it hits a wall, it chooses a new direction
    randomly.
    """
    def updatePositionAndClean(self):
        """
        Simulate the passage of a single time-step.

        Move the robot to a new position and mark the tile it is on as having
        been cleaned.
        """
        # TODO: Your code goes here
```

힌트: Position 클래스에 있는 getNewPosition 함수를 유용하게 사용할 수 있습니다.

문제 3

이 문제에서는 로봇 시뮬레이션을 완전히 동작하게 하는 코드를 작성합니다.

한 번 실행할 마다, 그 목적은 방의 지정된 부분만큼 청소되는데 몇 번의 시간-단계가 필요한지에 대한 자료를 모으는 것입니다. 다음 함수를 구현하세요.

```
def runSimulation(num_robots, speed, width, height, min_coverage, num_trials,
                  robot_type, visualize):
    """
    Runs NUM_TRIALS trials of the simulation and returns a list of
    lists, one per trial. The list for a trial has an element for each
    timestep of that trial, the value of which is the percentage of
    the room that is clean after that timestep. Each trial stops when
    MIN_COVERAGE of the room is clean.

    The simulation is run with NUM_ROBOTS robots of type ROBOT_TYPE,
    each with speed SPEED, in a room of dimensions WIDTH x HEIGHT.

    Visualization is turned on when boolean VISUALIZE is set to True.

    num_robots: an int (num_robots > 0)
    speed: a float (speed > 0)
    width: an int (width > 0)
    height: an int (height > 0)
    min_coverage: a float (0 <= min_coverage <= 1.0)
    num_trials: an int (num_trials > 0)
    robot_type: class of robot to be instantiated (e.g. Robot or
                RandomWalkRobot)
    visualize: a boolean (True to turn on visualization)
    """
    # TODO: Your code goes here
```

처음 여섯 개의 매개변수는 따로 설명이 필요 없습니다. 당분간 `robot_type` 매개변수에 Robot을 넘기세요. 다음과 같이 말입니다.

```
avg = runSimulation(10, 1.0, 15, 20, 0.8, 30, Robot, False)
```

그리고 `runSimulation`안에서 로봇을 생성할 때 마다 Robot(...) 대신에 robot_type(...)을 사용해야 합니다(문제 5에서 다루게 될 터인데, 다른 로봇으로 구현들을 실행할 때에 시뮬레이션을 용이하게 적용할 수 있습니다).

현재로서는 visualize 매개변수를 False로 지정하세요. 이 문제 뒤에서 작성할 visualization 코드를 추가한 후에 True로 바꿀 수 있습니다.

도움 함수가 필요하다면 여러분이 작성하여 얼마든지 사용하세요. 하지만 제출하는 파일에는 이러한 도움 함수와 지정된 클래스 및 함수를 제외하고 다른 코드를 포함할 수 없습니다.

참고로 다음은 방을 청소하는 데 필요한 대략적인 시간입니다. 이 시간은 로봇이 1.0의 속

도로 움직인다고 가정했을 때 입니다.

- 5 X 5 크기의 방을 완전히 청소하기 위해 한 로봇당 약 150 clock tick[1]의 시간이 필요합니다.
- 10 X 10 크기의 방을 75% 청소하기 위해 한 로봇당 약 190 clock tick의 시간이 필요합니다.
- 10 X 10 크기의 방을 90% 청소하기 위해 한 로봇당 약 310 clock tick의 시간이 필요합니다.
- 20 X 20 크기의 방을 완전히 청소하기 위해 한 로봇당 약 3250 clock tick의 시간이 필요합니다.

(이것들을 결정에 도움을 주는 지침 정도로 생각하세요. 어떻게 구현하는 지에 따라서 여기의 결과와 다른 시간을 얻을 겁니다.)

속도를 1.0과 다르게 하여 시뮬레이션의 결과물을 확인해야 합니다. 이것을 하는 한 가지 방법으로, 위의 시험 사례를 사용해서 속도를 변경하고 결과가 합리적인지 확인할 수 있습니다.

• 로봇을 시각화하기

로봇이 방을 청소하면서 나타나는 애니메이션animation을 발생시키는 코드를 제공했습니다. 이러한 애니메이션은 로봇이 올바르게 동작하지 않을 때에 시각적으로 무엇이 문제인지 결정하는 데 도움을 줍니다.

ps11_visualize.py를 다운로드 후에 ps11.py과 같은 경로에 저장하세요. ps11.py 맨 위에 다음 코드를 추가하세요.

```
import ps11_visualize
```

다음은 어떻게 시각화를 하는지에 대한 설명입니다.

1. 시뮬레이션 실행을 시작할 때에, 애니메이션을 시작하기 위해 다음을 하세요.

```
anim = ps11_visualize.RobotVisualization(num_robots, width, height)
```

1. [역자 주] PC에서 clock tick은 66MHz로 동작하는 메인 시스템의 클럭과 관련이 있습니다. 이것은 1초에 6,600만 clock tick이 있다는 것을 의미합니다. 현대 CPU는 매우 빠르게 동작하기 때문에(약 3GHz) CPU는 한 clock tick 동안 여러 명령을 수행할 수 있습니다. http://www.webopedia.com/TERM/C/clock_tick.html에서 가져옴.

(실행에 적합한 매개변수를 넘겨 주어야 합니다.) 위의 코드는 애니메이션을 보여주고 방의 형태를 그리기 위한 새 창을 열어 줍니다.

2. 그 후에, 각 시간-단계마다 애니메이션의 새로운 프레임을 그리기 위해 다음을 실행하세요.

```
anim.update(room, robots)
```

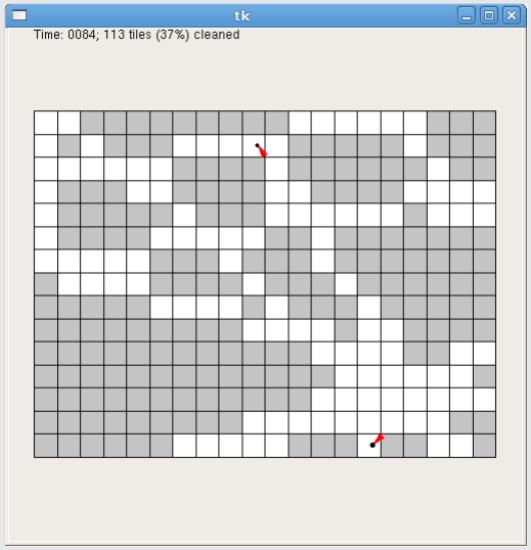
방의 현재 상태를 알려주는 RectangularRoom 객체와 방에 있는 로봇의 목록을 넘겨주세요.

3. 실험이 끝나면 다음 메소드를 호출하세요.

```
anim.done()
```

지금 시점에서 프로그램은 다음 실행을 시작하기 전에 사용자가 애니메이션 창을 닫기를 기다린다는 것을 기억하세요.

애니메이션의 결과는 다음과 같습니다.



시각화 코드는 애니메이션이 너무 빠르게 지나가지 않도록 시뮬레이션의 속도를 늦춰줍니다(디폴트 값으로 1초에 5시간-단계를 보여줍니다). 물론 한 번에 여러 번 실행을 하고자 한다면 애니메이션 코드 실행을 생략하고 싶을 겁니다(예를 들어, 전체 시뮬레이션을 실행한다면 더욱 그러하겠죠).

시뮬레이션을 디버깅하는 목적으로 애니메이션을 더욱 느리게 할 수도 있습니다. RobotVisualization을 다음과 같이 조금 변경하여 호출하면 됩니다.

