**Лабораторная работа №3**

Содержание

[Урок 18. Задачи: 4](#_Toc21809302)

[18.1. Улыбайтесь, господа! 4](#_Toc21809303)

[18.2. Скажи «пароль» и проходи 5](#_Toc21809304)

[18.3. Золотое сечение 6](#_Toc21809305)

[18.4. Правильная скобочная последовательность 7](#_Toc21809306)

[18.5. Уравнение прямой 8](#_Toc21809307)

[18.6. Точка на прямой 9](#_Toc21809308)

[Урок 19. Задачи: 10](#_Toc21809309)

[19.1. Месяц/Month 10](#_Toc21809310)

[19.2. Ход конём 11](#_Toc21809311)

[19.3. Уравнения степени не выше второй 12](#_Toc21809312)

[19.4. Палиндромы 13](#_Toc21809313)

[19.5. Простые числа 14](#_Toc21809314)

[19.6. Числа Каталана 15](#_Toc21809315)

[Урок 20. Задачи: 16](#_Toc21809316)

[20.1. НРЗБРЧВ 16](#_Toc21809317)

[20.2. Бюрократия 17](#_Toc21809318)

[20.3. Несвежие анекдоты 18](#_Toc21809319)

[20.4. Айболит 19](#_Toc21809320)

[20.5. Счастливый пассажир 20](#_Toc21809321)

[20.6. Делайте ваши ставки 21](#_Toc21809322)

[Урок 21. Задачи: 22](#_Toc21809323)

[21.1. Числа в строке 22](#_Toc21809324)

[21.2. Транспонирование 23](#_Toc21809325)

[21.3. Обмен личностями 24](#_Toc21809326)

[21.4. Фрактальный список – 2 25](#_Toc21809327)

[21.5. Печать фрактала 26](#_Toc21809328)

[21.6. Фрактальное дерево 27](#_Toc21809329)

[Урок 22. Задачи: 28](#_Toc21809330)

[22.1. Цезарь 28](#_Toc21809331)

[22.2. Частичные суммы 30](#_Toc21809332)

[22.3. Дартс 31](#_Toc21809333)

[22.4. Уравнения степени не выше второй — часть 2 32](#_Toc21809334)

[22.5. Уравнения степени не выше второй — часть 3 33](#_Toc21809335)

[Урок 23. Задачи: 34](#_Toc21809336)

[23.1. Мимикрия 34](#_Toc21809337)

[23.2. Самая далёкая планета 35](#_Toc21809338)

[23.3. Пам-парам парам-пам парам 36](#_Toc21809339)

[23.4. Астроида 37](#_Toc21809340)

[23.5. Все равны, как на подбор 38](#_Toc21809341)

[Урок 24. Задачи: 39](#_Toc21809342)

[24.1. Есть ли 0 39](#_Toc21809343)

[24.2. Гематрия по-английски 40](#_Toc21809344)

[24.3. Оформленные комментарии 41](#_Toc21809345)

[24.4. Ох уж эти анаграммы 42](#_Toc21809346)

**Урок 18. Задачи:**

# 18.1. Улыбайтесь, господа!

Вася любит в чате использовать смайлики, но поскольку в терминале встроенных смайликов нет, он попросил вас написать набор функций, которые помогут ему печатать сложные смайлики с символами юникода, которых нет на клавиатуре. Он показал вам три смайла, которые хотел бы в первую очередь.

Напишите три функции:  
**print\_shrug\_smile()**, которая печатает ¯\\_(ツ)\_/¯   
**print\_ktulhu\_smile()**, которая печатает {:€   
и **print\_happy\_smile()**, которая печатает (͡° ͜ʖ ͡°)

Обратите внимание, что вам не нужно в программе эти функции вызывать. Следите за тем, чтобы имена функций были написаны верно.

Пример

| Ввод | Вывод |
| --- | --- |
| print\_shrug\_smile()  print\_ktulhu\_smile()  print\_happy\_smile() | ¯\\_(ツ)\_/¯  {:€  (͡° ͜ʖ ͡°) |

# 18.2. Скажи «пароль» и проходи

Напишите функцию ask\_password(), которая запрашивает у пользователя пароль и сверяет его со строкой, в которой записано слово “password”. Пользователю дается три попытки. Как только пароль совпал с правильным значением, функция должна выводить «Пароль принят» и игнорировать дальнейший ввод. Если с трех попыток пользователь не смог угадать пароль, функция должна вывести на экран «В доступе отказано» и игнорировать ввод новых паролей.

Обратите внимание: в вашей программе должна быть функция ask\_password, но она не должна вызываться. Следите за тем, чтобы имя функции было написано верно.

Тестирующая программа выглядит следующим образом:

ask\_password()

Пример 1

| Ввод | Вывод |
| --- | --- |
| qwerty  1234  йцукен | В доступе отказано |

Пример 2

| Ввод | Вывод |
| --- | --- |
| qwerty  1234  password | Пароль принят |

# 18.3. Золотое сечение

Вам необходимо написать функцию **golden\_ratio(i)**, которая получает число i как аргумент и печатает i-ое последовательное приближение золотого сечения.

i-ое приближения золотого сечения вычисляется как отношение (i+1)-го числа Фибоначчи к i-му.

Числами Фибоначчи называется последовательность, первые два элемента которой (нумерация с единицы) равны 1, а каждое из последующих чисел равно сумме двух предыдущих. Первые члены последовательности Фибоначчи таковы:

https://contest.yandex.ru/testsys/tex/render/MSwxLDIsMyw1LDgsMTMsMjEsMzQsNTUsXGRvdHM=.png

Таким образом первое приближение золотого сечения — 1/1, второе — 2/1, третье — 3/2, четвертое — 5/3 и т. д.

