**Практическая работа № 1**

# Основы работы с Git, Binder и Python.

Оглавление

[Основы работы с Git, Binder и Python. 1](#_Toc50038761)

[Цель работы 1](#_Toc50038762)

[Задачи работы 2](#_Toc50038763)

[Перечень обеспечивающих средств 2](#_Toc50038764)

[Общие теоретические сведения 3](#_Toc50038765)

[**Определения** 3](#_Toc50038766)

[**Принцип работы Git** 3](#_Toc50038767)

[**Создание нового репозитория в GitHub** 4](#_Toc50038768)

[**Создание нового файла в GitHub** 5](#_Toc50038769)

[**Редактирование файла в GitHub** 6](#_Toc50038770)

[**Коммит в GitHub** 6](#_Toc50038771)

[**Создание пул-реквеста в GitHub** 7](#_Toc50038772)

[**Принятие и отклонение пул-реквеста в GitHub** 7](#_Toc50038773)

[**Важное замечание** 8](#_Toc50038774)

[**Основные операции Jupyter notebook** 9](#_Toc50038775)

[**Переменные и вывод результатов Python** 10](#_Toc50038776)

[**Типы данных и основные операции с ними в Python** 10](#_Toc50038777)

[**Условия Python** 12](#_Toc50038778)

[**Цикл for в Python** 14](#_Toc50038779)

[**Функции Python** 15](#_Toc50038780)

[Задание 17](#_Toc50038781)

[Требования к отчету 17](#_Toc50038782)

[Литература 18](#_Toc50038783)

## Цель работы

* Обучение работе с системой управления версиями Git с использованием сервиса GitHub.
* Обучение работе с языком программирования Python с использованием интерактивного инструмента Jupyter Notebook.
* Обучение работе с сервисом Binder (mybinder.org).
* Создание и настройка базовых инструментов для использования в последующих практических заданиях.

## Задачи работы

1. Изучить устройство и базовые команды Git.
2. Научиться создавать новый репозиторий и копировать уже существующий.
3. Научиться сохранять изменения в репозитории.
4. Изучить основные типы данных и операции Python.
5. Научиться работать с Jupyter notebook.

## Перечень обеспечивающих средств

1. ПК.
2. Учебно-методическая литература.
3. Задания для самостоятельного выполнения.

## Общие теоретические сведения

### **Определения**

*Git* – современная и самая распространенная система управления версиями.

Основное предназначение систем управления версиями:

1. Контроль версий проектов, состоящих из файлов и директорий, с возможностью переключения между любыми версиями.
2. Упрощение взаимодействия между участниками в случае совместной разработки.

*Репозиторий* – специальным образом подготовленное хранилище файлов.

*GitHub* – онлайн-сервис для хранения репозиториев проектов с большим количеством инструментов для выстраивания эффективной совместной разработки.

*Python* – интерпретируемый высокоуровненвый язык программирования общего назначения. Обладает простым для изучения и использования синтаксисом. Активно развивается, имеется большое количество библиотек для работы с различными предметными областями.

*Jupyter notebook* – инструмент для итерационного выполнения кода на Python. Запускается в веб-браузере.

*Binder* ([mybinder.org](http://www.mybinder.org/)) – интернет-сервис для работы с копиями Jupyter notebook в режиме онлайн. Использует в качестве источника данных проект GitHub.