```
anim = ps11_visualize.RobotVisualization(num_robots, width, height, delay)
```

매개변수 delay는 각 프레임마다 프로그램이 몇 초간 멈춰야 하는지 명시해줍니다. 디폴트 값은 0.2입니다(초당 5프레임을 의미합니다). 이 값을 증가시킴으로써 애니메이션을 더욱 느리게 할 수 있습니다.

애니메이션을 보기 위해 runSimulation의 visualize 매개변수를 True로 설정하는 것을 잊지 마세요.

문제 4

이제 시뮬레이션을 사용하여 로봇의 수행능력에 대한 질문들에 답하세요.

아래에 질문에 대해서 pylab을 사용하여 그래프를 만들어내는 코드를 작성하세요. ps11.py에 해당하는 스켈레톤 함수들 안에 코드를 입력하세요(각각 showPlot1, showPlot2, showPlot3, showPlot4로 하면 됩니다).

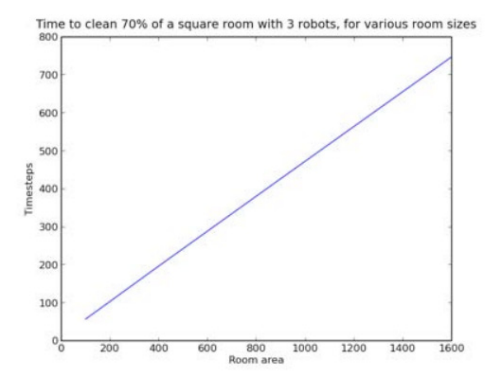
각 그래프는 제목, 각 축에 대해 설명하는 라벨, 범례(해당한다면)가 있어야 합니다. 로봇이 속도 1.0으로 움직인다고 가정하세요.

1. 한 로봇이 다음 유형의 방(5 X 5, 10 X 10, 15 X 15, 20 X 20, 25 X 25)을 75% 청소하는데 시간이 얼마나 걸리나요? 각 방의 면적에 해당하는 평균 시간을 Y축으로 하는 그래프를 그리세요.
2. 로봇 1-10개 각각에 대해서 25 X 25 크기의 방을 75% 청소하는데 시간이 얼마나 걸리나요? 각 방의 면적에 해당하는 평균 시간을 Y축으로 하는 그래프를 그리세요.
3. 로봇 2개로 다음 유형의 방(20 X 20, 25 X 16, 40 X 10, 50 X 8, 80 X 5, 100 X 4)을 75% 청소하는데 시간이 얼마나 걸리나요? (각각의 방이 같은 면적입니다) 각 너비 X 높이에 해당하는 평균 시간을 Y축으로 하는 그래프를 그리세요.
4. min_coverage의 변화에 따라 25 X 25 크기의 방을 청소하는데 시간이 얼마나 걸리나요? 청소된 비율에 각 로봇 1-5개에 해당하는 평균 시간을 Y축으로 하는 그래프를 그리세요. 곡선이 여러 개 포함되어야 합니다.

여러 실행 횟수들을 따라 실험하세요. 그래프를 그릴 때, 신뢰할 수 있는 결과물을 만들어 내는 실행 횟수를 사용하세요.

힌트: 그래프 1-3그리기 위해서 runSimulation의 반환 값인 리스트들의 리스트 안에 있는 리스트들의 평균 길이를 계산하는 도움 함수를 사용한다면 유용할 것입니다. 그래프 4에 대해서 우리가 제공한 computeMeans 함수를 유용하게 사용할 수 있습니다.

다음은 그래프의 좋은 예시입니다.



문제 5

iRobot은 새로운 로봇 디자인을 테스팅하고 있습니다. 새롭게 제안된 로봇은 **매 시간-단계 다음에** 무작위로 방향을 바꿉니다. 이전까지의 로봇이 벽에 부딪칠 때에만 방향을 바꾸는 것과 다른 부분 입니다. 이런 변화가 방을 청소하는데 걸리는 시간에 어떤 영향을 주는지 알기 위한 시뮬레이션을 디자인하는 요청이 여러분에게 왔습니다.

Robot처럼 BaseRobot에서 상속을 받지만 새로운 움직임 전략을 구현하는 RandomWalkRobot 클래스를 작성하세요. RandomWalkRobot은 Robot과 동일한 인터페이스를 가져야 합니다.

새로운 클래스를 시험하세요. 새로운 RandomWalkRobot 구현한대로 로봇이 정상적으로 동작하는지 확인하기 위해 실행을 일회만 해보세요. 로봇의 움직임에 대해 만족한다면, 매개변수로 Robot 대신 RandomWalkRobot을 사용하여 runSimulation을 호출하세요.

문제 6

두 가지 유형의 로봇의 수행 능력을 비교하는데 적합한 그래프(각자 디자인 해보세요)를 그리세요. 코드를 `showPlot5()`에 추가하세요. 항상 그렇듯이, 그래프는 제목, 축 라벨, 범례(해당한다면)가 있어야 합니다.

showPlot5의 주석 안에 두 종류의 로봇을 간단하게 비교하여 기술하세요.

제출 과정

저장. ps11.py 안에 수정한 코드를 작성하세요. 지정된 클래스와 함수 및 도움 함수를 제외한 코드를 사용하면 안됩니다.

테스트. 문법 에러가 없는지 다시 한 번 파일을 실행하세요. 그래프를 그리는 함수들을 실행해서 함수를 그리는지 확인하세요. runSimulation을 테스트하고 Robot과 RandomWalkRobot 클래스 모두에 대해 동작하는지 확인하세요(코드를 수정하면서 동작하지 못하게 만드는 경우가 종종 있습니다. 따라서 제출 전에 다시 한 번 확인해야겠죠!).

시간과 공동 작업 정보. 파일의 시작부분에 주석으로 해당 문제 세트를 해결하는데 대략적으로 몇 시간이 걸렸는지 적으세요. 또한 공동 작업한 학우가 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# Problem Set 11
# Name: Jane Lee
# Collaborators: John Doe
# Time: 3:30
... your code goes here ...
```

```
# ps11.py
#
# Problem Set 11: Simulating robots
# Name:
# Collaborators:
# Time:

import math
# === Provided classes

class Position(object):
```

```
    """
    A Position represents a location in a two-dimensional room.
    """
    def __init__(self, x, y):
        """
        Initializes a position with coordinates (x, y).

        x: a real number indicating the x-coordinate
        y: a real number indicating the y-coordinate
        """
        self.x = x
        self.y = y
    def getX(self):
        return self.x
    def getY(self):
        return self.y
    def getNewPosition(self, angle, speed):
        """
        Computes and returns the new Position after a single clock-tick has
        passed, with this object as the current position, and with the
        specified angle and speed.

        Does NOT test whether the returned position fits inside the room.

        angle: integer representing angle in degrees, 0 <= angle < 360
        speed: positive float representing speed

        Returns: a Position object representing the new position.
        """
        old_x, old_y = self.getX(), self.getY()
        # Compute the change in position
        delta_y = speed * math.cos(math.radians(angle))
        delta_x = speed * math.sin(math.radians(angle))
        # Add that to the existing position
        new_x = old_x + delta_x
        new_y = old_y + delta_y
        return Position(new_x, new_y)

# === Problems 1 and 2
class RectangularRoom(object):
    """
    A RectangularRoom represents a rectangular region containing clean or dirty
    tiles.

    A room has a width and a height and contains (width * height) tiles. At any
    particular time, each of these tiles is either clean or dirty.
    """
    def __init__(self, width, height):
        """
        Initializes a rectangular room with the specified width and height.
        Initially, no tiles in the room have been cleaned.
```

```
63
64          width: an integer > 0
65          height: an integer > 0
66          """
67          # TODO: Your code goes here
68      def cleanTileAtPosition(self, pos):
69          """
70          Mark the tile under the position POS as cleaned.
71          Assumes that POS represents a valid position inside this room.
72
73          pos: a Position
74          """
75          # TODO: Your code goes here
76      def isTileCleaned(self, m, n):
77          """
78          Return True if the tile (m, n) has been cleaned.
79
80          Assumes that (m, n) represents a valid tile inside the room.
81
82          m: an integer
83          n: an integer
84          returns: True if (m, n) is cleaned, False otherwise
85          """
86          # TODO: Your code goes here
87      def getNumTiles(self):
88          """
89          Return the total number of tiles in the room.
90
91          returns: an integer
92          """
93          # TODO: Your code goes here
94      def getNumCleanedTiles(self):
95          """
96          Return the total number of clean tiles in the room.
97
98          returns: an integer
99          """
100         # TODO: Your code goes here
101     def getRandomPosition(self):
102         """
103         Return a random position inside the room.
104
105         returns: a Position object.
106         """
107         # TODO: Your code goes here
108     def isPositionInRoom(self, pos):
109         """
110         Return True if POS is inside the room.
111
112         pos: a Position object.
113         returns: True if POS is in the room, False otherwise.
```

```
        """
        # TODO: Your code goes here


class BaseRobot(object):
    """
    Represents a robot cleaning a particular room.

    At all times the robot has a particular position and direction in
    the room.  The robot also has a fixed speed.

    Subclasses of BaseRobot should provide movement strategies by
    implementing updatePositionAndClean(), which simulates a single
    time-step.
    """
    def __init__(self, room, speed):
        """
        Initializes a Robot with the given speed in the specified
        room. The robot initially has a random direction d and a
        random position p in the room.

        The direction d is an integer satisfying 0 <= d < 360; it
        specifies an angle in degrees.
        p is a Position object giving the robot's position.

        room:  a RectangularRoom object.
        speed: a float (speed > 0)
        """
        # TODO: Your code goes here
    def getRobotPosition(self):
        """
        Return the position of the robot.

        returns: a Position object giving the robot's position.
        """
        # TODO: Your code goes here
    def getRobotDirection(self):
        """
        Return the direction of the robot.

        returns: an integer d giving the direction of the robot as an angle in
        degrees, 0 <= d < 360.
        """
        # TODO: Your code goes here
    def setRobotPosition(self, position):
        """
        Set the position of the robot to POSITION.

        position: a Position object.
        """
        # TODO: Your code goes here
```

```
    def setRobotDirection(self, direction):
        """
        Set the direction of the robot to DIRECTION.

        direction: integer representing an angle in degrees
        """
        # TODO: Your code goes here


class Robot(BaseRobot):
    """
    A Robot is a BaseRobot with the standard movement strategy.

    At each time-step, a Robot attempts to move in its current
    direction; when it hits a wall, it chooses a new direction
    randomly.
    """
    def updatePositionAndClean(self):
        """
        Simulate the passage of a single time-step.

        Move the robot to a new position and mark the tile it is on as having
        been cleaned.
        """
        # TODO: Your code goes here


# === Problem 3

def runSimulation(num_robots, speed, width, height, min_coverage, num_trials,
                  robot_type, visualize):
    """
    Runs NUM_TRIALS trials of the simulation and returns a list of
    lists, one per trial. The list for a trial has an element for each
    timestep of that trial, the value of which is the percentage of
    the room that is clean after that timestep. Each trial stops when
    MIN_COVERAGE of the room is clean.

    The simulation is run with NUM_ROBOTS robots of type ROBOT_TYPE,
    each with speed SPEED, in a room of dimensions WIDTH x HEIGHT.

    Visualization is turned on when boolean VISUALIZE is set to True.

    num_robots: an int (num_robots > 0)
    speed: a float (speed > 0)
    width: an int (width > 0)
    height: an int (height > 0)
    min_coverage: a float (0 <= min_coverage <= 1.0)
    num_trials: an int (num_trials > 0)
    robot_type: class of robot to be instantiated (e.g. Robot or
                RandomWalkRobot)
```

```
    visualize: a boolean (True to turn on visualization)
    """
    # TODO: Your code goes here

# === Provided function
def computeMeans(list_of_lists):
    """
    Returns a list as long as the longest list in LIST_OF_LISTS, where
    the value at index i is the average of the values at index i in
    all of LIST_OF_LISTS' lists.

    Lists shorter than the longest list are padded with their final
    value to be the same length.
    """
    # Find length of longest list
    longest = 0
    for lst in list_of_lists:
        if len(lst) > longest:
            longest = len(lst)
    # Get totals
    tots = [0]*(longest)
    for lst in list_of_lists:
        for i in range(longest):
            if i < len(lst):
                tots[i] += lst[i]
            else:
                tots[i] += lst[-1]
    # Convert tots to an array to make averaging across each index easier
    tots = pylab.array(tots)
    # Compute means
    means = tots/float(len(list_of_lists))
    return means


# === Problem 4
def showPlot1():
    """
    Produces a plot showing dependence of cleaning time on room size.
    """
    # TODO: Your code goes here

def showPlot2():
    """
    Produces a plot showing dependence of cleaning time on number of robots.
    """
    # TODO: Your code goes here

def showPlot3():
    """
    Produces a plot showing dependence of cleaning time on room shape.
```

```
    """
    # TODO: Your code goes here

def showPlot4():
    """
    Produces a plot showing cleaning time vs. percentage cleaned, for
    each of 1-5 robots.
    """
    # TODO: Your code goes here


# === Problem 5

class RandomWalkRobot(BaseRobot):
    """
    A RandomWalkRobot is a robot with the "random walk" movement
    strategy: it chooses a new direction at random after each
    time-step.
    """
    # TODO: Your code goes here


# === Problem 6

def showPlot5():
    """
    Produces a plot comparing the two robot strategies.
    """
    # TODO: Your code goes here
```

