Домашнее задание №1

Дедлайн 1 (16 баллов): 13 сентября, 23:59
Дедлайн 2 (8 баллов): 20 сентября, 23:59

Домашнее задание нужно написать на Python 3 и сдать в виде одного файла. Правило именования файла: `name_surname_01.py`. Например, если вас зовут Иван Петров, то имя файла должно быть: `ivan_petrov_01.py`.

Обратите внимание, что при выполнении задания можно использовать только то, что мы уже успели обсудить в рамках курса.

Индиана Джонс и яица Питона

1 Индиана Джонс в беде! После восьми лет поисков Джонс, наконец, добрался до яиц Питона, запрятанных в глубине лабиринта императора Рубиниуса. Коварный Рубиниус спроектировал свой лабиринт так, что как только яица Питона изменяют своё местоположение, лабиринт начинает разрушаться и через считанные минуты проваливается в центр Земли. Джонсу нужно помочь. Во что бы то ни стало!

a Чтобы помочь великому путешественнику, вам нужно сначала разобраться, как выглядит лабиринт Рубиниуса. Реализуйте функцию `print_map`, которая принимает на вход карту лабиринта в виде булевой матрицы и положение Джонса и выводит на экран ASCII изображение лабиринта в следующем формате:

- символ `"@"` соответствует ячейке, в которой находится Джонс;
- свободным ячейкам, для которых в матрице записано значение `True`, соответствует символ `"."`, занятым — `"#"`.

```
>>> def print_map(m, pos):
...     pass
...
>>> m = [[False, False, False, False],
...      [False, True, False, True],
...      [False, True, False, True],
...      [True, True, False, False]]
>>> print_map(m, (2, 1))
####
#.#.
#@#.
..##
```

Вам может быть полезна функция `shape`, возвращающая пару, первый элемент которой — количество строк в переданной матрице, а второй — количество столбцов.

```
>>> def shape(m):
...     return len(m), len(m[0])
...
>>> shape([[True, False]])
(1, 2)
>>> shape(m)
(4, 4)
```

б *Соседями* ячейки на карте лабиринта будем называть свободные ячейки, находящиеся слева, справа, сверху или снизу от рассматриваемой. Напишите функцию `neighbours`, которая принимает карту лабиринта и положение ячейки на ней и возвращает список её соседей.

```
>>> def neighbours(m, pos):
...     pass
...
>>> neighbours(m, (0, 0))
[]
>>> neighbours(m, (2, 1))
[(3, 1), (1, 1)]
```

в Реализуйте функцию `find_route`. Функция принимает карту лабиринта и положение на ней и возвращает **любой** путь до выхода из лабиринта. *Выходом* называется свободная ячейка, находящаяся на краю лабиринта.

```
>>> def find_route(m, initial):
...     pass
...
>>> find_route(m, (1, 1))
[(1, 1), (2, 1), (3, 1)]  # у вас может получиться другой ответ.
```

Для реализации вам может потребоваться метод `list.pop`, который удаляет из списка последний элемент и возвращает его в качестве результата.

```
>>> stack = [1]
>>> stack.append(2)
>>> stack.pop()
2
```

г Всё готово для того, чтобы помочь Джонсу! Напишите функцию `escape`, принимающую карту лабиринта и положение Джонса. В процессе работы функция должна найти путь до любого из выходов, а затем нарисовать последовательность шагов Джонса от начального положения до выхода.

```
>>> def escape(m, initial):
...     pass
...
>>> escape(m, (1, 1))
####
#@#.
#.#.
..##

####
#.#.
#@#.
..##

####
#.#.
#.#.
.@##
```

HBA1

2 Гемоглобин — животный белок, обеспечивающий перенос кислорода в различных тканях организма. Из-за своей важной функции структура гемоглобина похожа у разных видов.

В файле `HBA1.txt`[1] приведены последовательности гемоглобина в восьми различных видах: курица, мышь, горилла, кошка, кабан, корова, крыса, человек.

Определите номера строк, которые по вашему мнению соответствуют человеку и гориллe с помощью расстояния Хемминга. Расстояние Хемминга между двумя строчками одинаковой длины определяется как число позиций, в которых они различаются.

```
>>> def hamming(seq1, seq2):
...     pass
...
>>> hamming("foo", "boo")
1
>>> def hba1(path, distance):
...     pass
...
>>> hba1("./HBA1.txt", hamming)
(42, 24)  # у вас должен получиться другой ответ.
```

3 Другой способ посчитать расстояние между двумя строчками основан на сравнении частот всех существующих подстрок фиксированной длины в одной строке и в другой.

а Напишите функцию `kmers`, которая по заданной строке `seq` и положительному числу `k` строит *словарь частот*, где ключи — подстроки длины `k`, а значения — количество раз, которое соответствующая подстрока встречается в `seq`.

```
>>> def kmers(seq, k=2):
...     pass
...
>>> kmers("abracadabra", 2)
{'ca': 1, 'ra': 2, 'ad': 1, 'ac': 1, 'br': 2, 'ab': 2, 'da': 1}
```

б Напишите функцию `distance1`, которая принимает две строки и положительное число `k`, строит для каждой строки словарь частот, а затем вычисляет между полученными словарями метрику L_1. Метрика L_1 определяется как

$$L_1(p, q) = \sum_{k \in K} \left| p_k - q_k \right| \quad K = \text{keys}(p) \cup \text{keys}(q).$$

Функция, вычисляющая модуль числа в Python называется `abs`.

```
>>> def distance(seq1, seq2):
...     pass  # можно считать, что k = 2.
...
>>> distance1("abracadabra", "abracadabra")
0
>>> distance1("abracadabra", "anaconda")
13
>>> distance1("abracadabra", "abra")
7
```

[1] https://gist.github.com/superbobry/360faa9af122bc4c5f5e

в Убедитесь, что функция `hba1` с метрикой L_1 и $k = 2$ даёт результат, аналогичный результату из предыдущего задания.