### **Принцип работы Git**

4-й коммит

HEAD

3-й коммит

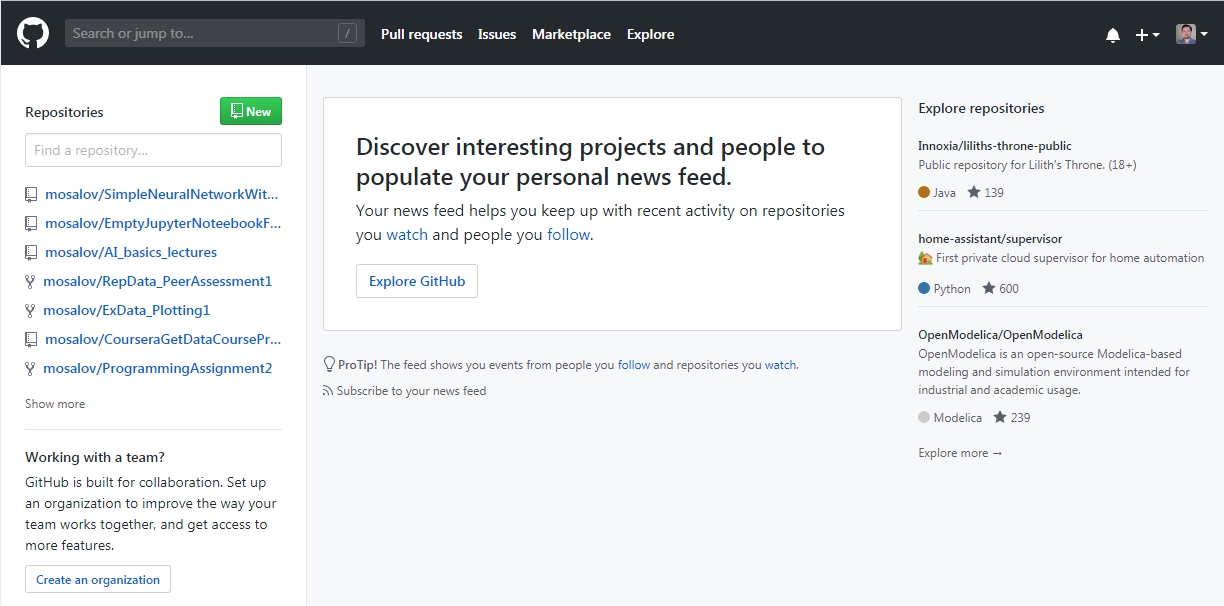
Рабочая копия

2-й коммит

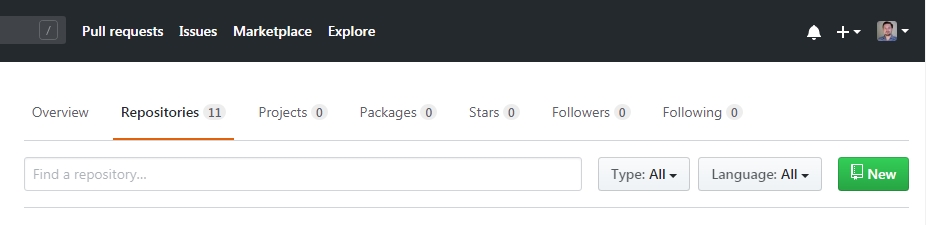
1-й коммит

### **Создание нового репозитория в GitHub**

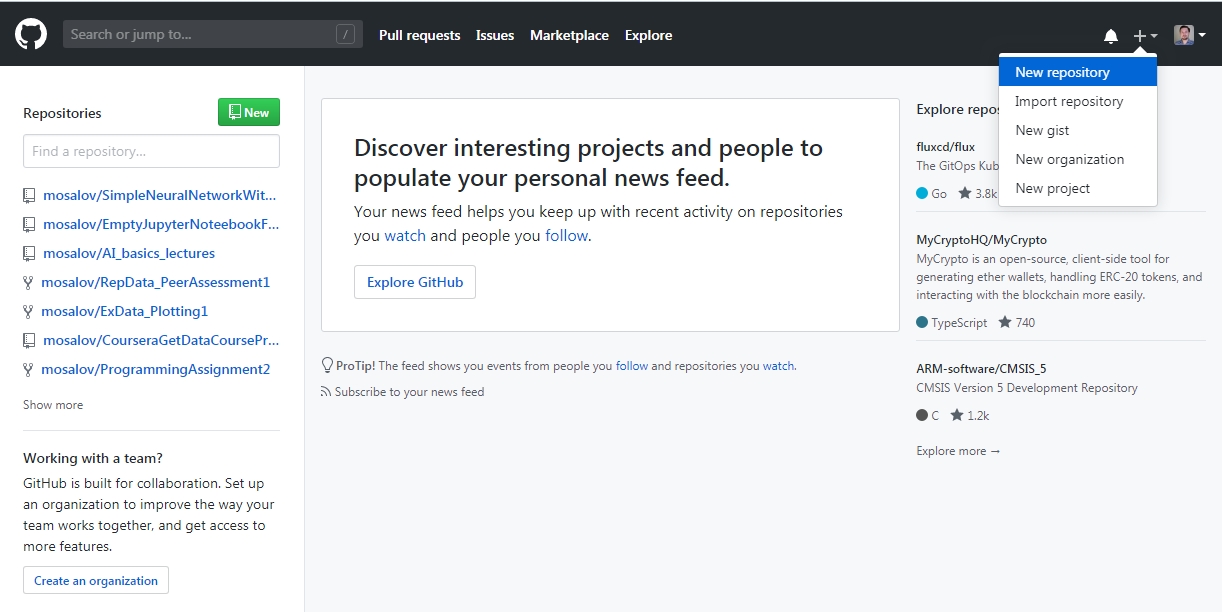
Вариант №1 – С помощью кнопки «New» в левом верхнем углу страницы <https://github.com/>