문제 세트 12
: 바이러스 개체군의 역동성 시뮬레이션

서문

이 문제 세트에서는 바이러스 개체군의 역동성dynamics에 관하여 확률적stochastic 시뮬레이션을 디자인하고 구현합니다.

바이러스 감염을 치료하는 의약품이 있습니다. 하지만 바이러스는 변형을 하며 해당 약물 혹은 여러 약물들에 대하여 저항력이 생기곤 합니다. 의학 대학원에 가지 않고서도, 바이러스 개체군이 서로 다른 약품에 어떻게 반응하는지를 관찰하여 최적의 투약 방식을 결정할 수 있습니다. 아쉽게도 6.00 코스를 위한 과학 실험실을 준비하지 못했기 때문에 여러분은 파이썬을 이용하여 바이러스 개체군의 역동성을 시뮬레이션해야 하며 시뮬레이션 결과에 기초하여 결과를 이끌어내야 합니다.

- **과제량**

각 문제마다 시간이 얼마나 걸렸는지 알려주세요. 우리가 예상했던 것보다 더 많은 시간이 걸리는 문제를 내지 않도록 주의할 것입니다.

- **공동 작업**

다른 학생들과 함께 작업을 할 수 있습니다. 하지만, 각각의 학생은 개별적으로 작성하고 제출해야 합니다. 누구와 함께 작업을 했는지 명시하는 것을 잊지 마세요. 더 자세한 사항은 강의계획서에 있는 공동 작업 조항을 참고하세요.

시작하기

파일을 다운로드하고 저장하세요.

- ps12.py: 시뮬레이션을 위한 코드 템플릿입니다.

배경: 바이러스, 약물 치료, 계산적 모델

HIV와 인플루엔자와 같은 바이러스는 현대 의학에 중요한 도전이 되고 있습니다. 그들은 진화하는 바이러스이기 때문에 치료하기 매우 어렵습니다.

생물학 기초 수업에서 배웠듯이, 유기체의 특징은 유전자 코드에 의해 결정됩니다. 유기체가 번식을 할 때, 후손은 그들의 부모로부터 유전자 정보를 상속받습니다. 부모의 유전자 정보가 혼합되고 혹은 유전자 복제 과정 중에 에러가 발생하여 결국 유전자 정보에 변화가 일어나게 되므로 개체군에 다양성이 나타내게 됩니다.

