소프트웨어입문설계, 과제 5-1

출제: 2020년 4월 15일

제출기한: 2020**년 4월 22일 16시 00분 (기한 내 미제출시 0점 처리)**

- *본인의 hconnect에 생성된 본 강좌 프로젝트 (<연도>_<학수번호>_<수업코드>/<년도>_<학수번호>_<학번>.git)에 git push를 통해 제출된 답안만 인정함.*
- *아래 예와 같은 식으로 본인의 프로젝트 아래 <과제 번호>/<문제 번호>/<각 문제의 답안 파일>의 구조가 되도록 답안 파일을 작성.*

```
+2020_ITE1014_12595/2020_ITE1014_본인학번/
  + 5-1/
      - 1.py
      - 2.py
      - 3.py
      - 4.py
      - 5.py
```

- *제출 시점은 commit이 작성된 시점이 아니라 git push가 이루어진 시점으로 판단함.*

1. 5명의 학생의 이름과 시험 점수(정수)를 차례대로 입력 받은 후, 점수를 조회하고 싶은 학생의 이름을 입력하면 해당 학생의 시험 점수를 출력하는 프로그램을 작성하시오. 아래 실행 예와 같이 출력을 해야한다 (⏎는 사용자가 입력 후에 엔터키를 누른 것을 의미한다).
   A. 프로그램 전체에 걸쳐 for나 while 같은 **반복문을 사용하지 말 것.**
   B. 학생별 점수의 저장과 학생의 존재여부를 판단할 때 **반드시 dictionary를 사용할 것.**
   C. 점수를 입력한 적 없는 학생의 이름을 조회하면 아래 실행 예와 같은 메시지 출력 후 종료.

D.

(실행 예 1)

```
Tom 90⏎
Amy 80⏎
Bob 85⏎
Emma 95⏎
Aaron 95⏎
Which student's score?
Bob
Bob's score: 85
```

(실행 예 2)

```
Tom 90⏎
Amy 80⏎
Bob 85⏎
Emma 95⏎
Aaron 95⏎
Which student's score?
Nancy
Nancy is not in the database.
```

E. 제출 파일: Python 소스 파일 1개 (파일 이름은 1.**py**)

2. 다음과 같은 프로그램을 작성하시오. 아래 실행 예와 같이 출력을 해야한다 (⏎는 사용자가 입력 후에 엔터키를 누른 것을 의미한다).

A. 100개의 임의의 숫자를 발생시켜, 이를 하나의 List에 저장한다. 숫자들은 1에서부터 1000까지의 범위를 가진다.

B. for문을 이용하여 이 List 안의 숫자들을 출력한다. 숫자들 간에는 한 칸의 빈 간격을 둔다. 숫자들을 모두 출력한 후에는 newline을 한 번 출력한다.

C. for문을 이용하여 이 List안의 숫자 중에서 가장 큰 숫자를 찾아서 출력한다. (max()함수 쓰지 말 것)

D.

(실행 예)

```
766 130 704 821 206 333 264 400 26 560 973 607 64 76 591 8 307 374
390 231 509 345 846 694 245 835 6 893 146 615 941 358 897 795 921
870 504 784 109 461 266 439 227 727 750 13 407 386 616 176 846 465
451 160 985 742 817 378 673 784 186 626 470 921 871 401 506 193 681
482 545 741 567 120 831 270 296 663 561 733 976 577 261 998 655 906
860 672 688 505 402 129 384 619 820 625 985 93 908 362

max value: 998
```

E. 제출 파일: Python 소스 파일 1개 (파일 이름은 2.**py**)

3. 공백으로 구분된 여러 단어로 이루어진 문자열을 하나 받아, 각각의 단어가 몇 번 등장하는지

출력하는 프로그램을 작성하시오. 아래 실행 예와 같이 출력을 해야한다 (⏎는 사용자가 입력 후에 엔터키를 누른 것을 의미한다).

A. 반드시 list와 dictionary를 사용하도록 한다.

B. Python의 str.count() 함수 사용 불가.

C.

```
(실행 예)
what are you doing doing how are doing do you wanna doing⏎
what: 1
are: 2
doing: 4
how: 1
you: 2
do: 1
wanna: 1
```

D. 제출 파일: Python 소스 파일 1개 (파일 이름은 3.**py**)

4. 문자열의 길이를 입력 받으면 **영어 소문자 알파벳으로만** 이루어진 그 길이만큼의 random string을 출력해내는 프로그램을 작성하시오. 다음 함수를 구현하고 이를 이용하여 프로그램을 작성하여야 한다. 아래 실행 예와 같이 출력을 해야한다 (⏎는 사용자가 입력 후에 엔터키를 누른 것을 의미한다).

