Книги

Берман Кеннеди Основы Python для Data Science. — СПб.: Питер, 2023

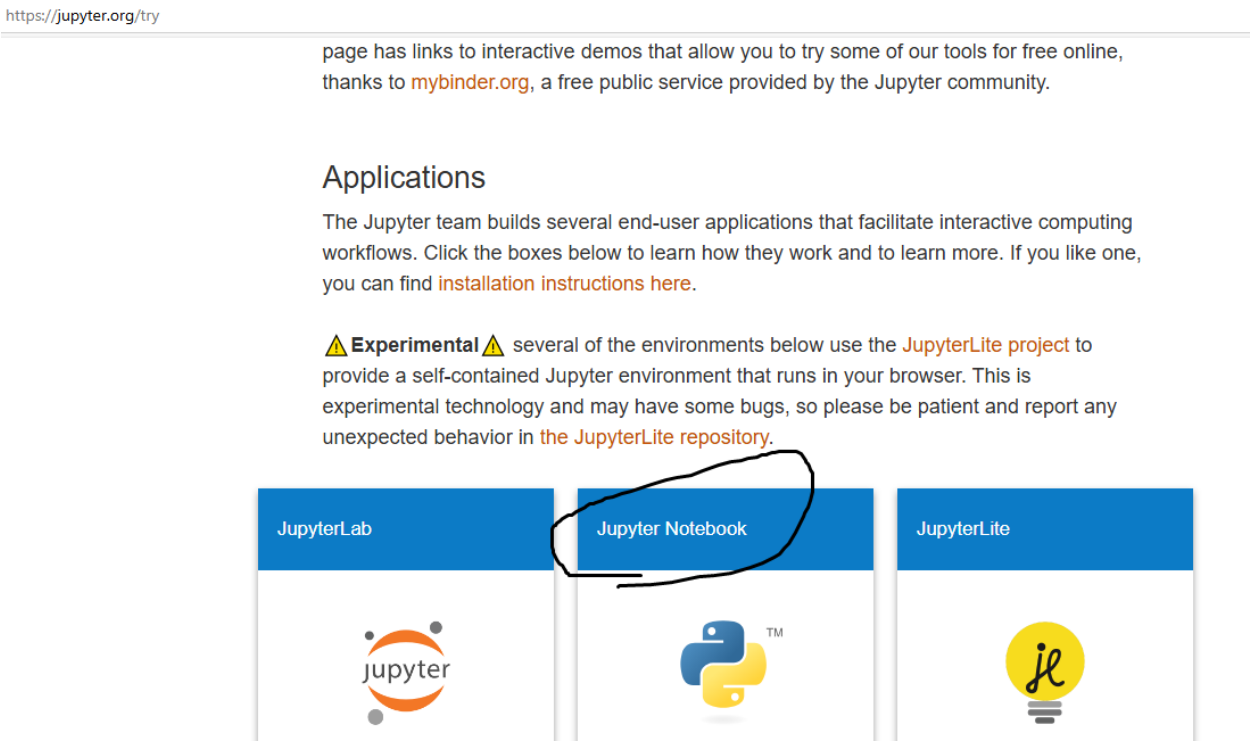
%CA%E5%ED%ED%E5%E4%E8%20%C1._%20%CE%F1%ED%EE%E2%FB%20Python%20%E4%EB%FF%20Data%20Science.(2023).pdf

Учебник Python

Python :: Учебник

Лабораторная работа №8: логические выражения и условные конструкции в Python

Project Jupyter | Try Jupyter или VStudio



Для выполнения этой работы необходимо:

- **понимать устройство и логику использования конструкции if-elif-else;**
- **уметь формулировать логические выражения с использованием операторов and и or.**

Задача считается решенной, если код успешно проходит автоматические тесты и не содержит грубых нарушений («подгонка» под тесты, копирование кода другого студента). «Подгонка» под тесты – написание кода не для общего случая, а для соответствия конкретным значениям в ячейке с тестами. Пример: пишем не универсальный код для проверки чётности числа, а код, который правильно работает только на числах 2, 5, 8, потому что именно эти числа используются в test_data .

Формат сдачи

Создать ipynb-файл можно, выбрав в Jupyter пункт меню *File* →... → *Notebook (.ipynb)*.