바이러스도 예외가 아니며, 바이러스 또한 그들 자신의 유전자 정보를 지니고 번식합니다. 바이러스의 두 가지 특징 때문에 그들을 다루기가 매우 어렵습니다. 첫 번째 특징은 바이러스보다 더 복잡한 유기체에 존재하는 에러 확인 메커니즘(방법)이 바이러스의 복제 메커니즘에는 종종 존재하지 않는 점입니다. 두 번째 특징은 인간 보다 수십 배나orders of magnitude 빠르게 바이러스가 복제된다는 점입니다. 따라서 우리가 진화라고 하면 매우 긴 시간에 걸쳐 일어나는 과정이라고 생각할 수 있지만, 바이러스 개체군은 환자의 치료 과정 중에도 이미 상당한 진화를 할 수 있습니다.

이러한 두 가지 특징들로 말미암아 바이러스 개체군은 병을 치료하는 과정 중에 사용하는 치료요법에 대항하여 빠르게 유전적 저항력을 얻게 됩니다. 이 문제 세트에서 우리는 바이러스 개체군에게 약품을 투여했을 때의 반응을 시뮬레이션으로 탐색할 것입니다. 또한 간소화된 모델을 사용하여 이러한 치료요법이 겪는 문제를 결정지을 것입니다.

계산적 모델링은 HIV와 같은 바이러스를 연구하는 데에 중요한 역할을 해왔습니다(예를 들어 타임지의 "올해의 인물"에 선택된 David Ho의 글을 보세요 http://biowiki.org/~yam/refs/PerelsonEtAl1996.pdf). 이 문제 세트에서 우리는 균체 내in vivo의 바이러스 개체 변화를 설명하는 매우 간소화한 확률 모델을 구현할 것입니다. 여러 세부사항은 비밀로 되어 있습니다(숙주 세포는 명확하게 모델링 하지 않고, 실제 바이러스 개체군보다 몇 배나 적은 양의 개체군을 사용합니다). 그럼에도 불구하고 우리의 모델은 생물학적으로 연관 있는 특징을 보여주며 시뮬레이션 자료를 분석하고 해석하는 기회를 제공합니다.

문제 1 간단한 시뮬레이션을 구현하기(약물치료는 아님)

우리는 바이러스 개체군의 한 보잘것없는 모델로 시작합니다. 환자는 어떠한 약도 먹지 않으며,

바이러스는 약에 대한 저항력도 생기지 않습니다. 우리는 환자 내에 있는 바이러스를 그대로 놓아두었다고 가정하고 단순하게 모델링 하겠습니다.

시뮬레이션의 모든 단계마다 바이러스 입자는 완전히 사라질(환자의 몸에서 제거될) 확률을 가지고 있습니다. 만약 바이러스 입자가 사라지지 않는다면, 번식을 할 것으로 간주됩니다. 사라질 확률은 일정하지만, 번식할 확률은 바이러스 개체군에 따른 함수로 정해집니다. 바이러스 개체군이 많다면, 환자의 몸 안에 개체군이 번식을 할 만큼의 자원이 없기 때문에 번식을 할 확률은 줄어듭니다. 이런 제한 조건을 이해하려면 바이러스 입자는 독립적으로 번식을 하지 못하며, 번식을 하기 위해서 환자의 세포를 사용해야만 한다고 생각할 수 있습니다. 바이러스 개체군이 증가할수록 바이러스가 번식을 하는 데 이용 가능한 숙주 세포host cells가 제한되는 것입니다.

이 모델을 구현하기 위해 SimpleVirus 클래스를 작성해야 합니다. 이 클래스는 단 하나의 바이러스 입자의 상태를 저장합니다. 또한 SimplePatient 클래스를 작성해야 하며 이 클래스는 환자와 관련된 바이러스 개체군의 상태를 저장합니다. SimplePatient 클래스의 `update()` 메소드는 시뮬레이션의 "내부 루프(inner loop)" 입니다. 이 메소드는 단 하나의 시간 단계a single time step 동안 일어난 바이러스 개체군의 상태를 수정하고, 시간 단계가 지난 후에 총 바이러스 개체군의 수를 반환합니다.

`update()`는 사라질 바이러스 입자와 생존할 바이러스 입자를 SimpleVirus 인스턴스의 `doesClear()` 메소드를 통해 결정해야 하고, 또한 그에 따라 SimpleVirus 인스턴스들을 모두 업데이트 해야 합니다. `update()`는 각 바이러스 입자에 대해 `reproduce()` 메소드를 호출해야 합니다. 개체군의 밀도에 기초해서 `reproduce()`는 바이러스 입자의 후손을 의미하는 SimpleVirus의 새로운 인스턴스를 반환하거나, 현재 시간 단계에 바이러스 입자가 번식하지 않았다는 것을 의미하는 NoChildException 에러를 발생시켜야 합니다. `update()` 메소드는 이러한 조건 아래에서 적절하게 환자의 특성attribute을 업데이트 해야 합니다. 모든 바이러스 입자들에 대해 메소드를 반복적으로 호출한 후에, `update()` 메소드는 시간 단계의 끝에 환자의 몸에 남아있는 바이러스 입자의 수를 반환합니다.

SimpleVirus의 `reproduce()` 메소드는 새로운 SimpleVirus 인스턴스를 반환하기 때문에 다음의 확률로 하나의 후손one offspring을 만들어 내야 합니다.

```
self.maxBirthProb * (1 - popDensity)
```

self.maxBirthProb은 최적의 조건 하에서의 출생률입니다(바이러스 개체군은 사용 가능한 숙주 세포와 무시할 수 있을 정도로 관련이 없습니다). popDensity는 (현재 바이러스 개체군) / (환

자가 가질 수 있는 최대 바이러스 개체군)로 정의되며, SimplePatient 클래스의 `update()` 메소드 내부에서 계산되어야 합니다.

힌트: 해당 요소들을 반복 연산하는iterating 동안에 한 변수의 값이 변화되는 것에 주의하세요. 이러한 반복 연산을 완전히 하지 않거나(추가적으로 "도움" 변수들을 사용할 것을 고려하세요), 자료 구조의 결과값이 정확하다는 확신을 가지고 있는 경우에만 그렇게 하도록 하세요(예를 들어, 리스트에서 요소를 하나 제거할 때, 리스트 안에 남아 있는 요소들의 순서에 대해 조심스럽게 생각하세요).

시간 단계들과 실제 시간을 매핑하는 것은 그때 다루고 있는 바이러스의 유형에 따라 달라질 것입니다. 하지만 이 문제 세트에서는 시간 단계를 일종의 모의시간simulated hour으로 생각하세요.

이러한 클래스들에 있는 각 메소드에 대한 자세한 사양은 템플릿을 참조하세요.

SimpleVirus 클래스의 `__init__()`, `doesClear()`, `reproduce()` 메소드를 제한 조건에 맞게 구현하세요. 임의의 숫자를 발생시키기 위해 `random.random()`을 사용하며, 우리의 결과와 같은지 확인하세요.

SimplePatient 클래스의 `__init__()`, `getTotalPop()`, and `update()` 메소드를 구현하세요.

문제 2에서 구현한 것을 확인해보세요.

```
class SimpleVirus(object):
    """
    Representation of a simple virus (does not model drug effects/resistance).
    """

    def __init__(self, maxBirthProb, clearProb):
        """
        Initialize a SimpleVirus instance, saves all parameters as attributes
        of the instance.

        maxBirthProb: Maximum reproduction probability (a float between 0-1)

        clearProb: Maximum clearance probability (a float between 0-1).
            """
            # TODO

    def doesClear(self):
```

```
        """
        Stochastically determines whether this virus is cleared from the
        patient's body at a time step.

      returns: Using a random number generator (random.random()), this method
      returns True with probability self.clearProb and otherwise returns
      False.
      """
      # TODO

    def reproduce(self, popDensity):
        """
        Stochastically determines whether this virus particle reproduces at a
        time step. Called by the update() method in the SimplePatient and
        Patient classes. The virus particle reproduces with probability
        self.maxBirthProb * (1 - popDensity).

        If this virus particle reproduces, then reproduce() creates and
        returns the instance of the offspring SimpleVirus (which has the
        same maxBirthProb and clearProb values as its parent).

        popDensity: the population density (a float), defined as the current
        virus population divided by the maximum population.

        returns: a new instance of the SimpleVirus class representing the
        offspring of this virus particle. The child should have the same
        maxBirthProb and clearProb values as this virus. Raises a
        NoChildException if this virus particle does not reproduce.
        """
        # TODO

class SimplePatient(object):
    """
    Representation of a simplified patient. The patient does not take any
    drugs and his/her virus populations have no drug resistance.
    """

    def __init__(self, viruses, maxPop):
        """
        Initialization function, saves the viruses and maxPop parameters
        as attributes.