Следите за правильностью названия функции.

Пример 1

| Ввод | Вывод |
| --- | --- |
| golden\_ratio(1) | 1.0 |

Пример 2

| Ввод | Вывод |
| --- | --- |
| golden\_ratio(2) | 2.0 |

Пример 3

| Ввод | Вывод |
| --- | --- |
| golden\_ratio(4) | 1.6666666666666667 |

# 18.4. Правильная скобочная последовательность

Правильной скобочной последовательностью называется строка, состоящая только из символов «скобки» (открывающих "(" и закрывающих ")"), где каждой закрывающей скобке найдётся соответствующая открывающая. Например, () и (()()) — правильные последовательности, а (()(() или )( — нет.

Напишите функцию **bracket\_check(test\_string)**, которая проверяет, является ли поступившая на вход строка правильной скобочной последовательностью. Если да, она должна печатать YES, в противном случае — NO. Обратите внимание, что пустая строка также является правильной скобочной последовательностью.

Пример 1

| Ввод | Вывод |
| --- | --- |
| bracket\_check("()") | YES |

Пример 2

| Ввод | Вывод |
| --- | --- |
| bracket\_check("(()((") | NO |

# 18.5. Уравнение прямой

Как известно, через любые две различные точки на плоскости проходит прямая, и только одна. Напишите функцию **equation(a, b)**, которая по двум заданным точкам находит уравнение прямой, проходящей через них.

В функцию передаются две строки — координаты точек в формате **x;y**, функция должна выводить на экран два числа через пробел — коэффициенты k и b найденной прямой.

Если в решении получается прямая вида y=c или x=c, функция должна печатать эту константу с.

Пример 1

| Ввод | Вывод |
| --- | --- |
| equation("0;0", "1;1") | 1.0 0.0 |

Пример 2

| Ввод | Вывод |
| --- | --- |
| equation("0;0", "0;4") | 0.0 |

Пример 3

| Ввод | Вывод |
| --- | --- |
| equation("4;6.9", "-5.2;6.9") | 6.9 |

# 18.6. Точка на прямой

Напишите функцию **line(s, t)**, которая получает на вход две строки: **s** — уравнение прямой в виде kx+b и **t** — координаты точки на плоскости в виде x;y. Ваша программа должна печатать True, если точка лежит на прямой, и False в противном случае.

Пример 1

| Ввод | Вывод |
| --- | --- |
| line("1x+6", "1;7") | True |

Пример 2

| Ввод | Вывод |
| --- | --- |
| line("5x-10", "5;-9") | False |

Пример 3

| Ввод | Вывод |
| --- | --- |
| line("0x+7", "3;7") | True |

Пример 4

| Ввод | Вывод |
| --- | --- |
| line("3.5x+0", "2;7") | True |

**Урок 19. Задачи:**

# 19.1. Месяц/Month

Напишите функцию, которая принимает номер месяца и язык (русский или английский), а возвращает его название.

Пример 1

| Ввод | 2 |
| --- | --- |
| print(month\_name(3, "en")) | march |

Пример 2

| Ввод | Вывод |
| --- | --- |
| print(month\_name(3, "ru")) | март |

# 19.2. Ход конём

Вам дана строка, содержащая координату клетки на шахматном поле (например, “A2”).

Напишите функцию possible\_turns(cell), которая будет по такой строке возвращать *список*координат (в такой же текстовой форме) всех клеток, на которые может попасть конь с указанной клетки за один ход. Не забудьте, что за пределы шахматного поля коню ходить нельзя.

Рекомендация:

Cделайте две вспомогательные функции, которые будут по строке делать целочисленные координаты и наоборот. Например, клетке “A2” можно сопоставить кортеж (1,2), а кортежу (8,8) — клетку “H8”. То есть, одна функция принимает строку, а возвращает кортеж из двух чисел. Другая функция принимает кортеж, возвращает строку.

Также полезно сделать функцию, которая по кортежу целочисленных координат сообщает, находится ли это клетка внутри шахматного поля или снаружи.

Проверяется только функция possible\_turns. Вспомогательные функции не проверяются, но они помогут вам написать простое и компактное решение.

Пример 1

| Ввод | Вывод |
| --- | --- |
| print(possible\_turns("B1")) | ['A3', 'C3', 'D2'] |

Пример 2

| Ввод | Вывод |
| --- | --- |
| print(possible\_turns("H8")) | ['F7', 'G6'] |

Примечания

Возвращаемый список нужно отсортировать по алфавиту.

# 19.3. Уравнения степени не выше второй

Напишите функцию roots\_of\_quadratic\_equation(a,b,c), которая принимает параметры a,b,c и возвращает решения уравнения ax2 + bx + c = 0.

Возвращаемое значение у функции ровно одно, а корней может быть от нуля до двух (есть еще один случай: a = b = c = 0, при котором решением является любое число; про него ниже), поэтому сделать так, чтобы эта функция возвращала просто значение корня, не получится.

Вместо этого вы должны вернуть из функции список, содержащий значения всех корней уравнения (одно, два или ни одного). Если корнем является любое значение x, верните вместо списка чисел – список, содержащий одну строку “all” (["all"])

Не забудьте рассмотреть случаи, когда квадратное уравнение вырождается до линейного (a = 0).  
Также бывает, что линейное уравнение вырождается еще дальше – до уравнения, в котором x не участвует вовсе (a = b = 0). В этом случае мы получаем уравнение c = 0, которое, в зависимости от значения c, либо не имеет корней, либо корнем является любой x.