Тесты для проверки кода

Итак, для предварительной проверки задания нужно сделать следующее:

1. **Создать файл ipynb-файл в Jupyter Notebook.**
2. **Активировать тесты**
3. **Вставить решение каждой задачи в ячейку для кода, следующую за его условием, там, где написан комментарий # YOUR CODE HERE. Отступ вашего кода должен составлять 4 пробела.**
4. **Запустить ячейку, в которую вы вставили код с решением. Ввести какие-то входные данные, проверить визуально правильность вывода.**
5. **Запустить следующую ячейку (в ней содержится тест). Если запуск ячейки с тестом не приводит к появлению ошибки (assertion), значит, всё в порядке, задача решена. Если приводит к появлению ошибки, значит, тест не пройден и нужно искать ошибку.**

Внимание! Если в какой-то момент забыть ввести входные данные и перейти на следующую ячейку, есть риск, что Notebook перестанет откликаться. В этом случае надо перезапустить ядро: *Kernel → Restart*. При этом потеряются все значения переменных, но сам код останется.

Активировать тесты

Запустите следующую ячейку, чтобы иметь возможность запускать тесты.

In [ ]:

```
# Фабрика тестов для проверки программ, принимающих данные через input()


from collections import deque


class Tester(object):
    def __init__(self, inp):
        self.outputs = []
        self.inputs = deque(inp)
    def print(self, *args, sep = " ", end = "\n"):
        text = sep.join(map(str, args)) + end
        newlines = text.splitlines(keepends=True)
        if self.outputs and self.outputs[-1] and self.outputs[-1][-1] != "\n" and newlines:
            self.outputs[-1] += newlines[0]
            self.outputs.extend(newlines[1:])
        else:
            self.outputs.extend(newlines)


    def input(self, *args):
        assert self.inputs, "Вы пытаетесь считать больше элементов, чем предусмотрено условием"
```

```
        return self.inputs.popleft()
    def __enter__(self):
        global print
        global input
        print = self.print
        input = self.input
        return self.outputs
    def __exit__(self, *args):
        global print
        global input
        del print
        del input
```

Задача 1. Nel mezzo del cammin di nostra vita

Напишите программу, которая спрашивает у пользователя, в каком круге Ада, если верить «Божественной комедии» Данте, находится Платон, и:

- **если ответ верный (лимб), выводит на экран сообщение «Молодец!»;**
- **если ответ неверный (все остальные значения), выводит на экран сообщение «Неверно, перечитайте Данте!».**

Примеры работы программы

Ввод:

лимб

Вывод:

Молодец!

Ввод:

2

Вывод:

Неверно, перечитайте Данте!

Ввод:

третий

Вывод:

Неверно, перечитайте Данте!

In [ ]:

```
def dante():
```

```
    # YOUR CODE HERE
```

```
dante()
```

```
In [ ]:
```

```
test_data = [
    ("2", "Неверно, перечитайте Данте!"),
    ("лимб", "Молодец!"),
    ("3", "Неверно, перечитайте Данте!"),
    ("второй", "Неверно, перечитайте Данте!")
]


for inp, out in test_data:
    with Tester([inp]) as t:
        dante()
        assert len(t) == 1, "Вам нужно вывести ровно одну строку с ответом"
        assert t[0] == out+"\n", "Неверный ответ, была введена строка " + inp
```

Задача 2. Smoke on the water

Напишите программу, которая запрашивает с клавиатуры значение индекса качества воздуха (целое неотрицательное число) и выводит на экран текстовый комментарий согласно следующим критериям:

- **0–33: Very Good**
- **34–66: Good**
- **67–99: Fair**
- **100–149: Poor**
- **150–200: Very Poor**
- **200+: Hazardous**

Уточнения:

- **Строки с текстовыми комментариями уже сохранены в переменные very_good – hazardous.**
- **Гарантируется, что пользователь всегда вводит с клавиатуры целые неотрицательные числа.**

Примеры работы программы

Ввод:

33

Вывод:

Very Good

Ввод:

125

Вывод:

Poor

In [ ]:

```
def air_q():

    very_good = "Very Good"
    good = "Good"
    fair = "Fair"
    poor = "Poor"
    very_poor = "Very Poor"
    hazardous = "Hazardous"

    # YOUR CODE HERE


air_q()
```

In [ ]:

```
test_data = [
    ("0", "Very Good"),
    ("30", "Very Good"),
    ("33", "Very Good"),
    ("40", "Good"),
    ("56", "Good"),
    ("66", "Good"),
    ("67", "Fair"),
    ("80", "Fair"),
    ("99", "Fair"),
```

```
    ("102", "Poor"),
    ("149", "Poor"),
    ("150", "Very Poor"),
    ("198", "Very Poor"),
    ("200", "Very Poor"),
    ("201", "Hazardous"),
    ("230", "Hazardous"),
]


for inp, out in test_data:
    with Tester([inp]) as t:
        air_q()
        assert len(t) == 1, "Вам нужно вывести ровно одну строку с ответом"
        assert t[0] == out+"\n", "Неверный ответ, была введена строка " + inp
```

Задача 3. Свободу попугаям!

Попугай Кеша останется жить у Вовки только, если одновременно выполняются два условия:

- **громкость телевизора всегда будет выставлена не ниже 28 единиц;**
- **на диване или хотя бы в кресле можно лежать в кедах.**

Напишите программу, которая проверяет, останется ли Кеша у Вовки, а именно:

- **запрашивает с клавиатуры значение громкости телевизора (целое число);**
- **запрашивает с клавиатуры наличие разрешения лежать в кедах на диване (ответ yes или no);**
- **запрашивает с клавиатуры наличие разрешения лежать в кедах в кресле (ответ yes или no);**
- **если условия Кешу устраивают, выводит на экран сообщение «Вовка, ты знаешь...», если нет – «Goodbye, my love, goodbye!».**

Уточнения:

- **Строки с текстовыми комментариями уже сохранены в переменные stay и leave.**
- **Гарантируется, что пользователь всегда вводит с клавиатуры корректные значения (целое положительное число для громкости и слова yes или no).**

In [ ]:

```
def stay_or_not():


    stay = "Вовка, ты знаешь..."
```

```
    leave = "Goodbye, my love, goodbye!"


    # YOUR CODE HERE


stay_or_not()
In [ ]:
test_data = [
    ("28 no yes", "Вовка, ты знаешь..."),
    ("28 yes no", "Вовка, ты знаешь..."),
    ("28 yes yes", "Вовка, ты знаешь..."),
    ("28 no no", "Goodbye, my love, goodbye!"),
    ("25 yes yes", "Goodbye, my love, goodbye!"),
    ("32 no no", "Goodbye, my love, goodbye!"),
    ("30 yes no", "Вовка, ты знаешь..."),
    ("35 no yes", "Вовка, ты знаешь..."),
    ("24 yes yes", "Goodbye, my love, goodbye!")


]


for inp, out in test_data:
    with Tester(inp.split()) as t:
        stay_or_not()
        assert len(t) == 1, "Вам нужно вывести ровно одну строку с ответом"
        assert t[0] == out+"\n", "Неверный ответ, была введена строка " + inp
```

Задача 4. «Люблю тебя, Петра творенье...»

Напишите программу, которая запрашивает у пользователя информацию с клавиатуры и, в зависимости от того, что было введено, выводит на экран разные сообщения.

Правила:

- **Сначала пользователь вводит с клавиатуры название места, где именно в Петербурге он сейчас находится, затем – какой раз он посещает Санкт-Петербург (в виде целого числа).**
- **Если пользователь первый раз в Петербурге, на экран выводится сообщение «Добро пожаловать!», если не первый — «Рады видеть Вас снова!».**
- **Если пользователь находится в любом месте, кроме Канонерского острова (ответы Канонерский остров или Канонерка), то ему выводится сообщение «Ваша**

прическа: легкий бриз!», если на Канонерском острове — сообщение «Ваша прическа: ураган Сюрприз!».

Уточнения:

- **Строки с текстовыми комментариями уже сохранены в переменные hello1, hello2, verdict1 и verdict2.**
- **Гарантируется, что, пользователь, отвечая на вопрос о том, какой раз он посещает город, приводит корректное значение – целое положительное число.**

Примеры работы программы

Ввод:

Канонерский остров

1

Вывод:

Добро пожаловать!