```

```
        viruses: the list representing the virus population (a list of
        SimpleVirus instances)

        maxPop: the  maximum virus population for this patient (an integer)
        """
        # TODO

    def getTotalPop(self):
        """
        Gets the current total virus population.

        returns: The total virus population (an integer)
        """
        # TODO

    def update(self):
        """
        Update the state of the virus population in this patient for a single
        time step. update() should execute the following steps in this order:

       - Determine whether each virus particle survives and updates the list
         of virus particles accordingly.

        - The current population density is calculated. This population density
          value is used until the next call to update()

        - Determine whether each virus particle should reproduce and add
          offspring virus particles to the list of viruses in this patient.

        returns: the total virus population at the end of the update (an
        integer)
        """
        # TODO
```

문제 2 간단한 시뮬레이션을 실행하고 분석하기(약물치료는 아님)

약물을 사용하기 전에, 개체군의 역동성에 대해 이해하는 것으로 시작해야 합니다. `problem2()` 함수를 작성하세요. 이 메소드는 SimplePatient 인스턴스를 만들어야 하며[instantiate], 바이러스 개체군의 변화를 시뮬레이션하기 위해 `update()` 메소드를 반복적으로 호출해야 합니다. 시뮬레이션을 진행하며 개체군 값들을 저장하고 pylab을 이용해서 시간에 관한 함수로써 바이러스 개

체군의 변화를 그리세요. 그래프에 제목과 라벨을 붙이세요.

SimplePatient는 다음 매개변수들을 사용하여 인스턴스로 만들어야 합니다[instantiated].

- viruses, SimpleVirus 인스턴스 100개로 구성된 리스트입니다.
- maxPop, 생존 가능한 최대 바이러스 개체군이며 1000입니다.

viruses 리스트의 각 SimpleVirus 인스턴스는 다음 매개변수로 초기화 되어야 합니다.

- maxBirthProb, 한 바이러스 입자의 최대번식확률[Max Reproduction Probability]은 0.1입니다.
- clearProb, 한 바이러스 입자의 최대소멸확률[Maximum Clearance Probability]은 0.05입니다.

problem2()를 작성하여 한 환자 인스턴스를 만들고, 300시간 단계 동안에(update()를 300번 호출하겠죠) 바이러스 개체군의 변화를 시뮬레이션하고, 시간에 관한 함수로써 바이러스 개체군의 변화를 그려보세요(plot). 시뮬레이션을 여러 차례 실행하고, 대표적인 결과를 선택하여 보고서에 포함하세요. 그래프에 제목과 축에 라벨을 붙이세요.

여러분의 보고서에 그래프를 추가하기 위해 그림 창 하단부에 있는 디스크 모양 아이콘을 클릭해서 이미지 파일로 저장하세요. 마이크로소프트 워드(혹은 어떠한 워드 프로세서를 사용해도 상관 없습니다)를 사용해서 이미지 파일을 불러오세요.

보고서에는 그래프를 추가하고, 다음 질문에 답하세요. 개체군의 수가 증가하는 것이 멈추기까지 시간이 얼마나 걸리나요?

```
def problem2():
    """
    Run the simulation and plot the graph for problem 2 (no drugs are
    used,viruses do not have any drug resistance).

    Instantiates a patient, runs a simulation for 300 timesteps, and plots
    the total virus population as a function of time.
    """
    # TODO
```

문제 3 약물을 투여한 시뮬레이션 구현하기

이 문제에서 우리는 환자에게 약물을 투여한 효과와 더불어 바이러스 입자가 후손에게 유전적인 특징을 상속하거나 변경시켜 약물에 저항력을 갖는 능력에 대해 고려해볼 것입니다.

바이러스 개체군이 번식하면서, 바이러스 후손에 있어 변형이 생길 것이고, 따라서 바이러스 개체군에 유전적 다양성을 더해집니다. 몇몇 바이러스 입자들은 약물에 저항력을 갖도록 변형될 것입니다.

Patient 클래스의 `addPrescription()` 메소드를 통해 환자에게 약물을 투여합니다. 약물이 들어가면 무슨 일이 일어날까요? 우리가 다루고 있는 약물은 약물에 저항력이 없는 바이러스 입자들을 직접적으로 죽이지 못하지만, 그런 바이러스 입자들의 번식을 막는 것이죠(HIV를 치료하는 약물도 실제적으로 이와 같이 작용합니다). 약물에 저항력을 가진 바이러스 입자들은 계속해서 정상적으로 번식을 합니다.

이런 효과를 모델링하기 위해 우리는 SimpleVirus의 하위 클래스 ResistantVirus를 도입합니다. ResistantVirus는 바이러스 입자의 약물 저항력 상태를 저장하고 있으며, 후손에게 약물에 대한 저항력을 상속하는 역할을 합니다.

또한 우리는 약물 치료를 받고 있고, ResistantVirus 인스턴스들의 집단을 처리하는 환자를 표현하는 방법이 필요합니다. 이것을 해결하기 위해 SimplePatient의 하위 클래스인 Patient 클래스를 도입합니다. Patient는 `ResistantVirus()`의 새로운 메소드들을 활용해야 하고, 환자에게 투여하는 약물 목록을 저장하고 있습니다.

이러한 두 클래스에 있는 메소드들의 자세한 사양은 템플릿을 확인하세요.

ResistantVirus와 Patient 클래스를 구현하세요.

문제 4에서 구현할 것을 확인할 것입니다.

```
class ResistantVirus(SimpleVirus):
    """
    Representation of a virus which can have drug resistance.
    """

    def __init__(self, maxBirthProb, clearProb, resistances, mutProb):
        """
        Initialize a ResistantVirus instance, saves all parameters as
        attributes of the instance.

        maxBirthProb: Maximum reproduction probability (a float between 0-1)

```

```
        clearProb: Maximum clearance probability (a float between 0-1).

        resistances: A dictionary of drug names (strings) mapping to the state
        of this virus particle's resistance (either True or False) to each drug.
        e.g. {'guttagonol':False, 'grimpex',False}, means that this virus
        particle is resistant to neither guttagonol nor grimpex.

        mutProb: Mutation probability for this virus particle (a float). This is
        the probability of the offspring acquiring or losing resistance to a drug.
        """
        # TODO

    def getResistance(self, drug):
        """
        Get the state of this virus particle's resistance to a drug. This method
        is called by getResistPop() in Patient to determine how many virus
        particles have resistance to a drug.
        drug: the drug (a string).
        returns: True if this virus instance is resistant to the drug, False
        otherwise.
        """
        # TODO

    def reproduce(self, popDensity, activeDrugs):
        """
        Stochastically determines whether this virus particle reproduces at a
        time step. Called by the update() method in the Patient class.

        If the virus particle is not resistant to any drug in activeDrugs,
        then it does not reproduce. Otherwise, the virus particle reproduces
        with probability:

        self.maxBirthProb * (1 - popDensity).

        If this virus particle reproduces, then reproduce() creates and returns
        the instance of the offspring ResistantVirus (which has the same
        maxBirthProb and clearProb values as its parent).

        For each drug resistance trait of the virus (i.e. each key of
        self.resistances), the offspring has probability 1-mutProb of
        inheriting that resistance trait from the parent, and probability
        mutProb of switching that resistance trait in the offspring.