Вариант №2 – С помощью кнопки «New» в правом верхнем углу страницы https://github.com/*<username>*?tab=repositories

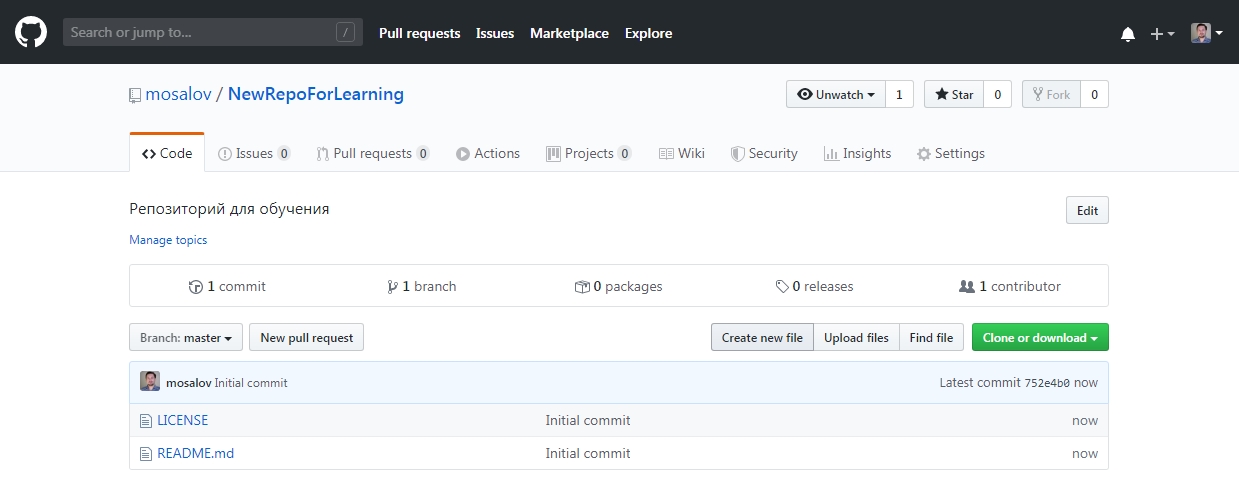


Вариант №3 – С помощью пункта «New repository» выпадающего меню, открывающегося по кнопке «+» в правом верхнем углу на любой из страниц



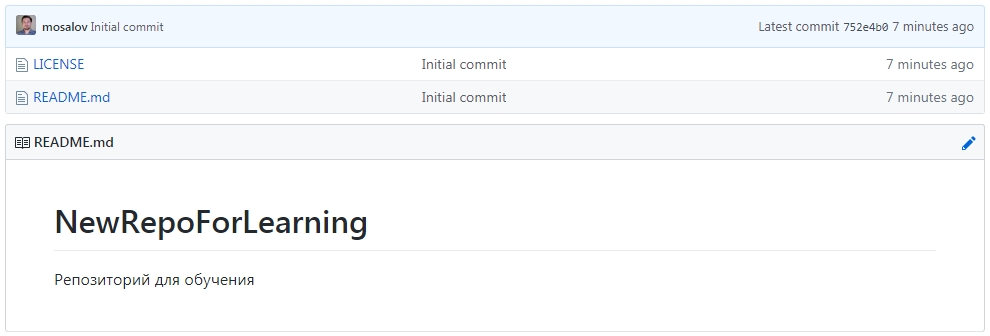
### **Создание нового файла в GitHub**

С помощью кнопки «Create new file» – первая в правом наборе кнопок над содержимым репозитория на странице репозитория



