# Программирование на Python
# Январь 2025
# Практические задания по основам Python

## Постановка задачи

Требуется разработать программы согласно указанным ниже заданиям (на выбор, но можно реализовать свою тему). Решения должны быть хорошо структурированы путем разбивки на модули, а внутри модулей – на функции.

Клиентский код должен импортировать вашу библиотеку и посредством объектов воспользоваться реализуемой в них функциональностью. В тексте необходимо использовать комментарии, поясняющие ваши действия и принятые решения.

Объем задания (конкретные указания, что надо выполнить) указан ниже. Программное решение общего задания требуется представить в виде отдельных решений по каждому заданию.

## Задание 1: Типы данных, ввод/вывод, арифметические операции и модуль math

Спасатель должен добраться до утопающего как можно быстрее, чтобы не допустить его гибели. Поскольку спасатель находится на берегу, а утопающий — в воде, то спасателю необходимо преодолеть часть дистанции по песку, а оставшуюся часть — вплавь. Направление движения, выбранное спасателем, определяет общее расстояние, которое ему необходимо будет преодолеть, поскольку скорость движения в воде меньше скорости движения по суше на определённую постоянную величину. Вам необходимо написать программу, которая бы рассчитывала время, которое требуется спасателю для того, чтобы добраться до утопающего. Используемые обозначения и геометрическое построение, иллюстрирующее задачу, приведены на Рисунке 1.

Напишите программу, которая запрашивает у пользователя 6 значений:

- Кратчайшее расстояние от спасателя до кромки воды, $d_1$ (в ярдах)
- Кратчайшее расстояние от утопающего до берега, $d_2$ (в футах)
- Боковое смещение между спасателем и утопающим, $h$ (в ярдах)
- Скорость движения спасателя по песку, $v_{sand}$ (в милях в час)
- Коэффициент замедления спасателя при движении в воде, $n$
- Направление движения спасателя по песку, $\theta_1$ (в градусах)

При выполнении вычислений все значения должны быть преобразованы в тип с плавающей точкой. Необходимо принять следующие соотношения: в одном ярде три фута, а в одной миле 5280 футов. Не забывайте, что тригонометрические функции из модуля `math` предполагают аргументы, выраженные в радианах. Используя следующие формулы, напишите код, вычисляющий общее время, необходимое спасателю для достижения утопающего:

$$x = d_1 \cdot tan\,(\theta_1)$$

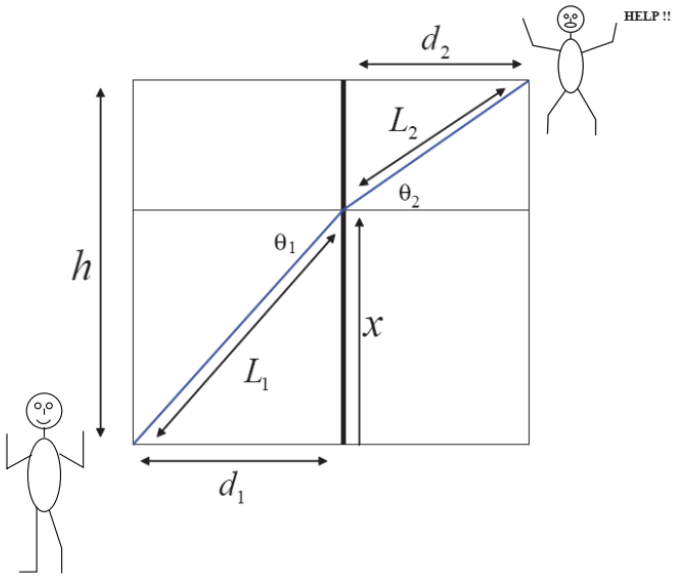


Рис. 1: Задача спасения утопающего (источник изображения: Massachusetts Institute of Technology, `https://ocw.mit.edu/courses/mechanical-engineering/2-71-optics-spring-2009/assignments/MIT2_71S09_usol1.pdf`)

$$L_1 = \sqrt{x^2 + d_1^2}$$

$$L_2 = \sqrt{(h - x)^2 + d_2^2}$$

Если мы обозначим скорость движения по песку как $v_{sand}$, то скорость движения в воде будет равна $v_{swim} = \frac{v_{sand}}{n}$. Общее время на достижение утопающего может быть найдено по следующей формуле:

$$t = \frac{1}{v_{sand}} (L_1 + nL_2)$$

В выводе программы значение направление (угла) должно быть представлено в виде целого числа, а значение времени — в виде десятичной дроби с одной цифрой после десятичной запятой.