Ваша прическа: ураган Сюрприз!

Ввод:

Канонерка

2

Вывод:

Рады видеть Вас снова!

Ваша прическа: ураган Сюрприз!

Ввод:

Заячий остров

2

Вывод:

Рады видеть Вас снова!

Ваша прическа: легкий бриз!

In [ ]:

```
def spb():


    # YOUR CODE HERE


spb()
```

In [ ]:

```
test_data = [
   ("Канонерский остров;1", ["Добро пожаловать!\n", "Ваша прическа: ураган Сюрприз!\n"]),
   ("Канонерка;2", ["Рады видеть Вас снова!\n", "Ваша прическа: ураган Сюрприз!\n"]),
   ("Заячий остров;2", ["Рады видеть Вас снова!\n", "Ваша прическа: легкий бриз!\n"]),
   ("Новая Голландия;1", ["Добро пожаловать!\n", "Ваша прическа: легкий бриз!\n"]),
   ("Пионерская площадь;3", ["Рады видеть Вас снова!\n", "Ваша прическа: легкий бриз!\n"]),
   ("Таврический сад;1", ["Добро пожаловать!\n", "Ваша прическа: легкий бриз!\n"])
]

for inp, out in test_data:
   with Tester(inp.split(";")) as t:
      spb()
      line_t = "".join(t)
      t = line_t.splitlines()
      assert len(t) == len(out), "Неверное количество строк в выводе"
      for l_test, l_out in zip(t, out):
         assert len(l_test.split()) == len(l_out.split()), \
            "Неверное количество элементов в строке " + l_out
         for el_test, el_out in zip(l_test.split(), l_out.split()):
            assert el_test == el_out, "Ошибка {} != {}".format(l_test,
                                                   l_out)
```

Лабораторная работа №9: методы на списках и методы на строках Python

Для выполнения этой работы необходимо:

- понимать устройство методов на разных объектах;
- уметь применять основные методы на списках и строках.

Будем считать основными следующие методы на списках: .append(), .extend(), .index(), .count(), .copy() и .sort().

Будем считать основными следующие методы на строках: .split(), .join(), .upper(), .lower(), .capitalize(), .replace(), .count(), .startswith(), .endswith().

Тесты для проверки кода

Итак, для предварительной проверки задания нужно сделать следующее:

1. Создать ipynb-файл на свой компьютер, открыть его в Jupyter Notebook.

   Project Jupyter | Home или VStudio

2. Вставить решение каждой задачи в ячейку для кода, следующую за его условием, там, где написан комментарий # YOUR CODE HERE. Отступ вашего кода должен составлять 4 пробела.

3. Запустить ячейку, в которую вы вставили код с решением. Ввести какие-то входные данные, проверить визуально правильность вывода.

4. Запустить следующую ячейку (в ней содержится тест). Если запуск ячейки с тестом не приводит к появлению ошибки (assertion), значит, всё в порядке, задача решена. Если приводит к появлению ошибки, значит, тест не пройден и нужно искать ошибку.

Внимание! Если в какой-то момент забыть ввести входные данные и перейти на следующую ячейку, есть риск, что Notebook перестанет откликаться. В этом случае надо перезапустить ядро: *Kernel* → *Restart*. При этом потеряются все значения переменных, но сам код останется.

Активировать тесты

Запустите следующую ячейку, чтобы иметь возможность запускать тесты.

In [ ]:

```
# Фабрика тестов для проверки программ, принимающих данные через input()


from collections import deque


class Tester(object):
    def __init__(self, inp):
        self.outputs = []
        self.inputs = deque(inp)
    def print(self, *args, sep = " ", end = "\n"):
        text = sep.join(map(str, args)) + end
        newlines = text.splitlines(keepends=True)
        if self.outputs and self.outputs[-1] and self.outputs[-1][-1] != "\n" and newlines:
            self.outputs[-1] += newlines[0]
            self.outputs.extend(newlines[1:])
        else:
            self.outputs.extend(newlines)


    def input(self, *args):
        assert self.inputs, "Вы пытаетесь считать больше элементов, чем предусмотрено условием"
```

```
        return self.inputs.popleft()
    def __enter__(self):
        global print
        global input
        print = self.print
        input = self.input
        return self.outputs
    def __exit__(self, *args):
        global print
        global input
        del print
        del input
```

Задача 1. Count on me

Напишите программу, которая:

- запрашивает у пользователя с клавиатуры строку с текстом
- с новой строки запрашивает символ (любой символ, буква, цифра, знак препинания, пробел)

и выводит на экран число вхождений этого символа во введенную строку с текстом.