Пример 1

| Ввод | Вывод |
| --- | --- |
| result = roots\_of\_quadratic\_equation(1, 2, 1)  print(\*sorted(result)) | -1 |

Пример 2

| Ввод | Вывод |
| --- | --- |
| result = roots\_of\_quadratic\_equation(1, -3, 2)  print(\*sorted(result)) | 1 2 |

# 19.4. Палиндромы

Палиндром — это строка (или слово), которая читается одинаково в обоих направлениях, например, «12321» или известный пример Афанасия Фета «А роза упала на лапу Азора».

Напишите функцию **palindrome(s)**, которая принимает в качества параметра строку и определяет, является ли она палиндромом. Функция должна возвращать строку «Палиндром» или «Не палиндром» соответственно.

Подсказка: в этой задаче вам пригодятся функции для работы со строками из предыдущего урока.

Пример 1

| Ввод | Вывод |
| --- | --- |
| print(palindrome('12321')) | Палиндром |

Пример 2

| Ввод | Вывод |
| --- | --- |
| print(palindrome('Палиндром')) | Не палиндром |

Примечания

При проверке считайте большие и маленькие буквы одинаковыми, а также игнорируйте пробелы.

# 19.5. Простые числа

Еще в 300 году до н.э. Евклид доказал, что простых чисел бесконечно много, однако задача поиска простых чисел актуальна и по сей день: каждый год математики находят новые, все большие простые числа.

Напишите функцию **prime(number)**, в которую передается натуральное число, большее единицы.   
Функция должна возвращать строку **«Простое число»** в случае, если оно простое, и строку **«Составное число»** в противном случае.

Пример 1

| Ввод | Вывод |
| --- | --- |
| print(prime(4)) | Составное число |

Пример 2

| Ввод | Вывод |
| --- | --- |
| print(prime(3)) | Простое число |

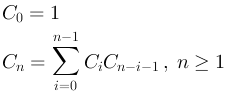
Примечания

Напомним, что простым числом называется такое натуральное число, которые делится только на единицу и на себя.   
Единица, при этом, не является ни простым, ни составным числом.

# 19.6. Числа Каталана

Числа Каталана — это числовая последовательность, которая встречается во многих комбинаторных задачах, например, в задаче про правильную скобочную последовательность: n-ое число Каталана — это количество правильных скобочных последовательностей длины 2n, то есть последовательностей из n левых и n правых скобок. Например, если n=3, то таких последовательностей всего 5: ()()(), (())(), ()(()), (()()), ((())).

Существует несколько формул для вычисления чисел Каталана. Например, они могут быть найдены, используя следующее рекуррентное соотношение:



Напишите функцию **catalan(n)**, возвращающую n-ое число Каталана (обратите внимание, что числа нумеруются с нуля!).

Пример 1

| Ввод | Вывод |
| --- | --- |
| print(catalan(0)) | 1 |

Пример 2

| Ввод | Вывод |
| --- | --- |
| print(catalan(1)) | 1 |

**Урок 20. Задачи:**

# 20.1. НРЗБРЧВ

В письменности языка НРЗБРЧВ отсутствуют гласные буквы и знаки препинания, в остальном язык ничем не отличается от русского языка. Чтобы из слова русского языка получить слово языка НРЗБРЧВ, достаточно выкинуть из слова все гласные. Чтобы перевести предложение, нужно перевести каждое слово и выкинуть знаки препинания. Регистр букв следует оставить неизменным.   
Напишите функцию **translate(text)**, которая переводит текст с русского языка на НРЗБРЧВ.  
Свой результат функция должна записать во внешнюю переменную **translatedText**.

Формат ввода

translatedText = None  
translate("Удивительный факт, но текст на языке НЕРАЗБОРЧИВО оказывается довольно просто читать. Достаточно небольшой тренировки - и вы сможете это делать.")

Формат вывода

translatedText == "двтльнй фкт н ткст н зк НРЗБРЧВ кзвтс двльн прст чтть Дсттчн нбльшй трнрвк в смжт т длть"

Примечания

Слова в НРЗБРЧВ разделяются ровно одним пробелом. Обратите внимание, что при переводе некоторые короткие слова «съедаются» полностью. Следите за тем, чтобы в этом случае не появлялись лишние пробелы.

# 20.2. Бюрократия

Отдел кадров одной крупной компании осознал, что тратит много времени на заполнение одних и тех же данных в куче разных бумажек для заявки на отпуск. Они попросили вас упростить их работу, чтобы им приходилось только один раз ввести имя, фамилию и некоторые вспомогательные данные. А сразу несколько справок печатать подстановкой указанных данных в разные шаблоны.

Чтобы облегчить страдания бюрократов, вы взялись им помочь. Они и раньше использовали Python для печати документов, но для каждого документа они передавали все параметры раз за разом — примерно так:

print\_application\_for\_leave("Иван Петров", "1 июня – 20 июня")  
print\_holiday\_money\_claim("Иван Петров", "15 тысяч пиастров")  
print\_attorney\_letter("Иван Петров", "1 июня – 20 июня", "Василий Васильев")

Больше всего отдел кадров раздражало, что приходится повторять параметры в каждом документе, к тому же есть риск ошибиться. Можно было бы решить эту задачу с помощью введения переменных, но набор документов нужен каждый раз немного разный (иногда какой-то документ не нужен, иногда наоборот нужно несколько копий). Кроме того, даже с использованием переменных функциям все равно пришлось бы указывать кучу параметров и не путать, какие параметры функции нужны, а какие — необязательны. Да и работать с переменными отдел кадров пока не умеет.