A. getRandomString(leng) : leng만큼의 길이의 string을 random하게 생성하여 return하는 함수.

B. Hint: str.join(ls) 함수는 str의 내용을 구분자로 하며 list ls에 들어있는 모든 문자열을 이어 붙인 새로운 문자열을 반환한다.

C. Hint: chr() 함수는 ASCII 코드 숫자값이 나타내는 알파벳 문자 하나를 담은 문자열을 리턴한다.

D.

```
(실행 예 1)
10⏎
dmbzkndzga
```

```
(실행 예 2)
5⏎
mqhwl
```

E. 제출 파일: Python 소스 파일 1개 (파일 이름은 4.**py**)

5. 치즈 공장 옆에 사는 쥐가 있는데 이 쥐가 치즈를 얼마나 갉아 먹고 있는지를 알아보는 프로그램을 작성하시오.

   A. 치즈는 여러 개의 조각으로 이루어져 있는데, 이 조각에는 한 글자씩의 소문자 알파벳이 쓰여있다. 즉, 치즈는 문자열로 표현된다는 것이다. 사용자가 치즈의 크기(문자열의 길이)를 입력하면, 치즈는 Random하게 만들어 진다. 이것은 앞의 문제 2번에서 구현된 함수를 이용하면 되겠다. 이 치즈의 크기는 10에서부터 30까지로 제한된다.

   B. 쥐는 머리가 나빠서, 치즈를 한 번 먹을 때 특정 알파벳이 쓰여진 치즈만 찾아서 먹는다. 예를 들면, 'aabbccddb' 라는 치즈가 있을 때, 쥐가 생각한 알파벳이 'b'라면, 쥐가 먹고 나서 남아있는 치즈는 'aa__ccdd_'가 된다. 먹고 나서의 치즈의 형태는 '_' 이다. 치즈 안에 생각한 알파벳이 없다면 치즈는 그 상태 그대로 유지할 수 있다. 그리고 한 번 생각했었던 알파벳은 다시 생각하지 않는다.

   C. 이 쥐는 머리가 좋지 않아서 치즈를 딱 10번동안 알파벳을 생각해서 먹을 수 있고 그 이후에는 잠들어버린다. 이 쥐가 한번 치즈를 먹을 때, 어떤 치즈를 먹을지 생각하는 알파벳은 Random하게 생성하도록 한다. 그리고 한 번 먹을 때마다 이 때까지 생각했었던 알파벳을 출력하고, 원래의 치즈 문자열과 현재의 치즈 상태를 출력하도록 한다.

   D. 10번 이내에 쥐가 치즈를 다 먹었을 경우에는 'Out of cheese!'를 출력하고 프로그램을 종료하도록 하고, 10번 이내에 쥐가 치즈를 다 먹지 못했을 경우에는 남은 치즈의 상태를 출력하고 프로그램을 종료한다.

   E. 제출 파일: Python 소스 파일 1개 (파일 이름은 5.**py**)

실행 예제)

```
Input the length of the string : 20
Generated Cheese is 'jmeksuasctzmzxwloghx'
Mouse starts eating!!
Start eating 'z'
Eaten alphabet of cheese : z
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jmeksuasct_m_xwloghx

Start eating 'n'
Eaten alphabet of cheese : z n
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jmeksuasct_m_xwloghx

Start eating 'e'
Eaten alphabet of cheese : z n e
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm_ksuasct_m_xwloghx

Start eating 'c'
Eaten alphabet of cheese : z n e c
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm_ksuas_t_m_xwloghx

Start eating 'd'
Eaten alphabet of cheese : z n e c d
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm_ksuas_t_m_xwloghx

Start eating 'y'
Eaten alphabet of cheese : z n e c d y
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm_ksuas_t_m_xwloghx

Start eating 'g'
Eaten alphabet of cheese : z n e c d y g
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm_ksuas_t_m_xwlo_hx

Start eating 'p'
Eaten alphabet of cheese : z n e c d y g p
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm_ksuas_t_m_xwlo_hx

Start eating 'x'
Eaten alphabet of cheese : z n e c d y g p x
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm_ksuas_t_m__wlo_h_

Start eating 'k'
Eaten alphabet of cheese : z n e c d y g p x k
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm__suas_t_m__wlo_h_


Finally remained Cheese Status :
Eaten alphabet of cheese : z n e c d y g p x k
Original cheese : jmeksuasctzmzxwloghx
Current cheese status :  jm__suas_t_m__wlo_h_
```

F.