[2023년도 봄학기] **프로그래밍기초**

기말고사 라이브 코딩 (버전 1.0)

2023. 6. 8. 오후 7시 ~ 오후 9시 사이에 90분 @ 제1학술관 101, 102호

1. 인수분해 풀이과정 출력하기 [총 14점]

두수의 곱은 다음과 같이 인수분해를 하면 쉽게 계산할 수 있습니다.

$$21 \times 19 = (20 + 1) \times (20 - 1) = 20 \times 20 - 1 \times 1 = 400 - 1 = 399$$

두자리수 자연수 두개를 입력받아서 위와 같이 풀이과정을 출력해주는 프로그램을 작성 해봅시다.

먼저 두 수가 인수분해를 적용할 수 있는지, 그리고 인수분해를 적용하는게 도움이 될지를 확인하는 함수를 만들어 봅시다.

[2점] 1.1 평균값을 구하는 함수 median 을 완성하세요.

- 두 정수를 입력받아 두 정수의 평균값을 리턴합니다. 단 평균 값이 정수가 아닌 경우에는 -1을 리턴합니다.
  - 19와 21의 평균은 정수 20 이므로 20 을 리턴합니다.
  - 23과 22의 평균은 22.5 로 실수이므로 -1 을 리턴합니다.
  - 입력받은 두 정수는 모두 0 이상의 정수라고 가정합니다.

구현할 함수의 이름	**`median`**
구현할 함수의 인자 (파라미터)	정수 두 개
인자의 조건 (구현 필요)	없음
인자에 대한 가정 (구현 불필요)	두 인자는 모두 0 이상의 정수
리턴 값의 타입	`int`

```
>>> median(21, 19)
15
>>> median(23, 22)
-1
```

[4점] 1.2 is_balanced 함수를 완성하세요.

- 두 정수가 10의 배수 중 하나와 (정수만큼) 같은 거리에 있으면 True, 아니면 False 를 리턴합니다.
  - 예를 들어 19와 21은 10의 배수 20과 각각 1씩 차이나므로 True 를 리턴합니다.
  - 22와 20은 21과 각각 1씩 차이나지만, 21은 10의 배수가 아니므로 False 를 리턴합니다.
  - 23과 22는 조건을 만족하지 않으므로 False 를 리턴합니다.
- 입력받은 두 값 중 하나라도 조건에 맞는 정수(10이상, 99 이하의 자연수)가 아닐 경우 False 를 리턴합니다.

  (미 구현시 1점 감점)
- 1.1 에서 구현한 `median` 을 활용하세요.

구현할 함수의 이름	**`is_balanced`**
구현할 함수의 인자 (파라미터)	정수 두 개
인자의 조건 (구현 필요)	두 정수 모두 10이상 99이하인 두자리수의 자연수 이어야 함
인자에 대한 가정 (구현 불필요)	-
리턴 값의 타입	`bool`

```
>>> is_balanced(21, 19)
True
>>> is_balanced(20, 22)
False
```

[8점] 1.3 solve 함수를 완성하세요.

두 정수를 입력받아서 인수분해 과정을 `print` 함수로 출력하는 함수 `solve` 를 완성하세요.

구현할 함수의 이름	**`solve`**
구현할 함수의 인자 (파라미터)	정수 두 개
인자의 조건 (구현 필요)	`is_balanced` 를 만족해야 함
인자에 대한 가정 (구현 불필요)	-
리턴 값의 타입	`NoneType` (프로시저이므로 `return` 문이 없어야 함)

입력받은 두 정수가 a, b 일 때,

`is_balanced(a, b)` 가 `True` 이면 다음과 같이 풀이 과정을 출력합니다.

```
>>> solve(21, 19)
21 x 19
= ( 20 + 1 ) x ( 20 - 1 )
= 20 x 20 - 1 x 1
= 400 - 1
= 399
>>> solve(18, 22)
18 x 22
= ( 20 - 2 ) x ( 20 + 2 )
= 20 x 20 - 2 x 2
= 400 - 4
= 396
```

`is_balanced(a, b)` 가 `False` 이면 다음과 같이 출력합니다.

```
>>> solve(20, 22)
20 x 22 = ?
```

출력시 주의사항

- 출력하는 수식에서 곱하기 기호는 영문 소문자 x 를 사용합니다. 대문자나 다른 기호(*)를 사용할 경우 1점 감점입니다.
- `is_balanced` 가 `True` 인 경우, 풀이과정이 총 5줄에 걸쳐 출력되어야 합니다. 5줄이 아닐 경우 2점 감점입니다.
- `is_balanced` 가 `False` 인 경우, 한줄만 출력해야 합니다. 한줄이 아닐 경우 1점 감점입니다.

2. 색인 만들기 [총 16점]

하냥이는 작년 여름방학 동안 전자 도서를 관리하는 구텐베르크 사에 인턴으로 취직하였습니다. 그리고 회사에서 보유한 각 도서에서 가장 많이 나오는 영단어가 무엇인지 파악하는 임무를 맡게 되었습니다.

단어는 어느위치에 있느냐에 따라 대문자로 시작하기도 합니다. 예를 들어 문장의 맨 처음에 나오는 단어는 첫 문자를 대문자로 표현합니다. 단어가 몇 번 나오는지 셀 때, 대문자로 시작하는 단어와 소문자로 시작하는 단어를 다르게 취급하면 안되므로, 소문자로 바꾸는 작업을 해야 합니다.