Вы сошлись на том, что теперь отделу кадров придется заполнять данные, вызывая функции языка Python с правильными параметрами. Одна функция — **setup\_profile(name, vacationDates)** — предназначена для того, чтобы задать всю информацию про работника. А ещё несколько функций пользуются установленными значениями, чтобы оформить документы на отпуск: заявление на отпуск **print\_application\_for\_leave()**, заявление на выплату отпускных **print\_holiday\_money\_claim(amount)**, и доверенность на передачу своих служебных полномочий заместителю **print\_attorney\_letter(toWhom)**.

Напишите эти четыре функции, так чтобы каждая печатала соответствующую справку. Примеры справок приведены в тестовом выводе. Не ошибайтесь в точных формулировках документов, бюрократы это не любят.

Пример

| Ввод | Вывод |
| --- | --- |
| setup\_profile("Иван Петров", "1 июня – 20 июня")  print\_application\_for\_leave()  print\_application\_for\_leave()  print\_holiday\_money\_claim("15 тысяч пиастров")  print\_attorney\_letter("Василий Васильев") | Заявление на отпуск в период 1 июня – 20 июня. Иван Петров  Заявление на отпуск в период 1 июня – 20 июня. Иван Петров  Прошу выплатить 15 тысяч пиастров отпускных денег. Иван Петров  На время отпуска в период 1 июня – 20 июня моим заместителем назначается Василий Васильев. Иван Петров |

Примечания

Если setup\_profile вызвана второй раз, необходимо перейти к документам второго сотрудника и т. д.

# 20.3. Несвежие анекдоты

Ваш друг любит пересылать вам в чат анекдоты и байки, но часто забывает, какие анекдоты он уже рассказал, причём теми же словами дословно. Чтобы не отвлекаться от важных дел на старые анекдоты, вы решили написать функцию **print\_only\_new(message)**, которая печатает сообщение только если вы такое сообщение ещё никогда не получали.

Пример

| Ввод | Вывод |
| --- | --- |
| print\_only\_new('Шутка номер 15')  print\_only\_new('Шутка номер 23')  print\_only\_new('Шутка номер 24')  print\_only\_new('Шутка номер 24')  print\_only\_new('Шутка номер 100')  print\_only\_new('Шутка номер 24')  print\_only\_new('Шутка номер 99')  print\_only\_new('Шутка номер 15')  print\_only\_new('Шутка номер 100') | Шутка номер 15  Шутка номер 23  Шутка номер 24  Шутка номер 100  Шутка номер 99 |

# 20.4. Айболит

На прием к доктору Айболиту как всегда большая очередь. Так как желающих очень много, то было принято решение в регистратуре открыть еще два окна, при этом написать программу для автоматизации приема больных.

Напишите функцию **hello(name)**, которая принимает на вход имя пациента, приветствует его и сообщает, что его медицинскую карту ищут.

Для поиска медицинской карты напишите функцию **search\_card(name)**, которая тоже принимает на вход имя пациента. Эта функция выводит информацию о номере карточки пациента или о том, что она не найдена. Список имён пациентов, пронумерованный с 1, находится в списке **base**, который является глобальным. Номер пациента соответствует номеру карты.

Пример

| Ввод | Вывод |
| --- | --- |
| base = ["Иван", "Юлия Иванкова"]  hello("Иван")  search\_card("Иван")  hello("Юлия Иванова")  search\_card("Юлия Иванова") | Здравствуйте, Иван! Вашу карту ищут...  Ваша карта с номером 1 найдена  Здравствуйте, Юлия Иванова! Вашу карту ищут...  Ваша карта не найдена |

# 20.5. Счастливый пассажир

Пассажир считается счастливым, если его текущий и предыдущий билеты на поездку в транспорте являются счастливыми. Напишите функцию **lucky(ticket)**, которая проверяет этот факт и возвращает соответствующую строку.

Переменная ticket содержит в себе целое число.  
В глобальной переменной **lastTicket** находится номер предыдущего билета пассажира.

Напомним, что билет является счастливым, если сумма первых трёх цифр совпадает с суммой трёх последних.

Пример 1

| Ввод | Вывод |
| --- | --- |
| lastTicket = 123456  print(lucky(100001)) | Несчастливый |

Пример 2

| Ввод | Вывод |
| --- | --- |
| lastTicket = 123321  print(lucky(100001)) | Счастливый |

Примечания

Если в переменной ticket находится не шестизначное число, то его следует дополнить нулями слева.

# 20.6. Делайте ваши ставки

Боб хочет собрать всех своих друзей-предпринимателей вместе. Но для того, чтобы заставить занятых друзей прийти к нему, следует сделать что-то необычное. У Боба есть ферма, на которой живут 10 лошадей. Он хочет провести скачки со ставками.

Помогите Бобу написать функцию **do\_bet()** для совершения ставок, которая первым аргументом принимает номер лошади, а вторым –- ставку. Лошади пронумерованы от 1 до 10. На одну и ту же лошадь нельзя сделать ставку дважды. Также нельзя делать нулевую ставку.