```

```
        For example, if a virus particle is resistant to guttagonol but not
        grimpex, and 'self.mutProb' is 0.1, then there is a 10% chance that
        that the offspring will lose resistance to guttagonol and a 90%
        chance that the offspring will be resistant to guttagonol.
        There is also a 10% chance that the offspring will gain resistance to
        grimpex and a 90% chance that the offspring will not be resistant to
        grimpex.

        popDensity: the population density (a float), defined as the current
        virus population divided by the maximum population

        activeDrugs: a list of the drug names acting on this virus particle
        (a list of strings).

        returns: a new instance of the ResistantVirus class representing the
        offspring of this virus particle. The child should have the same
        maxBirthProb and clearProb values as this virus. Raises a
        NoChildException if this virus particle does not reproduce.
        """
        # TODO

class Patient(SimplePatient):
    """
    Representation of a patient. The patient is able to take drugs and his/her
    virus population can acquire resistance to the drugs he/she takes.
    """

    def __init__(self, viruses, maxPop):
        """
        Initialization function, saves the viruses and maxPop parameters as
        attributes. Also initializes the list of drugs being administered
        (which should initially include no drugs).

        viruses: the list representing the virus population (a list of
        SimpleVirus instances)

        maxPop: the  maximum virus population for this patient (an integer)
        """
        # TODO

    def addPrescription(self, newDrug):
        """
        Administer a drug to this patient. After a prescription is added, the
```

```
        drug acts on the virus population for all subsequent time steps. If the
        newDrug is already prescribed to this patient, the method has no effect.

        newDrug: The name of the drug to administer to the patient (a string).

        postcondition: list of drugs being administered to a patient is updated
        """
        # TODO

    def getPrescriptions(self):
        """
        Returns the drugs that are being administered to this patient.

        returns: The list of drug names (strings) being administered to this
        patient.
        """
       # TODO

    def getResistPop(self, drugResist):
       """
       Get the population of virus particles resistant to the drugs listed
       in  drugResist.

       drugResist: Which drug resistances to include in the population (a list
       of strings - e.g. ['guttagonol'] or ['guttagonol', 'grimpex'])

       returns: the population of viruses (an integer) with resistances to all
       drugs in the drugResist list.
       """
       # TODO

    def update(self):
       """
       Update the state of the virus population in this patient for a single
       time step. update() should execute these actions in order:

       - Determine whether each virus particle survives and update the list
         of virus particles accordingly

       - The current population density is calculated. This population density
         value is used until the next call to update().

        offspring virus particles to the list of viruses in this patient.
```

```
        The listof drugs being administered should be accounted for in the
        determination of whether each virus particle reproduces.

        returns: the total virus population at the end of the update (an
        integer)
        """
        # TODO
```

문제 4 약물을 투여한 시뮬레이션을 실행하고 분석하기

이 문제에서 우리는 문제 3에서 작성한 구현을 사용해서 시뮬레이션을 실행하겠습니다. 다음의 매개변수를 넣어서 Patient 인스턴스를 만드세요. 그 뒤에, 시뮬레이션을 실행하고 몇 가지 질문에 답하세요.

- viruses, ResistantVirus 인스턴스 100개로 구성된 리스트
- maxPop = 1000, 생존 가능한 최대 바이러스 개체군Maximum Sustainable Virus Population

viruses 리스트에 있는 각 ResistantVirus 인스턴스는 다음 매개변수로 초기화 되어야 합니다.

- maxBirthProb = 0.1, 바이러스 입자가 가질 수 있는 최대 번식 확률
- clearProb = 0.05, 바이러스 입자가 사라질 수 있는 확률
- resistances = {'guttagonol' :False}, 실험에서 약물에 대한 바이러스의 유전적 저항력
- mutProb = 0.005, 바이러스 입자 후손의 변형 확률

150시간 단계로 구성되는 시뮬레이션을 실행하세요. 그 뒤에는 guttagonol이라는 약물을 추가하고, 또 다른 150시간 단계로 구성되는 시뮬레이션을 실행하세요. 문제 2와 같이 여러 번 시행하고, 그 결과들이 반복 가능하며, 대표성을 잘 나타내는지 확인하세요.

전체 개체군과 guttagonol 에 저항력이 있는 바이러스 입자 개체군의 기록을 그래프로 그리세요. 어떤 추세trend들을 관찰할 수 있나요? 그런 추세들이 여러분이 실험하기 전에 직관을 가졌던 것과 일치하나요? 그래프와 질문들에 대한 답안을 보고서에 포함하세요.

```
def problem4():
    """
    Runs simulations and plots graphs for problem 4.

    Instantiates a patient, runs a simulation for 150 timesteps, adds
```

```
    guttagonol, and runs the simulation for an additional 150 timesteps.

    total virus population vs. time  and guttagonol-resistant virus population
    vs. time are plotted
    """
    # TODO
```

문제 5 환자의 치료를 늦추는 것에 대한 효과 및 결과

이 문제에서 우리는 바이러스 개체군을 박멸하기 위해 환자의 치료를 늦추어 그 효과에 대해 살펴봅니다. 환자의 결과를 관찰하기 위해 시뮬레이션을 여러 번 실행해야 할 필요가 있을 것입니다.

환자에게 guttagonol을 투여하기 전에 시뮬레이션을 300, 150, 75, 0번의 시간 단계에 따라 실행하세요. 그 뒤에 추가로 150시간 단계만큼 시뮬레이션을 실행하세요. 문제 4에서 ResistantVirus와 Patient를 초기화할 때 사용한 동일한 매개 변수 값들을 사용하세요.

네 가지 조건들에 대해서 각각 여러 번 실험을 반복하면서 최종 바이러스 개체군들을 기록하세요. pylab의 `hist()` 함수를 사용하여 각각 하나의 조건에 따라 최종 바이러스 개체군에 대한 히스토그램을 그리세요. 히스토그램의 x축은 최종 바이러스 개체군의 총 합으로 하고, y축은 각 히스토그램 막대[bin]에 속하는 환자의 수로 합니다. 대표성을 띠는 분포를 찾기 위해 각 조건 마다 몇 번씩 반복해서, 시뮬레이션을 몇 번 시행할 것인지 결정해야 합니다. 여러분이 시뮬레이션을 하기로 한 횟수의 정당성을 보고서에 설명하세요.

히스토그램 4개를 보고서에 포함하고, 다음 질문들에 대해 답하세요. 만약 최종 바이러스 입자 수 0~50을 치료되었다(병에 차도가 있다)고 간주한다면, 시뮬레이션이 끝난 뒤에 환자 중 몇 퍼센트가 치료되었을까요(병에 차도가 있었을까요)? 치료된 환자 수(병에 차도가 있는 환자 수)와 치료를 늦추는 것 사이에는 어떤 관계가 있나요? 모델로부터 어떻게 이런 관계가 발생하는지 설명하세요.

힌트: 각 조건에 대한 분포를 그리기까지 충분한 만큼 시뮬레이션을 실행하려면 상당한 시간이 걸릴지 모릅니다. 실행 횟수를 줄여서 코드를 디버깅하세요. 일단 코드 디버깅이 끝나고 나면, 시뮬레이션을 하는데 몇 분 정도 걸릴 정도로 시행 횟수를 많이 늘리세요. print를 사용하여 시뮬레이션의 진행을 지켜보세요. 적당한 만큼 실행하려면 시뮬레이션을 돌리는데 3~6분 가량 걸려야 할 것입니다.

```
def problem5():
    """
    Runs simulations and make histograms for problem 5.

    Runs multiple simulations to show the relationship between delayed
    treatment and patient outcome.

    Histograms of final total virus populations are displayed for delays of
    300, 150, 75, 0 timesteps (followed by an additional 150 timesteps of
    simulation).
    """
    # TODO
```

문제 6 두 종류 약물을 사용하여 치료 계획을 디자인하기

약물에 저항력을 가진 문제를 해결하는 한가지 접근 방법으로 칵테일을 사용할 수 있습니다. 칵테일은 다양한 약물을 처방하여 각 약물들이 독립적으로 바이러스 개체군을 공격하도록 하는 방법을 말합니다.