Ниже приведён пример запуска программы и вывода:

```
Введите кратчайшее расстояние между спасателем и кромкой воды, d1 (ярды) => 8
8
Введите кратчайшее расстояние от утопающего до берега, d2 (футы) => 10
10
Введите боковое смещение между спасателем и утопающим, h (ярды) => 50
50
Введите скорость движения спасателя по песку, v_sand (мили в час) => 5
5
```

```
Введите коэффициент замедления спасателя при движении в воде, n => 2
2
Введите направление движения спасателя по песку, theta1 (градусы) => 39.413
39.413
Если спасатель начнёт движение под углом theta1, равным 39 градусам, он
достигнет утопащего через 39.9 секунды
```

### Задание 2: Функции, модульные тесты

Перепишите код из Задания 1, разделив его на функции таким образом, чтобы каждая функция имела строго определённую зону ответственности - взаимодействие с пользователем или выполнение вычислений.

Создайте необходимый набор модульных тестов, разместив их в глобальной области основного кода программы.

### Задание 3: Циклы

Перепишите код из Задания 2, используя циклы для подбора оптимального значения угла, под которым необходимо начать движение спасателю.

Решите данную задачу аналитически (выведите формулу зависимости угла от остальных параметров) и сравните её решение с решением, полученным числовым способом при помощи вашей программы.

### Задание 4: Циклы, скорость выполнения операций, сравнение производительности

Реализуйте один и тот же эталонной алгоритм (benchmark) DGEMM (`https://iq.opengenus.org/dgemm/`) на следующих языках программирования:

- C# (два подварианта - с использованием примитивных типов и с использованием больших чисел)
- Java (два подварианта - с использованием примитивных типов и с использованием больших чисел)
- Python

Необходимо обеспечить выполнение следующих требований:

- Все реализации получают исходные данные из одних и тех же файлов. Входные файлы должны быть подготовлены заранее при помощи отдельной программы-генератора. Пользователю должен быть предоставлена возможность указания диапазона значений элементов матриц. Результаты выполнения программы также записываются в файлы.
- Для каждого языка необходимо сделать реализации без использования многопоточности и с использованием многопоточности (количество потоков должно задавать в качестве параметра).

- Для каждого языка необходимо сделать реализации как со значением с плавающей точкой двойной точности (как и предусмотрено эталоном DGEMM), так и с целыми значениями элементов матриц.

Напишите программу на языке Python, которая представляет собой платформу для проведения экспериментов над Вашими реализациями DGEMM. Необходимо обеспечить выполнение следующих требований:

- Возможность многократного запуска на выполнение любого исследуемого кода с передачей ему необходимых параметров командной строки.
- Подсчёт базовой статистики по времени выполнения (минимальное, максимальное и среднее время, среднеквадратическое отклонение, медиана).
- Сохранение всех данных эксперимента в виде файла CSV.
- Построение графиков со статистическими данными времени выполнения и сохранение графиков в виде файлов.

Произведите тестирование ваших реализаций DGEMM, запуская тестовую платформу с параметрами, обеспечивающее выполнение каждой исследуемой реализации одинаковое количество раз. Проанализируйте полученные результаты. На основе Вашего анализа, сделайте выводы о производительности различных числовых типов данных в разных языках программирования.