Пример

| Ввод | Вывод |
| --- | --- |
| do\_bet(1, 10)  do\_bet(1, 100)  do\_bet(2, 0)  do\_bet(2, 200) | Ваша ставка в размере 10 на лошадь 1 принята  Что-то пошло не так, попробуйте еще раз  Что-то пошло не так, попробуйте еще раз  Ваша ставка в размере 200 на лошадь 2 принята |

**Урок 21. Задачи:**

# 21.1. Числа в строке

Напишите функцию from\_string\_to\_list(string, container), которая принимает два аргумента: строку string, состоящую из целых чисел, написанных через пробел, и список container. Функция должна извлечь из строки числа и добавить их в конец списка.

Пример 1

| Ввод | Вывод |
| --- | --- |
| a = [1, 2, 3]  from\_string\_to\_list("1 3 99 52", a)  print(\*a) | 1 2 3 1 3 99 52 |

Пример 2

| Ввод | Вывод |
| --- | --- |
| a = [77, 'abc']  from\_string\_to\_list("", a)  print(\*a) | 77 abc |

Примечания

В тестовую систему вы должны отправить файл только с функцией from\_string\_to\_list(string, container). Если в файле есть код вызова данной функции, то закомментируйте его. Выводить на экран ничего не нужно.

# 21.2. Транспонирование

Напишите функцию transpose(matrix), которая принимает матрицу N × M (т.е. список из N списков-строк по M элементов каждый) и транспонирует её, т.е. превращает строки в столбцы и наоборот.

i-ая строка при этом превращается в i-й столбец.

Пример 1

| Ввод | Вывод |
| --- | --- |
| matrix = [[1]]  transpose(matrix)  for line in matrix:  print(\*line) | 1 |

Пример 2

| Ввод | Вывод |
| --- | --- |
| matrix = [[1, 2], [3, 4]]  transpose(matrix)  for line in matrix:  print(\*line) | 1 3  2 4 |

# 21.3. Обмен личностями

Для хитроумного приспособления, которое обменивает людей личностями, вы должны написать функцию, которая принимает список мыслей одного человека и другого, а затем обменивает их содержимым.

Функция должна называться swap(first, second). Первый список должен получить содержимое второго списка, и наоборот.

Пример 1

| Ввод | Вывод |
| --- | --- |
| first = [1, 2, 3]  second = [4, 5, 6]  first\_content = first[:]  second\_content = second[:]  swap(first, second)  print(first, second\_content, first == second\_content)  print(second, first\_content, second == first\_content) | [4, 5, 6] [4, 5, 6] True  [1, 2, 3] [1, 2, 3] True |

Пример 2

| Ввод | Вывод |
| --- | --- |
| first = [1, 2, 3]  second = [4, 5, 6, 7]  first\_content = first[:]  second\_content = second[:]  swap(first, second)  print(first, second\_content, first == second\_content)  print(second, first\_content, second == first\_content) | [4, 5, 6, 7] [4, 5, 6, 7] True  [1, 2, 3] [1, 2, 3] True |

# 21.4. Фрактальный список – 2

Напишите функцию defractalize(fractal), которая принимает в качестве аргумента фрактальный список и удаляет из него все самоподобные части фрактала (т.е. части, эквивалентные самому фракталу).  
Фрактальный список в этой задаче может содержать в качестве элементов любые числа и сам себя (в любом числе копий).

Пример 1

| Ввод | Вывод |
| --- | --- |
| fractal = [2, 5]  fractal.append(fractal)  fractal.append(3)  defractalize(fractal)  print(fractal) | [2, 5, 3] |

Пример 2

| Ввод | Вывод |
| --- | --- |
| fractal = [2, 5]  fractal.append(fractal)  fractal.append(3)  fractal.append(fractal)  fractal.append(9)  defractalize(fractal)  print(fractal) | [2, 5, 3, 9] |

# 21.5. Печать фрактала

Когда встроенная функция print распечатывает список, содержащий сам себя, то вместо содержимого внутреннего списка она подставляет многоточие.

Это позволяет программе не зациклиться, пытаясь напечатать список, который за счет фрактальной структуры имеет бесконечную глубину.

Например, текстовое представление списка, состоящего из нуля, самого себя и двойки будет выглядеть так:

[0, [...], 2]

Напишите функцию fractal\_print(obj), которая также выводит фрактал на экран, но печатает не один, а два уровня вложенности:

[0, [0, [...], 2], 2]

Мы рассматриваем только простые фрактальные списки, состоящие из обычных несоставных объектов (числа, строки, логические величины) или самих себя.

Пример 1

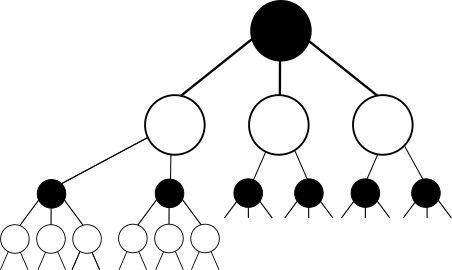
| Ввод | Вывод |
| --- | --- |
| fractal = [3]  fractal.append(fractal)  fractal\_print(fractal) | [3, [3, [...]]] |

Пример 2

| Ввод | Вывод |
| --- | --- |
| fractal = [3]  fractal.append(fractal)  fractal.append(2)  fractal\_print(fractal) | [3, [3, [...], 2], 2] |

# 21.6. Фрактальное дерево

Создайте «фрактальное дерево» (и сохраните в глобальной переменной wb\_tree) с чёрными и белыми вершинами (и корнем).  
Поясним немного, что такое дерево в понимании программиста:   
Обычно (укоренённым) деревом называется иерархическая структура, состоящая из набора объектов (вершин), расположенных иерархически. У любой вершины может быть несколько «детей» – других вершин. Если у вершины нет дочерних вершин, она называется листом. У каждой вершины, кроме корня, есть ровно один родитель, у корня родителя нет. Кроме того, часто вершины деревьев раскрашивают для добавления им дополнительных характеристик. В нашем случае вершина дерева может иметь один из двух цветов: белый или чёрный.