Важно: если символ является буквой, число вхождений нужно считать для этой буквы как в нижнем, так и верхнем регистре (например, в тексте Text буква t должна быть найдена 2 раза, так как здесь есть и заглавная, и строчная буква t).

Пример работы программы:

Ввод:

text

t

Вывод:

2

Ввод:

Some Interesting Text Here

t

Вывод:

4

Ввод:

Some Interesting Text Here

T

Вывод:

4

Ввод:

allatambov@gmail.com

@

Вывод:

1

Ввод:

Слова, слова, слова

,

Вывод:

2

In [ ]:

```
def count_on_me():

    # YOUR CODE HERE


count_on_me()
```

In [ ]:

```
test_data = [
    ("text;t", "2"),
    ("Some Interesting Text Here;t", "4"),
    ("Some Interesting Text Here;T", "4"),
    ("allatambov@gmail.com;@", "1"),
    ("Слова, слова, слова;,", "2"),
    ("Без умолку безумная девица...;б", "2"),
    ("Кричала: «Ясно вижу Трою, павшей в прах!»;«", "1"),
```

```
    ("Но ясновидцев — впрочем, как и очевидцев —;е", "4"),
    ("Во все века сжигали люди на кострах.;В", "3"),
    ("Владимир Высоцкий;И", "3"),
    ("Песня о вещей Кассандре 1967;1", "1")
]

for inp, out in test_data:
    with Tester(inp.split(";")) as t:
        count_on_me()
        assert len(t) == 1, "Вам нужно вывести ровно одну строку с ответом"
        assert t[0] == out+"\n", "Неверный ответ, была введена строка " + inp
```

Задача 2. Sort it out

В списке files сохранены названия файлов с разными расширениями. Напишите программу, которая сохранит в список docs только файлы с расширением .doc или .docx. Элементы в списке docs должны быть отсортированы по алфавиту.

Примечание: при сортировке строк слова на латинице предшествуют словам на кириллице, это нормально.

In [ ]:

```
files = ["review.doc", "main.html", "mydoc.docx", "dreams-data.csv", "intro.R",
         "hw01.tex", "Заявление.docx", "Заявление(копия).docx", "Расписание.xlsx", "Сценарий.doc"]


# YOUR CODE HERE
```

In [ ]:

```
# здесь нет сложных тестов, просто проверяется соответствие списка docs нужному
# но список docs нужно создавать с помощью правильного кода!
# не просто скопировать список ниже :)


assert docs == ['mydoc.docx','review.doc','Заявление(копия).docx',
                'Заявление.docx','Сценарий.doc'], "Проверьте список docs"
```

Задача 3. Кортеж Его Высочества

Чиполлино попросил медведя, освобождённого им из зоопарка, собрать некоторую информацию о кортеже принца Лимона, который должен проехать вечером по центральной улице, но забыл, что тот не знает, как пишутся цифры. Тогда Чиполлино придумал такую систему обозначений:

- & – принц Лимон (носит плащ с длинным «хвостом»);

- 8 – министры (носят тугие ленты поверх мундира);
- 0 – придворные лимончики (без лент и плащей).

Так как порядок следования в кортеже строго регламентирован, на первом месте всегда следует принц, далее – министры, а затем уже придворные лимончики. Исходя из этого, у медведя могут получаться записи такого вида: & 88 0000 (принц, два министра, четверо придворных), & 8 00000(принц, министр, пятеро придворных), и так далее.

Напишите программу, которая запрашивает с клавиатуры строку, состоящую из символов, описанных выше, и пробелов, и возвращает кортеж из этих символов и чисел, которые соответствуют количеству лимонов, занимающих разные должности.