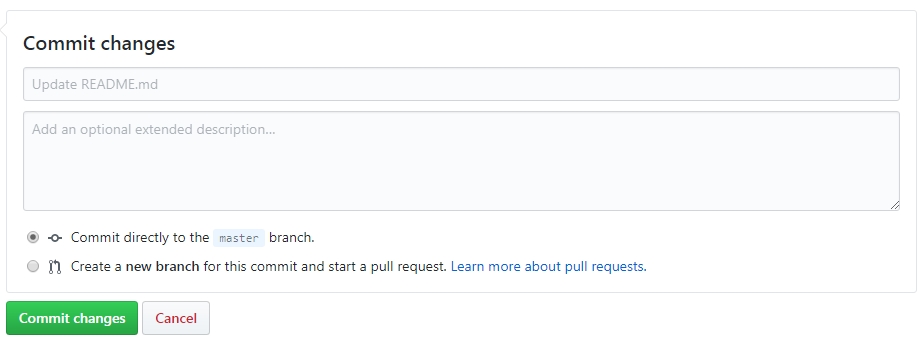
### **Редактирование файла в GitHub**

Нажать на ссылку с названием файла на странице репозитория, затем нажать кнопку с изображением карандаша в правом верхнем углу над содержимым файла



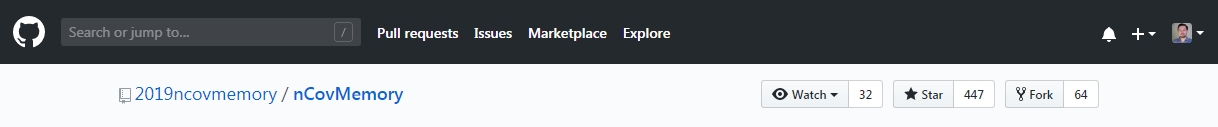
### **Коммит в GitHub**

После внесения изменений в структуру репозитория или содержимое файлов коммит делается с помощью кнопки «Commit changes» в левом нижнем углу под содержимым файла



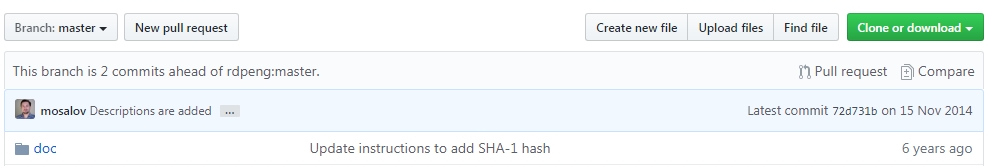
**Создание форка чужого репозитория в GitHub**

С помощью кнопки «Fork» в левом верхнем углу страницы репозитория

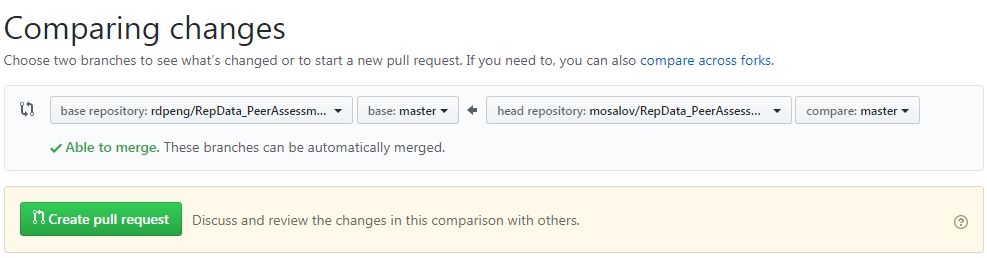


### **Создание пул-реквеста в GitHub**

Сначала с помощью кнопки «New pull request» открывается страница с данными пул-реквеста – вторая кнопка в левом наборе кнопок над содержимым репозитория на странице репозитория