### Задание 5: Работа со сторонними модулями, чтение и запись файлов

Разработайте реализацию игры Жизнь, работающую в консольном режиме. Программа должна прочитать входной файл с начальной конфигурацией поля. Количество шагов моделирования задаётся в качестве параметра. Программа должна выполнить моделирование колонии организмов в соответствии с правилами игры Жизнь, записывая в файл каждую новую конфигурацию, а также создавая изображение (снимок) состояния поля в виде отдельного файла PNG. Для работы с изображениями необходимо использовать библиотеку PIL (Pillow).

Базовый цвет “живой” ячейки должен задаваться в качестве параметра (это может быть только “чистый цвет”). Программа будет использовать оттенки этого базового цвета чтобы указать на “возраст” ячейки.

### Задание 6: Создание консольных приложений вида REPL

В рамках политики открытых данных правительства многих стран предоставляют свободный доступ к большому количеству наборов данных. Один из таких наборов данных содержит географические координаты всех почтовых индексов США. В данном задании мы воспользуемся этим набором данных. Почтовый индекс в США представляет собой 5 десятичных цифр. В наборе данных для каждого индекса приводятся географические координаты (широта и долгота), название города, штата и графства. Нашей целью является использовать предоставленные нам данные и разработать приложение, позволяющее пользователю не только выполнять запросы или поиск в существующем источнике данных, но также получить доступ к дополнительной функциональности, например, вычислению расстояния между двумя географическими точками.

Напишите программу, которая позволит пользователям находить местоположение по почтовому индексу, находить почтовый индекс по городу и штату, а также определять расстояние между двумя точками, заданными их почтовыми индексами. Программа будет взаимодействовать с пользователем в режиме REPL (read-evaluate-print), а именно путем выдачи приглашения к вводу команды и затем получения команды, которую пользователь введёт с клавиатуры. Выход из цикла взаимодействия с пользователем будет происходить при вводе им команды `'end'`. В этом случае программа печатает `Done` и завершает выполнение. В случае ввода неверной команды программа печатает `Invalid command, ignoring` и ожидает ввода следующей команды. Все команды являются нечувствительными к регистру символов (т.е. могут вводиться в верхнем или нижнем регистре или любым сочетанием строчных и заглавных букв).

Должно поддерживаться выполнение следующих команд:

1. **loc** позволяет пользователю ввести почтовый индекс, а затем находит соответствующие индексу город, штат, графство и географические координаты, которые соответствуют введённому почтовому индексу, после чего выводят на экран найденные данные. Если почтовый индекс введён неверно или не найден в наборе данных, вместо вывода на экран найденных данных программа должна вывести сообщение об ошибке. Ниже приведён пример работы данной команды:

```
Command ('loc', 'zip', 'dist', 'end') => loc
loc
Enter a ZIP Code to lookup => 12180
12180
ZIP Code 12180 is in Troy, NY, Rensselaer county,
coordinates: (042°40'25.32"N,073°36'31.65"W)
```

2. **zip** позволяет пользователю ввести название города и штата, а затем находит соответствующий им почтовый индекс (или несколько почтовых индексов) и выводит его (или их). Если название города и/или штата введёно неверно или они не найдены в наборе данных, вместо вывода на экран найденных данных программа должна вывести сообщение об ошибке. Ниже приведён пример работы данной команды:

```
Command ('loc', 'zip', 'dist', 'end') => zip
zip
Enter a city name to lookup => troY
troY
Enter the state name to lookup => ny
ny
The following ZIP Code(s) found for Troy, NY: 12179, 12180, 12181, 12182, 12183
```

3. **dist** позволяет пользователю ввести два почтовых индекса, а затем находит геодезическое расстояние между координатами местоположения, привязанными к соответствующим индексам. Если любой из двух почтовых индексов введён неверно или не найден в наборе данных, вместо вывода на экран найденных данных программа должна вывести сообщение об ошибке. Ниже приведён пример работы данной команды:

```
Command ('loc', 'zip', 'dist', 'end') => dist
dist
Enter the first ZIP Code => 19465
19465
```

```
Enter the second ZIP Code => 12180
12180
The distance between 19465 and 12180 is 201.88 miles
```

4. **end** завершает цикл обработки команд, введённых пользователем, и приводит к завершению программы.