Ввод:

& 8 00000

Вывод:

('&', 1, '8', 1, '0', 5)

Ввод:

& 88 0000

Вывод:

('&', 1, '8', 2, '0', 4)

Ввод:

& 888 00

Вывод:

('&', 1, '8', 3, '0', 2)

In [ ]:

```
def lemongrass():


    # YOUR CODE HERE


lemongrass()

строка " + inp
```

Задача 4. Sorry, Cassandra

В строке song сохранён фрагмент песни "Cassandra" группы ABBA.

Напишите программу, которая посчитает встречаемость каждого слова в тексте и выведет на экран пары вида *слово : частота*, где слова отсортированы по алфавиту (см. формат вывода ниже). Приводить слова к начальной форме не нужно.

Требования к обработке song перед подсчётом слов:

- все слова должны быть приведены к нижнему регистру;
- сокращения должны быть раскрыты, то есть вместо 're должно быть are, а вместо didn't – did not;
- все знаки препинания должны быть удалены.

Все пары должны быть уникальными, то есть каждое слово в песне должно быть посчитано один раз.

aching : 1

again : 1

alive : 1

all : 1

alone : 1

and : 3

are : 2

as : 1

bargain : 1

bed : 1

behind : 1

believe : 1

believed : 1

but : 3

cassandra : 3

caught : 1

city : 1

crying : 1

darkest : 1

dawning : 1

day : 1

dead : 1

did : 1

down : 1

dreams : 1

fight : 1

final : 1

from : 1

had : 1

hearts : 1

hiding : 1

hollow : 1

hour : 1

how : 1

i : 3

in : 3

is : 2

it : 1

knew : 1

last : 1

laughter : 1

listen : 1

lost : 1

misunderstood : 1

must : 1

nights : 1

no : 1

nobody : 1

none : 1

not : 1

now : 2

of : 4

on : 2

one : 1

only : 1

our : 3

pity : 1

playing : 1

power : 1

really : 1

saw : 1

secrets : 1

sell : 1

shame : 1

shouting : 1

singing : 1

sleep : 1

smart : 1

some : 1

sorry : 2

start : 1

staying : 1

street : 1

suffer : 1

that : 1

the : 7

their : 1

then : 1

they : 1

though : 1

to : 2

until : 1

us : 2

wanted : 1

warning : 1

we : 2

weave : 1

were : 2

while : 1

words : 1

would : 2

you : 5

your : 1

Подсказка. Чтобы убрать в списке повторяющиеся значения, можно превратить его в множество. А множество при желании можно сделать списком или кортежем.

L = [1, 1, 5, 5, 5]

print(list(set(L)))

[1, 5]

In [ ]:

def cassandra():

song **=** """

Down in the street, they're all singing and shouting

Staying alive though the city is dead

Hiding their shame behind hollow laughter

While you are crying alone on your bed

Pity Cassandra that no one believed you

But then again you were lost from the start

Now we must suffer and sell our secrets

Bargain, playing smart, aching in our hearts

Sorry Cassandra, I misunderstood

Now the last day is dawning

Some of us wanted, but none of us would

Listen to words of warning

But on the darkest of nights

Nobody knew how to fight

And we were caught in our sleep

Sorry Cassandra, I didn't believe

You really had the power

I only saw it as dreams you would weave

Until the final hour

"""