Мы будем представлять каждую вершину в виде списка её дочерних элементов. Таким образом, это дерево будет список списков списков списков...  
Пустое «белое» или «чёрное» дерево выглядит так: black = [].   
А дерево с «чёрной» вершиной – так: tree = black.

Обычно любое дерево имеет конечное число элементов. Вы же должны сделать имитацию «бесконечно-глубокого» дерева, имеющего повторяющуюся фрактальную структуру.   
Корень дерева – чёрный.  
Каждая белая вершина содержит два чёрных поддерева.  
Каждая чёрная вершина – три белых поддерева.

Белое и чёрное поддеревья везде одни и те же.

Примечания

В этой задаче вы должны ввести код, который создает глобальную переменную wb\_tree с правильным содержимым. Как и в прочих задачах, где не сказано обратного, ваша программа не должна ничего выводить на экран.

**Урок 22. Задачи:**

# 22.1. Цезарь

Напишите функцию encrypt\_caesar(msg, shift), которая кодирует сообщение шифром Цезаря и возвращает его. Шифр Цезаря заменяет каждую букву в тексте на букву, которая отстоит в алфавите на некоторое фиксированное число позиций.

В функцию передается сообщение и сдвиг алфавита. Если сдвиг не указан, то пусть ваша функция кодирует сдвиг алфавита на 3 позиции:

А →Г,

Б →Д,

В →Е,

…

Э →А,

Ю →Б,

Я →В

Все символы, кроме русских букв должны остаться неизменными. Маленькие буквы должны превращаться в маленькие, большие — в большие.

Напишите также функцию декодирования decrypt\_caesar(msg, shift), также использующую сдвиг по умолчанию. При написании функции декодирования используйте вашу функцию кодирования.

Пример 1

| Ввод | Вывод |
| --- | --- |
| msg = "Да здравствует салат Цезарь!"  shift = 3  encrypted = encrypt\_caesar(msg, shift)  decrypted = decrypt\_caesar(encrypted, shift)  print(encrypted)  print(decrypted) | Зг кзугефхецих фгогх Щикгуя!  Да здравствует салат Цезарь! |

Пример 2

| Ввод | Вывод |
| --- | --- |
| msg = "Да здравствует салат Цезарь!"  shift = 5  encrypted = encrypt\_caesar(msg, shift)  decrypted = decrypt\_caesar(encrypted, shift)  print(encrypted)  print(decrypted) | Йе мйхезцчзшкч цереч Ыкмехб!  Да здравствует салат Цезарь! |

Примечания

Символы русского алфавита расположены в стандартной для Python таблице кодировки подряд, то есть номера, выдаваемые функцией ord(symbol), идут подряд.

Буква «ё» идёт в таблице кодировки отдельно от основного алфавита. При решении задачи считайте, что буквы «ё» в русском алфавите нет.

# 22.2. Частичные суммы

k-ой частичной суммой списка называется сумма первых k элементов списка. Напишите функцию partial\_sums, которая принимает неограниченное число аргументов, а возвращает список частичных сумм этих элементов: на нулевой позиции — 0, на первой позиции — первое число, на второй — сумму первого и второго чисел, затем — сумму первого, второго и третьего и т.д.

Обратите внимание, что функция должна принимать не список, а именно неограниченное число аргументов.

Пример 1

| Ввод | Вывод |
| --- | --- |
| print(partial\_sums(13)) | [0, 13] |

Пример 2

| Ввод | Вывод |
| --- | --- |
| print(partial\_sums(1, 0.5, 0.25, 0.125)) | [0, 1, 1.5, 1.75, 1.875] |

# 22.3. Дартс

Стандартная мишень для игры в дартс разделена на 20 ячеек с номерами от 1 до 20. В центре расположено «яблочко», попадание в которое приносит игроку 50 очков. Вокруг него — зелёное кольцо, при попадании в которое засчитывается 25 очков. Попадание во внешнее (узкое) кольцо мишени удваивает число сектора, а во внутреннее — утраивает.

Эти правила показались игрокам слишком простыми, поэтому они решили присвоить секторам во внешнем и внутреннем кольцах случайные значения. В глобальной переменной **scoring** хранится словарь для подсчета очков (обратите внимание, что в случае внутреннего и внешнего колец значениями являются словари, ключами в которых являются номера сектора, а значениями –– количество очков):

Яблочко: 50

Зеленое кольцо: 25

Внешнее кольцо: 1: 8, 2: 6, 3: 42,…, 20: 50

Внутреннее кольцо: 1: 2, 2: 9, 3: 56,…, 20: 3

Напишите функцию **score()**, которая принимает на вход 1 (если это «Яблочко» или «Зеленое кольцо») или 2 аргумента (если это внутреннее или внешнее кольцо, то название кольца и номер сектора) и возвращает количество очков.

Пример 1

| Ввод | Вывод |
| --- | --- |
| print(score("Яблочко")) | 50 |

Пример 2

| Ввод | Вывод |
| --- | --- |
| print(score("Внешнее\_кольцо", 1)) | 8 |

# 22.4. Уравнения степени не выше второй — часть 2

Сделайте функцию solve(\*coefficients), которая умеет решать уравнения степени не выше второй (квадратные и линейные).