문제 6과 7에서 우리는 바이러스를 치료하기 위해 독립적으로 작용하는 두 가지 약물을 사용할 것입니다. 두 약물을 관리하는 최적의 방법을 결정하는 데 이 모델을 사용하겠습니다. 특히, 우리는 처음 약물 그리고 두 번째 약물을 사용할 때 그 사이에 지체 시간lag time을 두어, 이것이 환자의 결과에 어떠한 영향을 미치는지 조사합니다.

문제 6-7에서 Patient를 초기화하기 위해 다음 매개변수를 사용하세요.

- viruses, ResistantVirus 인스턴스 100개로 구성된 리스트
- maxPop = 1000, 생존 가능한 최대 바이러스 개체군Maximum Sustainable Virus Population

viruses 리스트의 각 ResistantVirus 인스턴스는 다음 매개변수로 초기화해야 합니다.

- maxBirthProb = 0.1, 바이러스 입자가 가질 수 있는 최대 번식 확률
- clearProb = 0.05, 바이러스 입자가 사라질 수 있는 확률
- resistances = {'guttagonol' :False, 'grimpex' :False}, 실험에서 약물에 대한 바이러스의 유전적 저항력
- mutProb = 0.005, 바이러스 입자 후손의 변형 확률

환자에게 guttagonol을 투여하기 전에 150시간 단계로 구성되는 시뮬레이션을 실행하세요. 그리고 나서, 환자에게 두 번째 약물 grimpex을 투여하기 전에, 300, 150, 75, 0번의 시간 단계로 시뮬레이션을 실행하세요. 마지막으로 뒤에 추가적으로 150시간 단계로 시뮬레이션을 실행하세요.

네 가지 조건들에 대해서 각각 실험을 30번 반복하면서, 그와 동시에 최종 바이러스 개체군도 기록해야 합니다. pylab의 `hist()` 함수를 사용하여 각 조건마다 최종 바이러스 개체군에 대한 히스토그램을 그리세요.

히스토그램을 보고서에 포함하고, 다음 질문들에 답하세요. 시뮬레이션이 끝난 뒤에 환자들 중 몇 퍼센트가 치료되었을까요(병에 차도가 있을까요)? 치료된 환자의 수(병에 차도가 있는 환자의 수)와 두 약물 투여 사이의 시간lag time은 어떤 관계는 무엇인가요?

힌트: 문제 5에서처럼, 여기서도 시뮬레이션을 실행하는 데 수 분의 시간이 걸립니다. print를 사용하여 시뮬레이션의 진행을 지켜보세요.

```
def problem6():
    """
    Runs simulations and make histograms for problem 6.

    Runs multiple simulations to show the relationship between administration
    of multiple drugs and patient outcome.

    Histograms of final total virus populations are displayed for lag times of
    150, 75, 0 timesteps between adding drugs (followed by an additional 150
    timesteps of simulation).
    """
    # TODO
```

문제 7 두 종류 약물을 사용한 바이러스 개체군의 역동성 분석

환자의 결과와 두 약물 투여 사이의 시간 간격lag time이 어떤 관계인지 이해하기 위해, 우리는 문제 6의 바이러스 개체군의 역동성에 대해 개별적인 두 번의 시뮬레이션을 자세히 살펴보겠습니다.

환자에게 guttagonol을 투여하기 전에 150시간 단계로 구성되는 시뮬레이션을 실행하세요. 그리고 그 후에 두 번째 약물 grimpex을 투여하기 전에, 300시간 단계로 시뮬레이션을 실행하세요. 그 후에 환자에게. 그 뒤에 추가적으로 150시간 단계로 시뮬레이션을 실행하세

요. 문제 6에서 사용한 매개변수 Patient와 Resistantvirus에 같은 초기화 값들을 사용하세요.

환자에게 guttagonol과 grimpex를 동시에 투여하기 전에 150시간 단계로 두 번째 시뮬레이션을 실행하세요. 그 후에 추가적으로 150시간 단계로 시뮬레이션을 실행하세요.

시뮬레이션을 여러 차례 돌려서 여러분이 분석하는 결과 값들이 가장 빈번하게 나오는 대표적인 결과 값이라는 것을 반드시 확인하세요.

위의 두 시뮬레이션들에 대해서 총 개체군과 guttagonol에 저항력이 있는 바이러스 개체군, grimpex에 저항력이 있는 바이러스 개체군, 두 약물 모두에 저항력이 있는 개체군을 시간에 대한 함수로 그리세요.

환자의 결과와 두 약물을 투여하는 시간과의 관계가 왜 올라가는지 설명하세요.

```
def problem7():
    """
    Run simulations and plot graphs examining the relationship between
    administration of multiple drugs and patient outcome.

    Plots of total and drug-resistant viruses vs. time are made for a
    simulation with a 300 time step delay between administering the 2 drugs and
    a simulations for which drugs are administered simultaneously.
    """
    # TODO
```

문제 8 처방전을 따르지 않는 환자

환자들이 처방 받은 약물을 처방대로 일정하게 약을 복용하지 않는 것은 매우 흔한 일입니다. 환자들은 종종 약의 복용 여부를 기억하지 못하기도 하고, 심지어는 약물 복용을 거부하기도 합니다. 이러한 경우 일어나는 결과들을 고려하는 모델링을 어떻게 할 수 있을지 보고서에 설명하세요. 이에 대한 코드는 작성하지 마세요.

제출 과정

저장. ps12.py 안에 수정한 코드를 작성하세요. 보고서 파일은 writeup.pdf 이름하여 제출해야 합니다.

시간과 공동 작업 정보. 파일 시작부분에, 주석으로 파일 이름과 해당 문제 세트를 해결하는데

대략적으로 몇 시간이 걸렸는지 적어주세요. 또한 공동 작업을 한 학우(들)이 있다면 적어주세요. 다음과 같이 적으면 됩니다.

```
# ps12.py
# Problem Set 12
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
... your code goes here ...
```

다시 한 번 확인하기. 문제 세트를 완성한 후에 다시 한 번 확인하세요.