 # YOUR CODE HERE

```
cassandra()
In [ ]:
test_data = [("",
            ['aching : 1\n','again : 1\n','alive : 1\n','all : 1\n',
            'alone : 1\n', 'and : 3\n','are : 2\n','as : 1\n',
             'bargain : 1\n','bed : 1\n','behind : 1\n',
             'believe : 1\n','believed : 1\n','but : 3\n',
             'cassandra : 3\n','caught : 1\n','city : 1\n',
             'crying : 1\n','darkest : 1\n','dawning : 1\n',
             'day : 1\n','dead : 1\n','did : 1\n','down : 1\n',
             'dreams : 1\n','fight : 1\n','final : 1\n','from : 1\n',
             'had : 1\n','hearts : 1\n','hiding : 1\n','hollow : 1\n',
             'hour : 1\n','how : 1\n','i : 3\n','in : 3\n','is : 2\n',
             'it : 1\n','knew : 1\n','last : 1\n','laughter : 1\n',
             'listen : 1\n','lost : 1\n', 'misunderstood : 1\n',
             'must : 1\n', 'nights : 1\n', 'no : 1\n', 'nobody : 1\n',
             'none : 1\n','not : 1\n','now : 2\n','of : 4\n','on : 2\n',
             'one : 1\n','only : 1\n','our : 3\n','pity : 1\n',
             'playing : 1\n','power : 1\n','really : 1\n','saw : 1\n',
             'secrets : 1\n','sell : 1\n','shame : 1\n','shouting : 1\n',
             'singing : 1\n','sleep : 1\n','smart : 1\n','some : 1\n',
             'sorry : 2\n','start : 1\n','staying : 1\n','street : 1\n',
             'suffer : 1\n','that : 1\n','the : 7\n','their : 1\n',
             'then : 1\n','they : 1\n','though : 1\n','to : 2\n',
             'until : 1\n','us : 2\n','wanted : 1\n','warning : 1\n',
             'we : 2\n','weave : 1\n','were : 2\n','while : 1\n',
             'words : 1\n','would : 2\n','you : 5\n','your : 1\n'])]

for inp, out in test_data:
    with Tester(inp.split()) as t:
        cassandra()
```

```
line_t = "".join(t)
t = line_t.splitlines()
assert len(t) == len(out), "Неверное количество строк в выводе"
for l_test, l_out in zip(t, out):
    assert len(l_test.split()) == len(l_out.split()),\
        "Неверное количество элементов в строке " + l_out
    for el_test, el_out in zip(l_test.split(), l_out.split()):
        assert el_test == el_out, "Ошибка {} != {}".format(l_test, el_out)
```

Лабораторная работа №10: функции и lambda-функции

Для выполнения этой работы необходимо уметь писать пользовательские функции.

Вставить решение каждой задачи в ячейку для кода, следующую за его условием, там, где написан комментарий # YOUR CODE HERE. Отступ вашего кода должен составлять 4 пробела.

Задача 1. Дивный новый мир

Напишите функцию hello_world(), которая принимает на вход имя человека и возвращает строку с приветствием, состоящую из слова Hello, имени, числа символов в имени и восклицательного знака в конце.

Примеры работы программы

Запуск функции:

hello_world("Alla")

Результат:

Hello, Alla4!

Запуск функции:

hello_world("Alligator")

Результат:

Hello, Alligator9!

Запуск функции:

hello_world("Python")

Результат:

Hello, Python6!

In [ ]:

 # YOUR CODE HERE

In [ ]:

assert hello_world("Alla") == "Hello, Alla4!", "Неверно"

assert hello_world("Python") == "Hello, Python6!", "Неверно"

assert hello_world("Alligator") == "Hello, Alligator9!", "Неверно"

assert hello_world("Cat") == "Hello, Cat3!", "Неверно"

Задача 2. Just function

Напишите функцию geom_mean(), которая принимает на вход три целых числа и возвращает их среднее геометрическое, округленное до второго знака после запятой. Если хотя бы одно из чисел отрицательно, функция возвращает пустой результат None.

Пример работы программы

Запуск функции:

geom_mean(1, 5, 8)

Результат:

6.32

Запуск функции:

geom_mean(0, 6, 7)

Результат:

0.0

Запуск функции:

geom_mean(-1, 5, 8)

Результат:

None

In [ ]:

 # YOUR CODE HERE

In [ ]:

assert geom_mean(0, 6, 7) == 0.0, "Неверно"

assert geom_mean(1, 5, 8) == 3.42, "Неверно"

assert geom_mean(25, 34, 100) == 43.97, "Неверно"

assert geom_mean(-1, 5, 8) == **None**, "Неверно"

assert geom_mean(-1, -5, 8) == **None**, "Неверно"

assert geom_mean(-1, -5, -8) == **None**, "Неверно"

assert geom_mean(25, 0, -100) **== None**, "Неверно"

Задача 3. Дикие лебеди

Элиза, чьих 11 братьев злая королева-мачеха превратила в диких лебедей, решила проверить, кто из лебедей действительно является её братом.