Вспомогательный модуль `zip_util` содержит функцию `read_zip_all()`, которая возвращает список, каждый элемент которого, в свою очередь, является списком, содержащим данные об одной почтовом индексе. Попробуйте выполнить следующие команды:

```
import zip_util
zip_codes = zip_util.read_zip_all()
print(zip_codes[0])
print(zip_codes[4108])
```

в результате чего должно быть выведено:

```
['00501', 40.922326, -72.637078, 'Holtsville', 'NY', 'Suffolk']
['12180', 42.673701, -73.608792, 'Troy', 'NY', 'Rensselaer']
```

Обратите внимание, что данные о каждом почтовом индексе содержат следующие поля: почтовый индекс (строка), широта (значение дано в градусах, тип данных float), долгота (значение дано в градусах, тип данных float), город (строка), штат (строка) и графство (строка).

### Задание 7: Ввод/вывод с файлами CSV, обработка больших объёмов данных

Напишите приложение для работы с данными фермерских рынков.

Требования к приложению:

1. Первоначально приложение должно быть реализовано с использованием элементов функционального программирования. После изучения объектно-ориентированного программирования создайте версию приложения с использованием ООП.

2. Первоначально приложение должно иметь текстовый (консольный) пользовательский интерфейс, организованный по принципу REPL, либо работать в неинтерактивном режиме через командую строку (интерфейс типа CLI). После изучения библиотек GUI (Graphic User Interface, графический пользовательский интерфейс), создайте версию приложения с оконным графическим пользовательским интерфейсом.

3. Первоначально в качестве исходного источника данных выступает файл `Export.csv`. После изучения возможностей Python по взаимодействию с СУБД загрузите данные из файла `Export.csv` в СУБД по вашему выбору и создайте версию приложения, работающего с данными через СУБД.

4. Первоначально модульные тесты необходимо создать при помощи блока `if␣__name__␣==␣'__main__'`. После изучения дополнительных возможностей Python по поддержке модульного тестирования, создайте версию приложения, использующую одну из этих возможностей.

5. Разработка должна производиться с использованием всех необходимых возможностей системы контроля версия git. В частности, необходимо широко использовать ветви (branches).

6. Необходимо реализовать следующую функциональность:
    - Просматривать список всех фермерских рынков в стране (включая рецензии и рейтинги) с разбивкой по страницам;
    - Осуществлять поиск фермерского рынка по городу и штату, а также по почтовому индексу с возможностью ограничить зону поиска определенной дальностью (например, на удалении не более 30 миль);
    - Переходить от результатов поиска к просмотру подробных данных о любом рынке, присутствующем в поисковой выдаче;
    - Просматривать и оставлять рецензии на любой фермерский рынок, состоящие из необязательного текста рецензии и обязательного рейтинга (от 1 до 5 звезд);
    - Создавать рецензии, привязанные к имени и фамилии пользователя;
    - Распределять рынки по различным критериям (рейтингу, городу и штату, удаленности от определенной точки и т.п.) от минимального к максимальному значению или наоборот;
    - Удалять требуемые записи и выходить из программы.

Выполненное задание представляет собой приложение в подготовленном для развертывания виде и комплект документации. Обязательно должно быть учтено следующее:

1. Приложение должно быть подготовлено для развёртывания на клиентской машине, т.е. должны быть предоставлены достаточные скрипты, установщики и т.п., обеспечивающие простое развертывание, первоначальное конфигурирование и подготовку к первому запуску приложения;
2. Все операции по развёртыванию должны быть максимально автоматизированы;
3. Если какие-либо операции по развертыванию затруднительно или невозможно автоматизировать, то руководство для администратора должно содержать подробное описание процесса ручного выполнения данных операций;
4. Документация должна состоять из руководства пользователя и руководства для администратора;
5. Руководства должны содержать текстовое описание и графические материалы, быть аккуратно и красиво оформлены, при необходимости иметь содержание;
6. Руководство пользователя должно содержать описание всех функций приложения;
7. Руководство для администратора должно содержать полную информацию о развертывании приложения, а также поддержки его безопасной и стабильной эксплуатации на всём протяжении жизненного цикла программного обеспечения;

