Введение

# **1. Присвоение по ссылке и по значению. Специфика создания объектов и присвоения в Python, особенности Python в связи с распространенностью использования неизменяемых типов.**

# **3. Циклы в Python, работа и устройство цикла for, типичное применение range и enumerate в цикле for.**

# **4. Списки в Python. Обращение к элементам списка и создание срезов. . Обход списка и поиск элементов в списке. Ключевые операции, приводящие к изменению списка и порождающие измененные списки, копирование списков.**

Словари, множества, кортежи

# **5. Словари в Python. Итерирование по словарям, преобразование между словарями и списками в Python. Операции с представлениями словарей.**

# **7. Множества в Python. Основные способы создания, получения и изменения значений. Обход множеств. Выполнение основных операций с парой множеств в Python.**

# **8. Кортежи в Python. Отличия кортежей от списков. Распаковка и частичная распаковка кортежей.**

# **11. Объявление и вызов функции в Python. Параметры функции со значением по умолчанию и комментирование функции. Получение информации о функции. Способы передачи параметров при вызове функции.**

# **12. Передача переменного количества параметров (именованных и не именованных) в функции Python. Вызов функции с позиционными параметрами, находящимися в списке, и именованными параметрами, находящимися в словаре.**

Исключения и файлы

# **+13. Синтаксис и семантика обработки исключительных ситуаций в Python. Создание пользовательских исключений и инструкция assert.**

# **14. Базовые операции для работы с файлами в Python. Использование инструкции with … as на примере работы с файлами.**

Модули и пакеты

Объектно-ориентированное программирование 1

# **16. Концепция класса и объекта. Принципы и механизмы ООП.**

# **17. Объявление класса, конструктор, создание объектов и одиночное наследование в Python. Управление доступом к атрибутам класса в Python.**

# **18. Полиморфизм и утиная типизация и проверка принадлежности объекта к классу в языке Python.**

# **19. Методы классов и статические переменные и методы в Python. Специальные методы для использования пользовательских классов со стандартными операторами и функциями.**

**20. Основные возможности, поддерживаемые функциональными языками программирования. Поддержка функционального программирования в Python.**

**23. Вложенные функции и замыкания, специфика реализации в Python.**

**24. Функции высшего порядка и декораторы в Python.**

# **26. Итераторы в Python: встроенные итераторы, создание собственных итераторов, типичные способы обхода итераторов и принцип их работы. Встроенные функции для работы с итераторами и возможности модуля itertools.**

# **27. Функции генераторы и выражения генераторы: создание и применение в Python.**

# **28. Специфика массивов, как структур данных. Динамические массивы – специфика работы, сложность операций. Специфика работа с array в Python.**

# **29. Абстрактная структура данных стек и очередь: базовые и расширенные операции, их сложность.**

# **30. Специфика реализации и скорости основных операций в очереди на базе массива и связанного списка.**

# **31. Связанные списки: однонаправленные и двунаправленные – принцип реализации. Сравнение скорости выполнения основных операций в связанных списках и в динамическом массиве.**

# **34. Алгоритм сортировки вставками, его сложность. Алгоритм быстрого поиска в отсортированном массиве. Сложность поиска в отсортированном и не отсортированном массиве.**

# **36. Алгоритм быстрой сортировки, сложность сортировки и возможности по ее улучшению.**

# **38. Реализация двоичных деревьев в виде связанных объектов. Различные реализации рекурсивного обхода двоичных деревьев.**

**Типы задач:**

· Обработка строк: преобразование строки в список, преобразования регистра символов, подсчет количества слов и т.д.

· Генераторы списков.

· Сортировка встроенная с использованием key. Сортировка отдельных элементов, удовлетворяющих условию.

· Рекурсивные функции.

· Работа с двумерными матрицами, в том числе сортировка строк, перестановка строк и столбцов.

· Создание классов. Уметь реализовать какой-нибудь магический метод, например, \_\_str\_\_. Метод вы выбираете сами.

· Реализовать декоратор.

· Уметь создать заданное бинарное дерево, просто связав вершины (без использования классов). Можно в виде вложенных списков (как в лекции). Уметь выполнять обход такого дерева (для проверки условия, вычисления какого-либо значения или вывода на экран). Например, найти количество четных элементов.

· Уметь создать связный список, просто связав вершины. Класс использовать только для вершины. Например, создать связный список из обычного списка чисел. В этом списке уметь выполнять основные операции (поиск, удаление, добавление элементов). В задании нужно будет реализовать только одну операцию.