**Вводное занятие в язык программирования Python**

Всем привет! Сегодня мы будем изучать **Python**.

Нам нужно разобрать несколько вводных моментов на языке Python:

Как язык Python вообще появился? Кто его создатель? Для чего он был задуман?

**Как язык Python появился?**

Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом. Для распределенной ОС Amoeba требовался расширяемый скриптовый язык, и Гвидо начал писать Python на досуге, позаимствовав некоторые наработки для языка ABC (Гвидо участвовал в разработке этого языка, ориентированного на обучение программированию, и по этой причине в Python вполне можно обучать программированию детей и он куда проще по синтаксису, чем большинство других языков программирования с такими же возможностями).Уже где-то к 1991 году язык был относительно готов, и Гвидо опубликовал свою первую редакцию. С самого начала Python проектировался как объектно-ориентированный язык, однако с другой стороны, само ООП лично Гвидо не так сильно нравится, и некоторые ООП идиомы не появляются в Python. Но не будем на этом особенно задерживаться, до ООП нам еще нужно дожить.

**Где используется Python?**

В огромном количестве сервисов. Стоит понимать, что на них вполне может быть написана только часть сервиса - но те, которые упомянуты на сайте, использую Python больше других. Но благодаря гибкости и широкой функциональности Python - отдел есть практически в любой крупной IT - компании

**Основные редакторы кода**

На самом деле для программирования на Python есть множество самых разных редакторов, каждый из которых несет свои плюсы и минусы. Мы постараемся разобрать основные их них, но на нашем курсе мы будем больше всего пользоваться **Sublime Text** и **Pycharm Community**( а так же хорошо знакомым вам **Google Collab**, которое уже было на первом занятии).

Итого, есть несколько вариант, которыми вы сможете пользоваться в том случае, если вас вдруг не устроят наши инструменты:

- ***VS Code***. Изначально VS Code был создан для веб разработки, и для этого он подходит куда лучше, потому что в нем хороший набор плагинов и расширений для всяких таких веб-штук, типа sass, css, html , pug, и так далее. На нем можно писать Python и даже запускать прямо из редактора, но не будет так удобно, как в PyCharm.

- ***Geany***. Редактор кода немного с другой историей, куда старше, чем VS code. Особенных причин пользоваться им сейчас нет, в нем также можно исполнять код прямо в редакторе, какие-то единственные отличия в том, что Geany немного страшнее внешне, хотя его можно настраивать.

- ***Pycharm Community***. Вне сомнения, самый сильный инструмент, у которого самые "умные" подсказки, мощная поддержка работы с классами. Не самый приятный интерфейс для новичка и потребуется довольно много времени, чтобы полностью настроить его под себя и привыкнуть. Однако, как говорится - выбор профессионала, которым точно нужно научиться пользоваться.

**Установка Python**

Нам все равно придется всем ставить Python, но первое занятие, если что, можно провести и Google Collab. Однако куда лучше, если все сделают это до занятия. Все инструкции можно найти в по этой ссылке:

<https://python-scripts.com/install-python-windows>

**Как мы будем сдавать домашки?**

Большую часть домашек мы сможем сдавать на **Github**. **Github** - это на самом деле только сайт, которые бесплатно предоставляет возможность хранить там свой код. Там есть как бесплатные, так и приватные репозитории, которыми в основном и пользуются команды, которым важна сохранить приватность кода, потому что там исходник алгоритма работы Yandex, как минимум, всех интересует даже не как сделать его клон, а как он работает, чтобы можно было не платить за рекламу и как-то поднимать свой продукт(здесь может быть какое-то отступление, которое не знаю, насколько всех интересует). Пока домашки мы будет только скидывать в архивах в нашем чате, если особенно не будет получаться, потому чтобы хотя бы на каких-то первоначальных порах освоить **Git**, надо начать работать с командной строкой и освоить самые базовые команды для **Git**.

**Интерпретируемые и компилируемые языки**

**(эта подчасть связанная с Computer Science)**

В Computer Science существует в основном два типа языков программирования: компилирумые и интерпретеруемые. Python относится к интерпретируемым, и это напрямую относится к его синтаксису, потому что компилируемые языки куда сложнее по своей структуре(такие типичные примеры, как C++ и C #, которые довольно сложные визуально, потому что там есть типизация данных); с другой стороны компилируемые языки "компилируются" непосредственно в машинный код, поэтому они практически всегда быстрее - и большинство сервисов, которые "быстрые" - игры, высоконагруженные сервисы, программы на компьютеры, такие как ворд, приложения на телефон( но там правда есть масса исключений) - построены на компилируемых языках.

Python относится к интерпертируемым и одним из самых медленных языков. Да, время честно сказать о недостатках. Но медленный не значит, что плохой - Python используется в куче всего полезного. Мало того, в корпорации Google его вместе с английским требуется машинное обучение и Python - это навроде как универсального языка для составления программ. Часто на Python делают прототипы - потому что его понимают большинство специалистов.

В частности, у Python есть следующие возможности:

* Работа с xml/html файлами
* Работа с http запросами
* GUI (построение графического интерфейса)
* Создание веб-сценариев, работа с FTP
* Работа с изображениями, аудио и видео файлами
* Программирование математических и научных вычислений, программирование нейросетей
* Построение полноценных серверов, микросервисов
* Огромная вариация средств для устранения рутинных действий и системного администрирования: кроссплатформенные программы для бекапов, создания файлов, чтения файлов, скаффолдинга проектов и т.д.