### Задание 8: ООП, АСД (абстрактные структуры данных)

Спроектируйте и реализуйте абстрактную структуру данных RatPoly, представляющую собой полинома с рациональными коэффициентами. Для представления рациональных чисел разработайте и реализуйте отдельную АСД RatNum. При необходимости, обращайтесь к справочным данным по рациональным числам и полиномам, например к `http://vmath.ru/vf5/polynomial`.

RatNum представляет собой неизменяемое (immutable) рациональное число. RatNum может использоваться для представления любого элемента множества рациональных чисел, а также специального элемента “NaN” (не-число), получающегося в результате деления на ноль.

Элемент “NaN” является во многом особенным. При выполнении любых арифметических операций с “NaN”, результат будет “NaN”. Что касается операций сравнения, таких как “меньше чем”, “NaN” считается равным самому себе и большим, чем любые другие рациональные числа.

Примеры объектов RatNum включают “-1/13”, “53/7”, “4”, “NaN” и “0”.

Необходимо обеспечить выполнение следующих операций:

- Для RatNum:
  - is_nan()
  - is_negative()
  - is_positive()
  - compare_to()
  - float_value()
  - int_value()
  - унарный - (аддитивная инверсия)
  - + (сложение)
  - бинарный - (вычитание)
  - * (умножение)
  - / (деление)
  - gcd()
  - __str__()
  - __hash__()
  - __eq__()
- Для RatPoly:
  - degree()
  - get_coeff()
  - is_nan()
  - scale_coeff()
  - унарный - (аддитивная инверсия)
  - + (сложение)
  - бинарный - (вычитание)
  - * (умножение)
  - / (деление)
  - eval()
  - differentiate()
  - anti_differentiate()
  - integrate()

  - value_of()
  - __str__()
  - __hash__()
  - __eq__()

В процессе проектирования и реализации требуется создать, соблюдая именно такой порядок, следующее:

- Спецификации всех методов (кроме вспомогательных методов, которые не предназначены для вызова клиентским кодом), содержащие для каждого метода:
  - Краткое описание
  - Определение поля или полей представления (representation fields)
  - Определение инварианта представления (representation invariant)
  - Определение функции абстракции (abstraction function)
  - @requires
  - @modifies
  - @effects
  - @throws
  - @returns
- Полный набор модульных тестов
- Реализацию всех классов и методов

### Задание 9: Итераторы и генераторы

При выполнении данного задания словом "перечисление" обозначается необходимость возврата элементов, а не требование использования функции `enumerate`. Например, фраза "итератор перечисляет последовательность $a0, a1, a2, \ldots$" означает, что эти элементы должны возвращаться при вызове метода `__next__`, а не то, что элементы должны быть кортежами $(i, a_i)$, как, например, кортежи, возвращаемые функцией `enumerate`.

1. Напишите класс итераторов `Fibo`, который перечисляет все числа Фибоначчи. Для данного задания последовательность Фибоначчи начинается со следующих значений: $0, 1, 1, 2, 3, \ldots$. Формула, позволяющая определить $n$-ное число Фибоначчи, задаётся рекуррентным соотношением $f_n = f_{n-1} + f_{n-2}$. Ваше решение не должно использовать никаких функций, кроме сложения. В классе `Fibo` должны быть предусмотрены, как минимум, метод инициализации, метод `__iter__`, позволяющий получить итератор, а также метод `__next__`.
2. Напишите генератор `integers`, который перечисляет все неотрицательные целые числа по возрастанию начиная с 0.
3. Напишите генератор `primes`, который перечисляет все простые числа. Простым числом называется такое целое число $p > 1$, единственными делителями которого являются $p$ и 1.