[3점] 2.1 단어를 소문자로 바꾸기

문자열 하나를 입력받아서 모든 문자를 소문자로 변경한 문자열을 내어주는 함수 `lowercase` 를 완성하세요.

구현할 함수의 이름	**`lowercase`**
구현할 함수의 인자 (파라미터)	문자열 하나
인자의 조건 (구현 필요)	-
인자에 대한 가정 (구현 불필요)	ASCII 코드의 문자열
리턴 값의 타입	`str`

```
>>> lowercase("Apple")
apple
>>> lowercase("It's High Noon...")
"it's high noon..."
```

*** 도움말과 주의 사항**

- `ord`, `chr` 함수를 활용해보세요.
- 문자열의 메소드 `lower` 를 사용할 경우, 점수를 받을 수 없습니다. (3점 감점)

[5점] 2.2 단어 쪼개기

문자열을 입력받아서 단어의 리스트로 쪼개는 함수 `split_words` 을 완성하세요.

구현할 함수의 이름	**split_words**
구현할 함수의 인자 (파라미터)	문자열 하나
인자의 조건 (구현 필요)	-
인자에 대한 가정 (구현 불필요)	ASCII 코드의 문자열
리턴 값의 타입	list of str

```
>>> split_words("It's High Noon...")
['it', 'high', 'noon']
>>> split_words("Welcome to the summoner's Valley!")
['welcome', 'to', 'the', 'summoner', 'valley']
```

*** 도움말과 주의 사항**

- 영문 대소문자 (a-z, A-Z) 가 2번 이상 연속으로 나오는 경우만 단어로 취급합니다. 즉, a 나 I 등은 단어로 취급하지 않습니다. 만약 한글자인 영문자가 단어로 포함될 경우, 2점 감점입니다.
- 모든 단어는 소문자로 바꿔야 합니다. 그렇지 않을 경우 1점 감점입니다.

  2.1 lowercase 를 (제대로) 구현하지 못한 학생에 한해 아래 메소드의 사용을 허용합니다.

```
>>>"Welcome!".lower()
welcome!
>>>"It's High Noon...".lower()
it's high noon...
```

[4점] 2.3 단어 수 세기

파일이름을 입력받아서 단어의 수를 세어서 딕셔너리로 리턴하는 함수 `word_counter` 을 완성하세요.

구현할 함수의 이름	**`word_counter`**
구현할 함수의 인자 (파라미터)	파일 이름 (문자열)
인자의 조건 (구현 필요)	-
인자에 대한 가정 (구현 불필요)	파일은 항상 존재
리턴 값의 타입	`dict (키는 문자열, 값은 정수)`

```
>>> word_counter("sample2.txt")
{'not': 1, 'all': 2, 'that': 1, 'mrs': 1, 'bennet': 1, 'however': 1,
'with': 2, 'the': 4, 'assistance': 1, 'of': 4, 'her': 2, 'five': 1, 'daughters': 1,
'could': 1, 'ask': 1, 'on': 1, 'subject': 1, 'was': 1, 'sufficient': 1, 'to': 2,
'draw': 1, 'from': 1, 'husband': 1, 'any': 1, 'satisfactory': 1, 'description': 1,
'mr': 1, 'bingley': 1, 'they': 2, 'attacked': 1, 'him': 1, 'in': 1, 'various': 1,
'ways': 1, 'barefaced': 1, 'questions': 1, 'ingenious': 1, 'suppositions': 1, 'and': 2,
'distant': 1, 'surmises': 1, 'but': 1, 'he': 1, 'eluded': 1, 'skill': 1, 'them': 1,
'were': 1, 'at': 1, 'last': 1, 'obliged': 1, 'accept': 1, 'second': 1, 'hand': 1,
'intelligence': 1, 'their': 1, 'neighbour': 1, 'lady': 1}
```

*** 도움말과 주의 사항**

- 2.1, 2.2의 함수들을 사용합니다.
- 리턴하는 딕셔너리의 키는 단어(문자열), 값은 그 단어의 등장 횟수(정수) 입니다.

[4점] 2.4 최다 빈도 단어 수집하기

파일이름과 임의의 자연수 n 을 입력받아, 입력받은 파일에서 가장 많이 등장하는 단어 n 개를 찾아주는 함수 `most_used_words` 을 완성하세요.

구현할 함수의 이름	**`most_used_words`**
구현할 함수의 인자 (파라미터)	파일 이름 (문자열)과 정수
인자의 조건 (구현 필요)	정수 값은 1이상이어야 함 (1 미만일 경우 5로 간주)
인자에 대한 가정 (구현 불필요)	파일은 항상 존재
리턴 값의 타입	`list of tuple of (word and frequency)`

```
>>> booktitle = "prideandprejudice.txt"
>>> most_used_words(booktitle, 3)
[('the', 4521), ('to', 4246), ('of', 3735)]
```

*** 도움말과 주의 사항**

- 2.1, 2.2, 2.3 의 함수들을 활용하세요.
- 가장 많이 등장하는 순으로 단어와 등장한 빈도 수를 튜플로 묶은 리스트로 리턴합니다.
- 정수 값은 1이상이어야 하고, 1 미만의 값이 들어온 경우 기본값을 5로 가정합니다. (미구현시 1점 감점)
- 딕셔너리를 정렬할 때 `sorted` 함수를 사용하세요.
- 테스트에 사용한 '오만과 편견 (Pride and Prejudice)' 은 프로젝트 구텐베르크 홈페이지 (https://www.gutenberg.org) 에서 내려 받았습니다.