Вопросы/Задачи к экзамену
(Python язык программирования) Junior developer.

Экзамен состоит из билетов, билет содержит в себе 3 основных теоретических вопроса и 1 задачу, а также дополнительные вопросы.

Всего - **50** вопросов для подготовки. (± *32-33 различных билета*)

⚠️Прекрасные новости - за экзамен можно получить **60 баллов** если:

- ответить максимально идеально на основную теорию (+25 баллов)
    - 1 вопрос это 8,33 баллов;
- решить задачу (+20 баллов)
- ответить на доп вопросы (+15 баллов)

или 😩 0 баллов (*если пустой лист*);

БИЛЕТ ФОРМИРУЕТСЯ СЛУЧАЙНЫМ ОБРАЗОМ.
НА ПОДГОТОВКУ К ОТВЕТУ ВЫДЕЛЯЕТСЯ РОВНО **59 МИНУТ** ВРЕМЕНИ.
(*ЕСЛИ ГОТОВЫ РАНЬШЕ - ЭТО ВСЕЦЕЛО ПРИВЕТСТВУЕТСЯ*)

РАЗРЕШАЕТСЯ 1 РАЗ ПЕРЕБРАТЬ БИЛЕТ (БЕЗ ПОТЕРИ БАЛЛОВ) ЗА КАЖДЫЙ ПОСЛЕД-Й ПЕРЕБОР **-20 БАЛЛОВ**.

ПРИ ПОДГОТОВКЕ МОЖНО ИСПОЛЬЗОВАТЬ КОНСПЕКТЫ ЛЕКЦИЙ!!! (НО ЭТО ВСЕ РАВНО НЕ ПОМОЖЕТ)

СПИСЫВАНИЕ КАРАЕТСЯ АВТОМАТИЧЕСКОЙ НЕ-СДАЧЕЙ ЭКЗАМЕНА!!!

ЗАПРЕЩЕНО ИСП-Е ИНТЕРНЕТА И ДР СРЕДСТВ. НО РАЗРЕШЕНО ПРИ РЕШЕНИИ ЗАДАЧИ ИСП-ТЬ VSCODE.

ЭКЗАМЕН 😎***УСПЕШНО СДАН*** *- ЕСЛИ ВЫ НАБРАЛИ* ***>= 30,5 БАЛЛОВ****.*

1 БЛОК (основы языка)

1. Компиляция программы. Ввод/Вывод данных.
2. Переменные. Типы данных. Операции над переменными.
3. Условная конструкция IF-ELSE. Зачем нужен ELIF?
4. Виды циклов в Python. While. For.
5. Работа с файлами и строками. F-строки.
6. PEP. Виды PEP.
7. Коллекции в Python. Списки. Срезы.
8. Коллекции в Python. Кортежи. Множества. Словари.
9. Оценка сложности программы.
10. Алгоритмы сортировок. Сортировка пузырьком, и выбором.
11. Функции. Вызов функций, способы задания параметров (по-умолчанию, *argv, **kwargs… и др)
12.Функции. Декораторы. Понятие перегрузки функции.
13. Полезные функции min/max/…
14. Комментарии в Python, Docstring, + что такое и зачем используется Typehints? Примеры.
15. Глобальные переменные.
16. Исключения и ошибки. Обработка исключений. RAISE.

2 БЛОК (ООП, модули)

1. ООП. Преимущества / недостатки ООП. SOLID принципы. Также основные понятия класса/объекта/атрибута/методы.
2. ООП. Статические и динамические атрибуты (поля). Понятие конструктора/деструктора класса, и что такое SELF?
3. ООП. Основные концепции ООП.
4. ООП. Public/Private атрибуты (поля). Public/Private методы. Что такое single underscore?
5. ООП. Соглашение NM. Основные типы методов класса. Магические методы класса.
6. ООП. Понятие стека/очереди/дека.
7. Модули и библиотеки. В чем их отличия?
8. Стандартные модули и библиотеки. OS. JSON. RE. Характеристики и параметры функций, примеры.
10. Стандартные модули и библиотеки. MATH. RANDOM. DATETIME. Характеристики и параметры функций, примеры.
11. Внешние модули и библиотеки. REQUESTS. NumPy. PIP.
12. Внешние модули и библиотеки. Pillow. Matplotlib. PIP.
13. Внешние модули и библиотеки. PyAutoGui. Selenium. Автоматизация процессов.
14. Парсинг данных.

