Вводное занятие. Основы Python

Рассмотрим следующие вопросы:

- Python и Jupyter Notebook. Почему и как.
- Простые типы данных
- Работа с Git

ПОЧЕМУ PYTHON

Для решения задач анализа данных можно было бы использовать любой язык программирования. Но многие, в том числе и мы, будем использовать Python. Его преимущества:

- Простой синтаксис (многие задачи решаются просто, буквально в 1 строку)
- Динамическая типизация (язык сам определяет тип значения)
- Большое количество готовых библиотек
- Большое сообщество
- Стоит по умолчанию на Unix-системах

Python – не змея, а название комик-группы Monty Python!

Нужно понимать, что Python критикуют. И даже считают, что «это не язык программирования». Аргументы следующие:

- Python написан на C (правда, но это не является проблемой)
- Аналитики не пишут production-код (чаще всего так, но это не их забота. В production пойдёт уж код, над которым поработают профессиональные разработчики)

Аналитикам есть что ответить:

- Есть высоконагруженные проекты на Python
- Не все люди знают ассемблер со школы

JUPYTER NOTEBOOK

Для занятия рекомендуется установить дистрибутив Anaconda, в составе которого находится Jupyter Notebook. Это не обязательно – при желании можно использовать что угодно вплоть до Блокнота. Однако:

- Anaconda - Python + предустановленные библиотеки, которых хватит для решения большинства задач. Но можно установить и с нуля
- В Jupyter Notebook есть удобная реализация визуализации и построчного выполнения кода
- Могут быть проблемы при параллельно стоящих версиях Python

Вообще, Jupyter Notebook – своеобразный мини-сервер, который позволяет писать код в браузере. Позволяет выполнять программу по частям. Инструмент №1 для аналитиков.

Запускать удобно через Anaconda Navigator, Anaconda Prompt или через консоль с помощью команды *Jupyter Notebook.* Работа производится в файлах *.ipynb.

РЕКОМЕНДАЦИИ ПО РАБОТЕ

Рекомендации:

- В одной ячейке оставляйте по 2-3 простых действия. Это позволит отслеживать правильность выполнения, но не перегружать Jupyter Notebook огромным количеством ячеек
- Проверяйте промежуточные результаты. Это абсолютно нормально
- Комментируйте код. Это значительно облегчит работу (особенно через какое-то время)
- Используйте автодополнение (Tab)
- Не тратьте на задачу (из ДЗ) больше 20 минут. Не получается – спросите у коллег или аспирантов. Нет смысла сидеть подолгу, мы только учимся.
- **Пишите так, чтобы другие могли понять ваш код**

БАЗОВЫЕ МОМЕНТЫ

В первую очередь для написания программ понадобятся переменные. Типов много, но самыми базовыми являются:

- Integer (целые числа: 2; -1; 155845)
- Float (вещественные числа: 2,1; -1,5; 155845, 2134)
- String (строки: «мама мыла раму»)
- Boolean(логические: true или false)

Строки удобны тем, что туда можно поместить практически всё. Использовать одинарные или двойные кавычки – на ваш выбор. Используйте экранирование (\), чтобы поместить внутрь строки кавычки (можно также банально использовать разный тип кавычек) или использовать служебные операторы (например, для переноса строки: \n).

УСЛОВНЫЕ ОПЕРАТОРЫ

После определения переменных понадобятся условия, которые будут определять то или иное действие. Для этого существует конструкция if-else:

```
if условие_1:
    действие_1
elif условие_2:                    #можно опускать, если условие одно
    действие_2
else:
```

```
    действие_3
```

Пример:

```
number = -2
if number != 0:
    print(“нуль”)
elif number > 0:
print(“положительное”)
else:
print(“отрицательное”)
```

Важно: в Jupyter Notebook необязательно писать print() для вывода. По умолчанию выводится последнее значение:

Пример:

```
number_1 = 1
number_2 = 2
number_3 = number_1 + number_2
print(number_3)          # писать необязательно, всё равно выведется number_3
```

ОПЕРАТОРЫ СРАВНЕНИЯ

В предыдущем примере был использован оператор сравнения <>, который является не единственным:

- > (больше)
- < (меньше)
- >= (больше или равно)
- <= (меньше или равно)
- == (равно)
- != (на равно)

Каждая подобная операция возвращает Boolean.

Пример:

```
4 == 1          #false
5 <= 6          #true
```

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Их всего два:

- and (и)
- or (или)

Они позволяют комбинировать условия по правилам алгебры логики. Это легко показать на булевых значениях:

And		
Результат_1	Результат 2	Итог
true	false	false
false	true	false
true	true	true
false	false	false

Or		
Результат_1	Результат 2	Итог
true	false	true
false	true	true
true	true	true
false	false	false

Проще говоря:

- при **and** будет **true** тогда, когда **оба значения true**
- при **or** будет **true** тогда, когда **хотя бы одно из значений true**

Для выполнения домашнего задания понадобится функция len(), которая вычисляет длину того, что находится в скобках. Применяется к строкам, массивам и т.д.

Пример

```
len('м а м а  м ы л а  р а м у')          #14
```

РАБОТА С GIT

Для выполнения домашних заданий необходимо готовые файлы загружать в облачное хранилище через систему контроля версии Git. Хранилища есть разные, мы будем использовать GitHub как самое популярное. Рекомендуемый способ работы – через консоль, но если будут сложности, то существуют графические интерфейсы (GitKraken) или загрузка через сайт. Однако, настоятельно рекомендуется использовать именно командную строку, так как это облегчит дальнейшее взаимодействие с разработчиками. Кроме того, некоторые действия можно выполнить лишь там. Подробные инструкции и видео находятся в личном кабинете.