* Если у функции три аргумента, их надо трактовать как a, b и c в уравнении ax2+bx+c=0.
* Если два — как коэффициенты b и c в уравнении bx+c=0.
* Если один — как коэффициент c в уравнении c = 0.
* Если список коэффициентов пуст или коэффициентов больше трёх, то функция должна вернуть None.

Значения корней возвращайте в том же виде, как их возвращала функция roots\_of\_quadratic\_equation(a,b,c) из задачи «Уравнения степени не выше второй» Для решения можно как переписать ту функцию, так и использовать её внутри новой функции.

Пример 1

| Ввод | Вывод |
| --- | --- |
| print(sorted(solve(1, 2, 1))) | [-1.0] |

Пример 2

| Ввод | Вывод |
| --- | --- |
| print(sorted(solve(1, -3, 2))) | [1.0, 2.0] |

# 22.5. Уравнения степени не выше второй — часть 3

Напишите программу, которая запрашивает у пользователя строку с коэффициентами уравнения (степени не выше второй), затем печатает корни уравнения.

Коэффициенты уравнения вводятся в одну строку в порядке от наибольшей степени неизвестной к свободному члену, через пробел. Их количество заранее неизвестно: от 3 до 1 коэффициентов – для квадратного уравнения, линейного уравнения и уравнения, не зависящего от x вовсе – аналогично прошлым частям этой задачи.

Ответ выводится на экран так же, в одну строку через пробел.

При решении задачи используйте написанную в прошлой задаче функцию solve(\*coefficients).

Обратите внимание, что функции solve и print могут принимать списки аргументов различной длины. Благодаря этому вам не потребуется разбирать случаи различного числа коэффициентов и различного числа корней, а решение можно записать буквально в одну строку.

Пример 1

| Ввод | Вывод |
| --- | --- |
| 1 2 1 | -1 |

Пример 2

| Ввод | Вывод |
| --- | --- |
| 1 -3 2 | 1 2 |

**Урок 23. Задачи:**

# 23.1. Мимикрия

У вас есть код, который вы не можете менять (так часто бывает, когда код в глубине программы используется множество раз и вы не хотите ничего сломать):

transformation = <???>   
values = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29] *#*илилюбойдругойсписок   
transformed\_values = **list**(**map**(transformation, values))

Единственный способ вашего взаимодействия с этим кодом – посредством задания функции transformation.

Однако вы поняли, что для вашей текущей задачи вам не нужно никак преобразовывать список значений, а нужно получить его как есть.

Напишите такое лямбда-выражение transformation, чтобы transformed\_valuesполучился копией values. Переменная transformation должна быть глобальной, чтобы проверяющая система смогла его найти. Кроме transformation вам ничего писать не нужно. Печатать на экран – тоже.

Пример

| Ввод | Вывод |
| --- | --- |
| values = [1, 23, 42, "asdfg"]  transformed\_values = list(map(transformation, values))  if values == transformed\_values:  print('ok')  else:  print('fail') | ok |

# 23.2. Самая далёкая планета

Планеты вращаются вокруг звезд по эллиптическим орбитам. Назовём самой далёкой планетой ту, орбита которой имеет самую большую площадь.

Напишите функцию find\_farthest\_orbit(list\_of\_orbits), которая среди списка орбит планет найдет ту, по которой вращается самая далёкая планета. Круговые орбиты не учитывайте: вы знаете, что у вашей звезды таких планет нет, зато искусственные спутники были запущены на круговые орбиты.

Результатом функции должен быть кортеж, содержащий длины полуосей эллипса орбиты самой далёкой планеты.

Каждая орбита представляет из себя кортеж из пары чисел – полуосей её эллипса. Площадь эллипса вычисляется по формуле S = πab, где a и b – длины полуосей эллипса.

При решении задачи используйте списочные выражения.

Подсказка: проще всего будет найти эллипс в два шага: сначала вычислить самую большую площадь эллипса, а затем найти и сам эллипс, имеющий такую площадь.

Гарантируется, что самая далёкая планета ровно одна.

Пример

| Ввод | Вывод |
| --- | --- |
| orbits = [(1, 3), (2.5, 10), (7, 2), (6, 6), (4, 3)]  print(\*find\_farthest\_orbit(orbits)) | 2.5 10 |

# 23.3. Пам-парам парам-пам парам

Винни-Пух попросил вас посмотреть, есть ли в его стихах ритм. Поскольку разобраться в его кричалках не настолько просто, насколько легко он их придумывает, вам стоит написать программу.

Винни-Пух считает, что ритм есть, если число слогов (т.е. число гласных букв) в каждой фразе стихотворения одинаковое. Фраза может состоять из одного слова, если во фразе несколько слов, то они разделяются дефисами. Фразы отделяются друг от друга пробелами.

Стихотворение Винни-Пух вбивает в программу с клавиатуры.  
В ответе напишите «Парам пам-пам», если с ритмом всё в порядке и «Пам парам», если с ритмом всё не в порядке.

При решении используйте списочные выражения, либо функции высшего порядка в сочетании с лямбда-функциями.

Пример

| Ввод | Вывод |
| --- | --- |
| пара-ра-рам рам-пам-папам па-ра-па-дам | Парам пам-пам |

# 23.4. Астроида

Астроидой называется кривая на плоскости, описываемая параметрическими уравнениями

https://contest.yandex.ru/testsys/tex/render/eD1SXGNvcyBeezN9dA==.png

https://contest.yandex.ru/testsys/tex/render/eT1SXHNpbiBeezN9dA==.png

при https://contest.yandex.ru/testsys/tex/render/dFxpblswOzJccGld.png.