3 БЛОК (АИСД и сторонние модули)

1. АИСД. Связный список. Виды связного списка.
2. АИСД. ХЕШ-таблица. Коллизии.
3. АИСД. Динамический массив данных.
4. АИСД. Деревья. Основные виды деревьев.
5. АИСД. Графы, алгоритмы.
6. Telebot. Как создать простого бота для вывода фразы Hello world? Какой алгоритм создания ботов?
7. Табличные форматы файлов. Формат CSV. Описание, методы работы, характеристики и применение на практике.
8. Табличные форматы файлов. Формат CSV. Описание. Диалекты. Регистрация диалектов и их применение на практике.
9. Табличные форматы файлов. Формат Excel. Описание, методы работы, характеристики и применение на практике.
10. ИИ. Описание, принципы работы, данные. Построение ИИ.
11. ПО, GUI, Desktop APP, PyQT. Версии PyQt. Основные компоненты PyQt. Сравнение Tkinter с PyQt (что лучше?).
12. FLASK/Django.
13. Docker. Определение, принцип работы, виды Docker. Сравнение Docker CE/EE. Docker-Checker.
14. Docker. DockerFile - определение, расширение и формат. Основные инструкции/функции. Docker-Compose, swarm, kubernetes.
15. Docker. Основные определения, Сети/Тома в Docker. Практическое применение Docker.
16. Интернет. HTTP. DNS. UDP. TCP/IP.. UDP.
17. ОС. Работа с терминалом. Основные команды терминала. POSIX. Концепции построения сетей.
18. ОС. Процессы и потоки. Понятие сигнала. Виды сигналов и их применение.
19. ОС. Процессы, межпроцессорное взаимодействие. Multiprocessing.

20*. <самый крутой вопрос> Зачем нужен Python? Где применяется Python?

Примеры задач для самоподготовки:

1. Дан список (который задает пользователь - вводит с использованием спец. функции заполнения), найти минимальный положительный элемент в этом списке.

2. Дан словарь (который задает пользователь), создайте функцию get_index() и реализуйте поиск индекса первого вхождения положительного четного ключа, а после вывести значение по этому ключу (key).

3. Даны три кортежа, соедините три кортежа воедино, написав соответствующую функцию объединения. А также выведите длину получившегося кортежа.

4. <задача на стек данных> Напишите функцию, которая принимает на вход строку, содержащую только символы '(', ')', '{', '}', '[' и ']', и определяет, является ли расстановка скобок в этой строке корректной.

5. <задача на стек данных> Напишите функцию, которая проверяет балансировку символов в строке. В строке могут присутствовать только символы '(' и ')'. Функция должна вернуть True, если каждой открывающей скобке соответствует закрывающая, и False в противном случае.

6. Напишите функцию которая принимает список вещественных чисел и возвращает два самых наибольших отрицательных числа в этом списке данных.

7. <задачка с leetcode> Учитывая список целых чисел **nums** и целое число **target**, верните индексы двух чисел так, чтобы их сумма составляла **target** . Вы можете предположить, что каждый вход будет иметь ровно одно решение , и вы не можете использовать один и тот же элемент дважды. Вы можете вернуть ответ в любом порядке.

Пример 1:

> **Ввод:** nums = [2,7,11,15], target = 9
> **Выход:** [0,1]
> **Объяснение:** поскольку nums[0] + nums[1] == 9, мы возвращаем [0, 1].

Пример 2:

> **Ввод:** nums = [3,2,4], цель = 6
> **Вывод:** [1,2]

Пример 3:

> **Ввод:** nums = [3,3], цель = 6
> **Вывод:** [0,1]

** стоит отметить что задача может быть на любую из пройденных тем, советую повторить ООП, АИСД (деревья, хеш-таблицы). Также работу с файлами .txt, .csv, и др*

простой пример задачки на ооп.

Задача: Библиотека и Книги

Описание задачи: Реализуйте простую систему для управления библиотекой. У каждого экземпляра книги есть название и автор. У библиотеки есть список книг, и она может добавлять книги, удалять их и выводить информацию обо всех имеющихся книгах.

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
    ...


class Library:
    def __init__(self):
        self.books = []
    ...
```

Задачки на файлы:

- открыть файл .csv - прочитать его содержимое вывести на экран.
- открыть файл .csv - внести в него какие то изменения, сохранить изменения.
- открыть файл .txt - прочитать и вывести его содержимое.
- открыть файл .txt - записать изменения в файл.
- попробуйте сделать операцию объединения двух текстовых файлов в одине единый файл.

Типовая задача:
создать функцию foo() с чтением из потока данных argv целых чисел до тех пор пока не встретиться отрицательное число.

Таблица-1 Оценивание ученика:

Имя Фамилия ученика	Задача (20б)	Т1 (8,3б)	Т2 (8,3б)	Т3 (8,3б)	Доп1 (5б)	Доп2 (5б)	Доп3 (5б)	Итого:

пометка **Т1, **Т2**, **Т3** - основные теоретические вопросы по билету;*

Таблица-2 Оценивание задачи:

Номер билета	Работоспособность кода (согласно условию и требованиям) (9б)	Соблюдение всех требований условия задачи (6б)	Соблюдение PEP (2б)	Docstring + Typehints (2б)	Читаемость кода, оптимизированность, оценка сложности программного кода (1б)