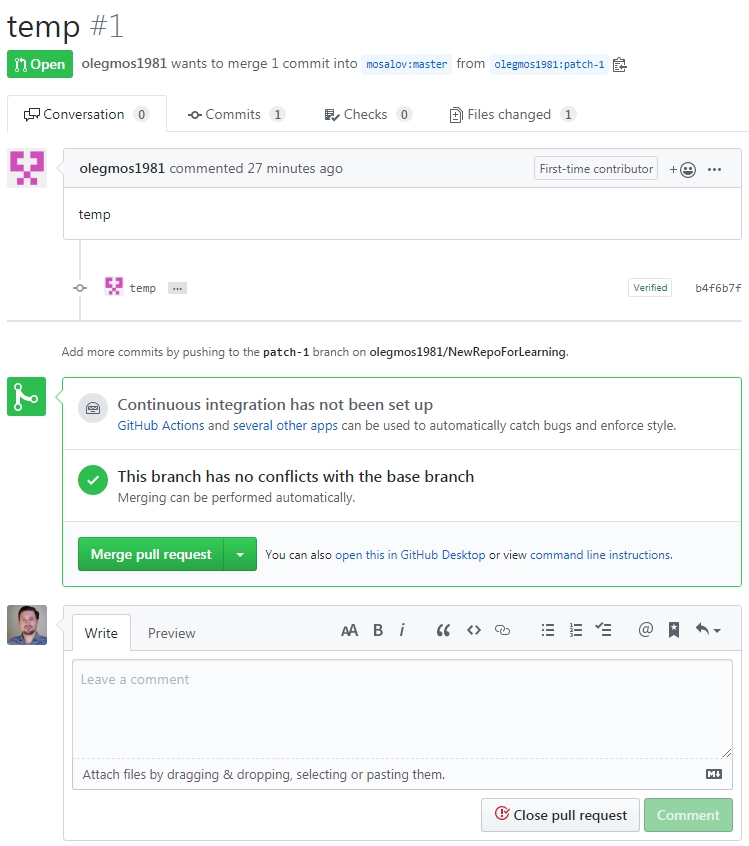
Затем с помощью кнопки «Create pull request» создается пул-реквест



### **Принятие и отклонение пул-реквеста в GitHub**

Принятие пул-реквеста осуществляется с помощью кнопки «Merge pull request» под описанием пул-реквеста.

Отклонение пул-реквеста осуществляется с помощью кнопки «Close pull request» в нижней части страницы с описанием пул-реквеста.

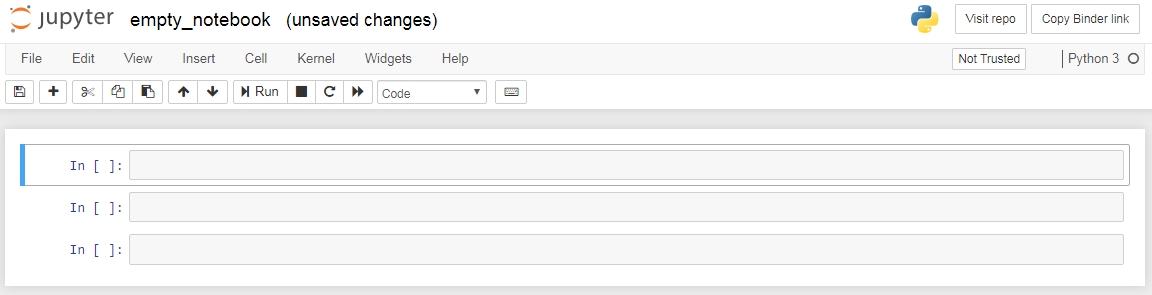


### **Важное замечание**

Все действия, допускающие комментарии, должны быть прокомментированы.

### **Основные операции Jupyter notebook**

Код может быть размешен и выполнен в нескольких ячейках:



Основные горячие клавиши:

|  |  |
| --- | --- |
| **Сочетание клавиш** | **Действие** |
| Enter | Открыть выделенную ячейку на редактирование (синяя полоска слева становится зеленой). |
| Esc | Выйти из режима редактирования выделенной ячейки (зеленая полоска слева становится синей). |
| Курсор вверх | Выделить ячейку сверху. |
| Курсор вниз | Выделить ячейку снизу. |
| a | Создать ячейку над выделенной. |
| b | Создать ячейку под выделенной. |
| Двойное нажатие d | Удалить выделенную ячейку. |
| z | Отменить последнее удаление ячейки. |
| x | Вырезать выделенную ячейку. |
| c | Скопировать выделенную ячейку. |
| v | Вставить вырезанную или скопированную ячейку. |
| Shift + Enter | Выполнить код в выделенной ячейке и выделить следующую ячейку. |
| Ctrl + Enter | Выполнить код в выделенной ячейке. |