То есть каждая точка (x,y) астроиды получается по вышеприведённым формулам при подстановке некоторого t.

Наша астроида имеет R=1.

Перебирая точки для различных значений t найдите расстояние от точки (0.75, 0) до ближайшей точки астроиды. Дайте ответ с точностью 10-4 (полученное вами расстояние должно отличаться от правильного ответа не больше, чем на 10-4).

# 23.5. Все равны, как на подбор

Напишите функцию same\_by(characteristic, objects), которая проверяет, все ли объекты имеют одинаковое значение некоторой характеристики, и возвращает **True**, если это так. Если значение характеристики для разных объектов отличается – то **False**. Для пустого набора объектов, функция должна возвращать True. Аргумент characteristic – это функция, которая принимает объект и вычисляет его характеристику.

Пример 1

| Ввод | Вывод |
| --- | --- |
| values = [0, 2, 10, 6]  if same\_by(lambda x: x % 2, values):  print('same')  else:  print('different') | same |

Пример 2

| Ввод | Вывод |
| --- | --- |
| values = [1, 2, 3, 4]  if same\_by(lambda x: x % 2, values):  print('same')  else:  print('different') | different |

**Урок 24. Задачи:**

# 24.1. Есть ли 0

Напишите программу, которая ищет нули в таблице чисел и печатает True, если нули нашлись.

В противном случае надо напечатать False.

Эту задачу надо постараться решить «в одну строчку». В этом вам помогут функции any и all.

Формат ввода

Текст c матрицей (таблицей) целых чисел в диапазоне от 0 до 99, разделённых пробелами и символами перевода строки (см. пример).

Формат вывода

True или False

Пример

| Ввод | Вывод |
| --- | --- |
| 64 33 79 56 78 70 45 71 82 3  96 27 8 36 72 14 91 10 21 65  95 28 91 23 78 38 21 50 64 37  97 54 94 6 48 17 37 19 78 58  69 58 35 1 70 24 60 17 3 11  48 9 13 23 82 49 79 55 29 53  9 2 67 90 0 17 34 55 49 63  98 98 23 71 66 57 15 94 34 81  58 37 32 29 10 19 53 46 95 19  41 24 95 47 58 17 74 69 62 4 | True |

# 24.2. Гематрия по-английски

Словесной гематрией называется сумма номеров (кодов, числовых значений) входящих в слово букв.

На вход программы поступает список английских слов. На одной строке записано одно слово, количество слов неизвестно.

Для вычисления гематрии поступим следующим образом:

1. Переведём слово в верхний регистр.
2. Числовое значение буквы вычислим как **КодБуквы - КодБуквыA** **+ 1**

Выведите полученные слова в порядке возрастания их гематрии. Если для каких-то слов гематрия совпадает, то их выводите в алфавитном порядке.

Формат ввода

Набор слов на английском языке, каждое слово на отдельной строке.

Формат вывода

Набор слов в требуемом порядке.

Пример 1

| Ввод | Вывод |
| --- | --- |
| mother  Daddy  sIster | Daddy  mother  sIster |

Пример 2

| Ввод | Вывод |
| --- | --- |
| bBb  aaaaaa  word | aaaaaa  bBb  word |

Примечания

Для получения кода символа воспользуйтесь функцией ord(c).

Слова во входных данных могут повторяться.

# 24.3. Оформленные комментарии

При помощи итераторов и функций высшего порядка выведите список комментариев с указанием номера строки у каждого из них (нумерация строк с единицы).  
  
Знаки решетки и пробелы в начале строки (а также в начале комментария – после символа решётки) отбросьте. Также отбросьте пробелы и символы табуляции, если они встречаются в конце строки.

Оформление строки вывода сделайте аналогично примеру. Пробел между двоеточием и комментарием не должен зависеть от содержания комментария (так как лидирующие и замыкающие пробелы в строке комментария отбрасываются).

Пример

| Ввод | Вывод |
| --- | --- |
| import sys  for line in sys.stdin:  # rstrip(’\n’) "отрезает" от строки line,  # идущий справа символ перевода строки,  # ведь print сам переводит строку  print(line.rstrip(’\n’)) | Line 3: rstrip(’\n’) "отрезает" от строки line,  Line 4: идущий справа символ перевода строки,  Line 5: ведь print сам переводит строку |

# 24.4. Ох уж эти анаграммы

Анагра́мма (от греч. ανα- — «пере» и γράμμα — «буква») — литературный приём, состоящий в перестановке букв или звуков определённого слова (или словосочетания), что в результате даёт другое слово или словосочетание (Википедия).

Мы будем работать с набором слов. Ваша задача — выписать все слова, которые являются анаграммами друг для друга, например «замок» и «мазок». Проверка слов должна быть регистронезависимой. Слова, для которых анаграммой является только оно само, выписывать не нужно, даже если это слово встречается в тексте в разном регистре.

Формат ввода

В первой строке идёт целое число n (1 ≤ n ≤ 100 000), количество исходных слов.

Далее следует n слов, по одному слову в строке, слова могут идти в разном регистре!

Формат вывода

В одной строке должны идти слова, которые являются анаграммами друг для друга, в нижнем регистре, через пробел. Порядок слов — лексикографический (как в словаре). Порядок строк так же лексикографический.

Пример

| Ввод | Вывод |
| --- | --- |
| 11  окорок  петлей  Плетей  рококо  теплей  Тишь  ТОМНО  тонко  тонок  тоном  шить | окорок рококо  петлей плетей теплей  тишь шить  томно тоном  тонко тонок |