Реализуйте программу, которая определяет, является ли лебедь братом Элизы или нет, и возвращает список имён братьев. Определите функцию detector(), которая принимает два аргумента:

- brothers — список имён братьев (строки), который может быть пустым.
- new — новое имя (строка).

Если список brothers уже состоит из 11 элементов или если имя new в этом списке уже есть, функция возвращает кортеж из оригинального списка brothers и слова "пернатый". В обратном случае функция добавляет имя new в список brothers и возвращает кортеж из обновлённого списка brothers и слова "брат".

Примеры работы программы

Запуск функции:

detector(["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "R"], "лебедь")

Результат:

(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'R'], 'пернатый')

Запуск функции:

detector(["A", "B", "C", "D", "E", "F", "G", "H"], "лебедь")

Результат:

(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'лебедь'], 'брат')

Запуск функции:

detector(["A", "B", "C", "D", "E", "F", "G", "H"], "H")

Результат:

(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'], 'пернатый')

In [ ]:

 # YOUR CODE HERE

In [ ]:

assert detector(["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "R"], "лебедь") **==** (['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'R'], 'пернатый'), "Неверно"

assert detector(["A", "B", "C", "D", "E", "F", "G", "H"], "лебедь") **==** (['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'лебедь'], 'брат'), "Неверно"

assert detector(["A", "B", "C", "D", "E", "F", "G", "H"], "H") **==** (['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'], 'пернатый'), "Неверно"

Задача 4. Expecto problema

Используя lambda-функции, функции map() и filter() (это обязательное условие), реализуйте следующую программу, которая:

1. Запрашивает с клавиатуры целые числа через пробел – очки, набранные учениками Хогвартса за неделю, могут быть как отрицательными, так и неотрицательными.
2. На основе введённых значений формирует список целых чисел points.
3. На основе списка points создаёт список minus, содержащий только *отрицательные* значения.
4. Выводит на экран через пробел элементы minus, отсортированные по возрастанию.

In [ ]:

 # YOUR CODE HERE

Задача 5. Best to worst

Напишите функцию best_worst(), которая принимает на вход словарь с именами студентов и набранными ими баллами и возвращает кортеж из двух строк:

- строка с именами студентов, упорядоченными по убыванию в соответствии с суммой набранных ими баллов;
- строка с именами студентов, упорядоченными по убыванию в соответствии с баллом за последнюю работу (число работ любое больше 0).

Ниже приведён пример словаря для работы.

```
students = {
   "Анна" : [5, 10, 9, 7, 4],
   "Василий" : [10, 8, 2, 0, 5],
   "Владимир" : [10, 8, 9, 7, 10],
   "Ольга": [6, 8, 9, 10, 7],
   "Софья": [10, 9, 8, 6, 6],
   "Ян" : [2, 7, 8, 5, 9]
}
```

Пример работы программы

Запуск функции:

best_worst(students)

Результат:

('Владимир Ольга Софья Анна Ян Василий', 'Владимир Ян Ольга Софья Василий Анна')

Запуск функции:

best_worst({"Елена" : [5, 10, 9], "Владимир" : [10, 8, 7], "Ольга": [6, 8, 4]})

Результат:

('Владимир Елена Ольга', 'Елена Владимир Ольга')

In [ ]:

 # YOUR CODE HERE

In [ ]:

```
assert best_worst({"Елена" : [5, 10, 9],
                   "Владимир" : [10, 8, 7],
                   "Ольга": [6, 8, 4]}) == ('Владимир Елена Ольга', 'Елена Владимир Ольга')


assert best_worst({"Анна" : [5, 10, 9, 7, 4],
                   "Василий" : [10, 8, 2, 0, 5],
                   "Владимир" : [10, 8, 9, 7, 10],
                   "Ольга": [6, 8, 9, 10, 7],
                   "Софья": [10, 9, 8, 6, 6],
                   "Ян" : [2, 7, 8, 5, 9]}) == ('Владимир Ольга Софья Анна Ян Василий',
                                                'Владимир Ян Ольга Софья Василий Анна')
assert best_worst({"Alex" : [7, 10, 8],
                   "Sam" : [5, 6, 10],
                   "Pam" : [2, 0, 4],
                   "Harry": [7, 8, 10]}) == ('Alex Harry Sam Pam', 'Sam Harry Alex Pam')
```