### **Переменные и вывод результатов Python**

Для определения переменной достаточно выполнить код с указанием ее имени, например:

|  |
| --- |
| x  my\_first\_variable |

Значение переменной присваивается с помощью оператора «=», например:

|  |
| --- |
| x = 0  my\_first\_variable = 'Привет!' |

Можно сразу же определить переменную и присвоить ей значение, например:

|  |
| --- |
| x = 0 |

Для вывода значения переменной используется функция «print()», например:

|  |
| --- |
| print(x) |

### **Типы данных и основные операции с ними в Python**

В Python тип переменной определяется по ее значению. Ниже рассмотрены базовые типы данных.

*Целые числа* (int) задаются цифрами и, при необходимости, знаком минуса. Примеры целочисленных переменных:

|  |
| --- |
| x = 5  y = 100  z = -12 |

*Дробные числа* (float) задаются цифрами, десятичной точкой и, при необходимости, знаком минуса. Примеры дробных переменных:

|  |
| --- |
| x = 1.5  y = -0.125  z = 3.415926 |

Арифметические операции, т. е. операции с числами, с примерами:

|  |  |  |  |
| --- | --- | --- | --- |
| **Операция** | **Оператор** | **Пример записи** | **Результат** |
| сложение | + | 1 + 1 | 2 |
| вычитание | - | 8 - 0.5 | 7.5 |
| умножение | \* | 2 \* 5 | 10 |
| деление | / | 5.5 / 0.5 | 11 |
| возведение в степень | \*\* | 2 \*\* 5 | 32 |
| взятие по модулю (остаток от деления) | % | 7 % 3 | 1 |
| целочисленное деление | // | 7 // 3 | 2 |

Порядок выполнения перечисленных арифметических операций:

* Возведение в степень.
* Умножение, деление, взятие по модулю и целочисленное деление.
* Сложение и вычитание.

Для управления порядком выполнения операции используются круглые скобки.

*Строки* (str) задаются последовательностью символов заключенных в одинарные или двойные кавычки. Примеры строковых переменных:

|  |
| --- |
| x = "Привет!"  y = 'Кто здесь?'  z = "I like Python." |

Операции со строками, с примерами:

|  |  |  |  |
| --- | --- | --- | --- |
| **Операция** | **Оператор** | **Пример записи для x = "Привет!"**  **y = 'Кто здесь?'** | **Результат** |
| конкатенация (сложение, слияние) | + | x + y | "Привет!Кто здесь?" |
| дублирование | \* | x \* 2 | "Привет!Привет!" |
| получение символа по его индексу:  1) если индекс ≥ 0, то отсчёт идёт с начала строки, начиная с 0;  2) если индекс < 0, то отсчёт идёт с конца строки, начиная с -1. | [] | x[1]  x[-3] | "р"  "е" |
| определение длины строки | len() | len("Тест") | 4 |

*Списки* (list) – заключенная в квадратные скобки последовательность элементов любых типов (в том числе других списков), разделенных запятыми. Примеры списков:

|  |
| --- |
| [1, 'текст', 2, -0.19]  [[0, 1], 2, 'три'] |

### **Условия Python**

Структура условного оператора (оператора ветвления):

|  |
| --- |
| **if *<Условие 1>* :**  ***<Инструкция 1>***  **elif *<Условие 2>* :**  ***<Инструкция 2>***  **elif *<Условие 3>* :**  ***<Инструкция 3>***  **else:**  ***<Инструкция 4>*** |

**Elif** можно повторять столько раз, сколько требуется, либо не использовать вообще. При этом указывается очередное условие для проверки.

**Else** либо употребляется один раз, либо не используется. Инструкции в блоке else выполняются, если ни одно из вышеперечисленных условий не выполнено.

Возможные значения условий: истина — true и ложь — false.

