Методические рекомендации для выполнения лабораторных работ

Тема 1. Структуры данных. Работа в VS Code со структурами данных «массив/список»

Алгоритм создания своего репозитория на GitHub через веб-интерфейс:

Зарегистрироваться на GitHub в зоне .com или войти в учётную запись. Можно выбрать публичный аккаунт (репозитории видят все пользователи) или приватный (репозитории видят только одобренные участники).
В правом верхнем углу экрана нажать на кнопку «+» и выбрать пункт «New repository».
Заполнить поля:
«Repository name» — задать название для репозитория.
«Description» (необязательно) — добавить краткое описание проекта.
«Public/Private» — выбрать видимость репозитория (публичный или приватный).
Отметить галочку «Initialize this repository with a README», если нужно создать файл README.md в корне репозитория.
Нажать кнопку «Create repository».
Важно: можно добавлять файлы (README.md, .gitignore) только в новый репозиторий, который ещё не создан нигде, кроме GitHub, иначе возникнет конфликт.

Чтобы создать папку в репозитории на GitHub, нужно:
Перейти в нужный репозиторий.
Нажать на кнопку «Добавить файл», затем на ссылку «Создать новый файл» в выпадающем меню.
Ввести имя папки, за которым следует символ косой черты (например, empty_folder/).
Ввести имя файла.
Нажать на кнопку «Зафиксировать изменения».
Написать сообщение коммита, выбрать ветку, в которую коммитить файлы, и нажать на кнопку «Зафиксировать изменения» для сохранения изменений.

Чтобы загрузить файлы с компьютера в свой репозиторий на GitHub через веб-версию, нужно:
Перейти на главную страницу репозитория на GitHub.
Над списком файлов выбрать выпадающее меню «Добавить файл» и нажать «Загрузить файлы». Альтернативно можно перетащить файлы в браузер.
Выбрать файлы для загрузки, перетащив их или нажав «Выбрать файлы».
В поле «Сообщение о коммите» ввести краткое, но информативное сообщение о внесенных изменениях.
Ниже полей с сообщением о коммите решить, добавлять коммит в текущую ветку или в новую.
Нажать «Предложить изменения».

1.1. Типовая задача со структурой данных «массив/список» на ЯП Python.

Дана строка 'AaBbCcDd'. Используя срезы с шагом получите две строки: только с заглавными и только со строчными буквами. Выведите их на экран.

```
# Сохраним строку в переменной.
str_1 = 'AaBbCcDd'
# Выводим на экран требуемые строки.
print('str_1[::2] ->', str_1[::2], end='\n\n')
# Смещаем вывод на единицу.
print('str_1[1::2]->', str_1[1::2])
```

Дан список ['a', '1', 'b', '2', 'c', '3']. Разбейте его на два списка: только с буквами и только с числами. Сам список затем удалите, а новые списки выведите на экран, каждый на отдельной строке.

```
# Сохраним список в переменной.
li = ['a', '1', 'b', '2', 'c', '3']
# Получаем требуемые срезы.
li_1 = li[0::2]
li_2 = li[1::2]
# Сам список удаляем.
del li
# Выводим полученные списки на экран.
print(li_1, li_2, sep='\n\n')
```

В Python для организации стека (структуры данных, работающей по принципу «последним пришёл, первым вышел» (LIFO)) можно использовать разные подходы: списки, класс deque из модуля collections или связанный список.

С использованием списков:

```
stack =
stack.append('a')
stack.append('b')
stack.append('c')
print(stack)
```

Объяснение: создан пустой список stack, с помощью метода append() добавлены элементы a, b и c. Функция print() отображает содержимое стека — [‘a’, ‘b’, ‘c’].

С использованием deque:

```
from collections import deque
stack = deque()
stack.append('a')
stack.append('b')
stack.append('c')
print(f'Stack: {list(stack)}')
```

Объяснение: создан объект класса deque stack, с помощью метода append() добавлены элементы, функция print() отображает содержимое стека.

С использованием связного списка:

```
class Stack:
    def __init__(self):
        self.top = None
    def push(self, data):
        new_node = Node(data)
        new_node.next = self.top
        self.top = new_node
    def pop(self):
        if not self.is_empty():
            popped = self.top
            self.top = self.top.next
            popped.next = None
```

Объяснение: стек — связный список, где каждый узел содержит данные и указатель на предыдущий узел. При добавлении новый элемент становится вершиной стека, а при удалении на вершине оказывается предыдущий элемент.

Типовая задача со структурой данных «массив/список» на ЯП C++.

Подготовить самостоятельно две задачи с решениями.

1.2. Типовая задача со структурой данных «массив/список» на ЯП Java.

Подготовить самостоятельно две задачи с решениями.