- ps12.py를 실행하고, 에러 없이 동작하는지 확인하세요.

```
1  # ps12.py
2  #
3  # 6.00 Problem Set 12
4  #
5  # Name:
6  # Collaborators:
7  # Time:
8
9  import numpy
10 import random
11 import pylab
12
13 class NoChildException(Exception):
14     """
15     NoChildException is raised by the reproduce() method in the SimpleVirus
16     and ResistantVirus classes to indicate that a virus particle does not
17     reproduce. You can use NoChildException as is, you do not need to
18     modify/add any code.
19     """
20
21 #
22 # PROBLEM 1
23 #
24
25 class SimpleVirus(object):
26     """
27     Representation of a simple virus (does not model drug effects/resistance).
28     """
29
30     def __init__(self, maxBirthProb, clearProb):
31         """
32         Initialize a SimpleVirus instance, saves all parameters as attributes
33         of the instance.
34         maxBirthProb: Maximum reproduction probability (a float between 0-1)
35
```

```
        clearProb: Maximum clearance probability (a float between 0-1).
        """
        # TODO

    def doesClear(self):
        """
        Stochastically determines whether this virus is cleared from the
        patient's body at a time step.

        returns: Using a random number generator (random.random()), this method
        returns True with probability self.clearProb and otherwise returns
        False.
        """
        # TODO

    def reproduce(self, popDensity):
        """
        Stochastically determines whether this virus particle reproduces at a
        time step. Called by the update() method in the SimplePatient and
        Patient classes. The virus particle reproduces with probability
        self.maxBirthProb * (1 - popDensity).

        If this virus particle reproduces, then reproduce() creates and returns
        the instance of the offspring SimpleVirus (which has the same
        maxBirthProb and clearProb values as its parent).

        popDensity: the population density (a float), defined as the current
        virus population divided by the maximum population.

        returns: a new instance of the SimpleVirus class representing the
        offspring of this virus particle. The child should have the same
        maxBirthProb and clearProb values as this virus. Raises a
        NoChildException if this virus particle does not reproduce.
        """
        # TODO

class SimplePatient(object):
    """
    Representation of a simplified patient. The patient does not take any drugs
    and his/her virus populations have no drug resistance.
    """

    def __init__(self, viruses, maxPop):
        """
        Initialization function, saves the viruses and maxPop parameters as
        attributes.

        viruses: the list representing the virus population (a list of
        SimpleVirus instances)
        maxPop: the  maximum virus population for this patient (an integer)
        """
        # TODO
```

```
88
89      def getTotalPop(self):
90          """
91          Gets the current total virus population.
92
93          returns: The total virus population (an integer)
94          """
95          # TODO
96
97      def update(self):
98          """
99          Update the state of the virus population in this patient for a single
100         time step. update() should execute the following steps in this order:
101
102         - Determine whether each virus particle survives and updates the list
103           of virus particles accordingly.
104
105         - The current population density is calculated. This population density
106           value is used until the next call to update()
107
108         - Determine whether each virus particle should reproduce and add
109           offspring virus particles to the list of viruses in this patient.
110
111         returns: the total virus population at the end of the update (an integer)
112       """
113       # TODO
114
115 #
116 # PROBLEM 2
117 #
118
119 def problem2():
120     """
121     Run the simulation and plot the graph for problem 2 (no drugs are used,
122     viruses do not have any drug resistance).
123
124     Instantiates a patient, runs a simulation for 300 timesteps, and plots the
125     total virus population as a function of time.
126     """
127     # TODO
128
129 #
130 # PROBLEM 3
131 #
132
133 class ResistantVirus(SimpleVirus):
134     """
135     Representation of a virus which can have drug resistance.
136     """
137
138     def __init__(self, maxBirthProb, clearProb, resistances, mutProb):
139         """
```

```
        Initialize a ResistantVirus instance, saves all parameters as attributes
        of the instance.

        maxBirthProb: Maximum reproduction probability (a float between 0-1)

        clearProb: Maximum clearance probability (a float between 0-1).

        resistances: A dictionary of drug names (strings) mapping to the state
        of this virus particle's resistance (either True or False) to each drug.
        e.g. {'guttagonol':False, 'grimpex',False}, means that this virus
        particle is resistant to neither guttagonol nor grimpex.

        mutProb: Mutation probability for this virus particle (a float). This is
        the probability of the offspring acquiring or losing resistance to a drug.
        """
        # TODO

    def getResistance(self, drug):
        """
        Get the state of this virus particle's resistance to a drug. This method
        is called by getResistPop() in Patient to determine how many virus
        particles have resistance to a drug.

        drug: the drug (a string).

        returns: True if this virus instance is resistant to the drug, False
        otherwise.
        """
        # TODO

    def reproduce(self, popDensity, activeDrugs):
        """
        Stochastically determines whether this virus particle reproduces at a
        time step. Called by the update() method in the Patient class.

        If the virus particle is not resistant to any drug in activeDrugs,
        then it does not reproduce. Otherwise, the virus particle reproduces
        with probability:

        self.maxBirthProb * (1 - popDensity).

        If this virus particle reproduces, then reproduce() creates and returns
        the instance of the offspring ResistantVirus (which has the same
        maxBirthProb and clearProb values as its parent).

        For each drug resistance trait of the virus (i.e. each key of
        self.resistances), the offspring has probability 1-mutProb of
        inheriting that resistance trait from the parent, and probability
        mutProb of switching that resistance trait in the offspring.

        For example, if a virus particle is resistant to guttagonol but not
        grimpex, and 'self.mutProb' is 0.1, then there is a 10% chance that
```

```
192         that the offspring will lose resistance to guttagonol and a 90%
193         chance that the offspring will be resistant to guttagonol.
194         There is also a 10% chance that the offspring will gain resistance to
195         grimpex and a 90% chance that the offspring will not be resistant to
196         grimpex.
197
198         popDensity: the population density (a float), defined as the current
199         virus population divided by the maximum population
200         activeDrugs: a list of the drug names acting on this virus particle
201         (a list of strings).
202
203         returns: a new instance of the ResistantVirus class representing the
204         offspring of this virus particle. The child should have the same
205         maxBirthProb and clearProb values as this virus. Raises a
206         NoChildException if this virus particle does not reproduce.
207         """
208         # TODO
209
210 class Patient(SimplePatient):
211     """
212     Representation of a patient. The patient is able to take drugs and his/her
213     virus population can acquire resistance to the drugs he/she takes.
214     """
215
216     def __init__(self, viruses, maxPop):
217         """
218         Initialization function, saves the viruses and maxPop parameters as
219         attributes. Also initializes the list of drugs being administered
220         (which should initially include no drugs).
221
222         viruses: the list representing the virus population (a list of
223         SimpleVirus instances)
224
225         maxPop: the  maximum virus population for this patient (an integer)
226         """
227         # TODO
228
229     def addPrescription(self, newDrug):
230         """
231         Administer a drug to this patient. After a prescription is added, the
232         drug acts on the virus population for all subsequent time steps. If the
233         newDrug is already prescribed to this patient, the method has no effect.
234
235         newDrug: The name of the drug to administer to the patient (a string).
236
237         postcondition: list of drugs being administered to a patient is updated
238         """
239         # TODO
240
241     def getPrescriptions(self):
242         """
```

```
        Returns the drugs that are being administered to this patient.

        returns: The list of drug names (strings) being administered to this
        patient.
        """
        # TODO

    def getResistPop(self, drugResist):
        """
        Get the population of virus particles resistant to the drugs listed in
        drugResist.

        drugResist: Which drug resistances to include in the population (a list
        of strings - e.g. ['guttagonol'] or ['guttagonol', 'grimpex'])

        returns: the population of viruses (an integer) with resistances to all
        drugs in the drugResist list.
        """
        # TODO

    def update(self):
        """
        Update the state of the virus population in this patient for a single
        time step. update() should execute these actions in order:

        - Determine whether each virus particle survives and update the list of
          virus particles accordingly

        - The current population density is calculated. This population density
          value is used until the next call to update().

        - Determine whether each virus particle should reproduce and add
          offspring virus particles to the list of viruses in this patient.
          The listof drugs being administered should be accounted for in the
          determination of whether each virus particle reproduces.

        returns: the total virus population at the end of the update (an integer)
        """
        # TODO

#
# PROBLEM 4
#
def problem4():
    """
    Runs simulations and plots graphs for problem 4.

    Instantiates a patient, runs a simulation for 150 timesteps, adds
    guttagonol, and runs the simulation for an additional 150 timesteps.

    total virus population vs. time  and guttagonol-resistant virus population
```

```
294     vs. time are plotted
295     """
296     # TODO
297
298 #
299 # PROBLEM 5
300 #
301
302 def problem5():
303     """
304     Runs simulations and make histograms for problem 5.
305
306     Runs multiple simulations to show the relationship between delayed treatment
307     and patient outcome.
308
309     Histograms of final total virus populations are displayed for delays of 300,
310     150, 75, 0 timesteps (followed by an additional 150 timesteps of
311     simulation).
312     """
313     # TODO
314
315 #
316 # PROBLEM 6
317 #
318
319 def problem6():
320     """
321     Runs simulations and make histograms for problem 6.
322
323     Runs multiple simulations to show the relationship between administration
324     of multiple drugs and patient outcome.
325
326     Histograms of final total virus populations are displayed for lag times of
327     150, 75, 0 timesteps between adding drugs (followed by an additional 150
328     timesteps of simulation).
329     """
330     # TODO
331
332 #
333 # PROBLEM 7
334 #
335
336 def problem7():
337     """
338     Run simulations and plot graphs examining the relationship between
339     administration of multiple drugs and patient outcome.
340
341     Plots of total and drug-resistant viruses vs. time are made for a
342     simulation with a 300 time step delay between administering the 2 drugs and
343     a simulations for which drugs are administered simultaneously.
344     """
345     # TODO
```