Операции сравнения:

|  |  |  |  |
| --- | --- | --- | --- |
| **Операция** | **Оператор** | **Пример записи для x = 1 и y = 2** | **Результат** |
| равно | == | x == y | false |
| не равно | != | x != y | true |
| больше | > | x > y | false |
| меньше | < | x < y | true |
| больше или равно | >= | x >= y | false |
| меньше или равно | <= | x <= y | true |

Логические операции:

|  |  |  |  |
| --- | --- | --- | --- |
| **Операция** | **Оператор** | **Пример записи для x = true и y = false** | **Результат** |
| и | and | x and y | false |
| или | or | x or y | true |
| отрицание | not | not x | false |

Пример:

|  |
| --- |
| x = 5  **if** x < 5 **:**  print('Меньше пяти')  **elif** x > 5 **:**  print('Больше пяти')  **else :**  print('Равно пяти') |

### **Цикл for в Python**

Цикл **for** может выполняться по списку или по диапазону чисел.

Структура цикла **for** по списку:

|  |
| --- |
| **for *<Элемент>* in *<Список>* :**  ***<Инструкция>*** |

Пример цикла for по списку:

|  |
| --- |
| x = [1, 2, 3, 4]  **for** e **in** x **:**  print(e) |

Структура цикла **for** по диапазону чисел:

|  |
| --- |
| **for *<Индекс>* in range(*<Начало>*, *<Остановка>*, *<Шаг>*) :**  ***<Инструкция>*** |

При этом функция range(x, y, z), где x, y и z — целые числа, создает диапазон целых чисел, такой что:

* первое число в диапазоне равно x,
* разность между последовательными числами равна z,
* последнее число
  + меньше y, если z > 0,
  + больше y, если z < 0.

При этом, если не указано значение z, то оно считается равным 1; если не указано значение x, то оно считается равным 0.

Например, range(0, 5, 1) сформирует диапазон 0, 1, 2, 3, 4.

range(2, 10, 2) — 2, 4, 6, 8.

range(5, 0, -1) — 5, 4, 3, 2, 1.

range(2, 6) – 2, 3, 4, 5.

range(4) – 0, 1, 2, 3.

Пример цикла for по диапазону чисел:

|  |
| --- |
| **for** i **in** range(4) **:**  print(i) |

### **Функции Python**

Структура задания функции:

|  |
| --- |
| **def *<Имя функции>*(*<Список параметров>*) :**  ***<Инструкции>*** |

Если функция должна возвращать какое-либо значение, используется выражение:

|  |
| --- |
| **return *<Возвращаемое значение>*** |

Например, задание функции, вычисляющей квадрат числа:

|  |
| --- |
| **def** squared**(**x**) :**  **return** x \*\* 2 |

И вызов этой функции:

|  |
| --- |
| x = 2  print(squared(x)) |

## Задание

**Часть 1**

* Если вы не зарегистрированы на GitHub (<https://github.com>) – зарегистрируйтесь и активируйте учётную запись (вам понадобится работающий e‑mail).
* Зайдите в свою учётную запись.
* Сделайте форк репозитория

<https://github.com/mosalov/Notebook_For_AI_Main>. Далее в задании работаем с этим репозиторием.

**Часть 2**

* Зайдите на сайт <https://mybinder.org/>
* Введите ссылку на свой репозиторий в поле «GitHub repository name or URL».
* Нажмите кнопку «launch». Дождитесь открытия репозитория.
* Откройте (кликните) файл «empty\_notebook.ipynb».
* Cоздайте две переменные, в одной сохраните своё имя, во второй – фамилию.
* Создайте третью переменную, используя две ранее созданные переменные, сохраните в ней свои имя и фамилию, разделённые пробелом.
* Выведите значение третьей переменной.
* Выведите длину значений третьей переменной.
* Загрузите файл на локальный компьютер: меню «File → Download as → Notebook (.ipynb)». Назовите файл «*Фамилия*\_Задание 1.ipynb».

**Часть 3**

* Загрузите файл, созданный на предыдущем шаге, в ваш репозиторий, созданный на первом шаге, по пути «Notebook\_For\_AI\_Main/2020 Осенний семестр/Практическое задание 1/».
* Сделайте пул-реквест в репозиторий <https://github.com/mosalov/Notebook_For_AI_Main>

## Требования к отчету

Требуется представить отчет в виде пул-реквеста (детали см. выше).

## Литература

1. <https://habr.com/ru/company/playrix/blog/345732/>
2. <https://proglib.io/p/git-cheatsheet/>
3. <https://pythonworld.ru/samouchitel-python>
4. <https://www.coursera.org/learn/diving-in-python>
5. <https://stepik.org/course/67/promo>