**Философия Python**

В языке Python есть своя философия. В других языках, как это не странно нет особой философии, она принимается скажем так скорее в каких-то компаниях ну или отдельных командах. Если вы уже поставили Python, тогда открываем командную строку (командная строка открывается с помощью WIN+R, и вводом **cmd** на Windows, и открытием программы Terminal на Mac. На маке вам потребуется открыть программа terminal, а лучше всего скачать **Iterm 2**, он просто топ, хотя бы из-за внешнего вида ).

Туда вводим следующие команды:

<code>

python

import this

</code>

После этого у нас выведется на английском философия Python:

Текст философии

* Красивое лучше, чем уродливое.
* Явное лучше, чем неявное.
* Простое лучше, чем сложное.
* Сложное лучше, чем запутанное.
* Плоское лучше, чем вложенное.
* Разреженное лучше, чем плотное.
* Читаемость имеет значение.
* Особые случаи не настолько особые, чтобы нарушать правила.
* При этом практичность важнее безупречности.
* Ошибки никогда не должны замалчиваться.
* Если они не замалчиваются явно.
* Встретив двусмысленность, отбрось искушение угадать.
* Должен существовать один и, желательно, только один очевидный способ сделать это.
* Хотя он поначалу может быть и не очевиден, если вы не голландец
* Сейчас лучше, чем никогда.
* Хотя никогда зачастую лучше, чем прямо сейчас.
* Если реализацию сложно объяснить — идея плоха.
* Если реализацию легко объяснить — идея, возможно, хороша.
* Пространства имён — отличная штука! Будем делать их больше!

Не будем обсуждать все эти утверждения, лучше обсудим, зачем это нужно. Программистам сложно принимать решения, и им нужно что-то, с помощью чего можно толковать эти понятия. Дзен именно для этого, что бы вы могли принять решение в спорной ситуации.

**Особенности синтаксиса Python**

В отличии от большинства других языков программирования, Python использует табуляцию для отделения синтаксических конструкций друг от друга. Другие языки, как, например, С++ или JavaScript, используют фигурные скобки для отделения синтаксических конструкций, таких, как классы, циклы, функции.

**Начинаем программировать**

Мы можем начать программирование на Python. И по классике, мы начнем с определения, что такое переменная, и как она работает. Переменная у нас - это “некоторая коробка” в которую у нас может складываться какой-либо значение для того, чтобы потом сделать с ним любую операцию, обновлять значения, и так далее. В Python представлены следующие типы данных:

- **None** - неопределенное значение переменной

-числовые типы данных(**Numeric Type**) - которые подразбиваются на различные типы данных, такие как int (целое число), float( Число с плавающей точкой), complex - это у нас комплексное число.

- **Списки** (Sequence Type)

**List** – список

**Tuple** – кортеж

**Range** – диапазон

**Строки** (Text Sequence Type )

**Str** (только один тип строковых данных)

**Бинарные списки** (Binary Sequence Types)

**bytes** – байты

**bytearray** – массивы байт

**memoryview** – специальные объекты для доступа к внутренним данным объекта через protocol buffer

**Множества** (Set Types)

**set** – множество - "контейнер", содержащий не повторяющиеся элементы в случайном порядке

**frozenset** – неизменяемое множество

**Словари** (Mapping Types)

**dict** – словарь. Словари это неупорядоченные коллекции произвольных объектов с доступом по ключу.

Бинарные списки мы пока брать не будем, потому что они достаточно узко применяются.

Давайте попробуем создать какие-то виды из этих переменных, чтобы попробовать

их в деле.

Создадим новую переменную, которая будет у нас строкой:

<code>

a = "str"

b = 4

</code>

Окей, неплохо, теперь у нас есть две переменные. Что будет, если мы сейчас попробуем их сложить? Выслушиваем варианты студентов( правильный ответ, что не смогут).

<code>

print(a+b)

</code>

Объясняем, что за print такой и что он делает в программах( типо это встроенная функция, и они нужна для вывода в консоль того, что у нас получается). Говорим, что во взрослом программировании она вполне может применятся для дебага программ, к примеру. Мы можем попробовать запустить эту программу, и мы получим вполне заслуженную ругань нашего интерпретатора. Если мы написали этот код в любом файле, к примеру, на рабочем столе, файлы можно создать с помощью следующих команд:

<code>

**touch test.py**  - создание файла на Linux или Mac

**copy NUL test.py** - Windows, создание "пустого" файла

</code>

Теперь мы займемся запуском нашей программы. Мы делаем это с помощью терминала по той причине, что этот способ немного короче и проще, чем дефолтное создание файлов на window, например.

Также можно привести следующие несколько правил, по которым у нас могут существовать переменные:

- Имена переменных могут состоять только из букв, цифр и символов подчеркивания. Они могут начинаться с буквы или символа подчеркивания, но не могут начинаться с цифры.

- Пробелы использовать нельзя, можно использовать только знаки подчеркивания.

- Нельзя использовать ключевые слова и имена функций в Python для этих целей( имен переменных).

Строки как типы переменных

Пришла пора познакомиться со строками. Строками в Python считаются все данные, которые стоят в кавычках;

Давайте сделаем простую программу, которая занимается тем, что может с нами поздороваться, узнать наше имя и вывести его:

<code>

print("привет, User!")

name = input("Представьтесь, пожалуйста!")

print("добрый день, "+ name)

</code>

Совсем неплохо. Но если мы случайно напишем свое имя с маленькой буквы, то программа у нас поздоровывается немного невежливо. Мы можем обратиться к встроенным функциям Python, для того чтобы запрограммировать только уважительное общение:

<code>

print("привет, User!")

name = input("Представьтесь, пожалуйста!")

print("добрый день, "+ name.title())

</code>

Теперь у нас получилась такая вежливая программа. Однако что сейчас произошло? Мы использовали встроенный метод title, который приводит первые буквы строки к заглавным. Немного дополним нашу программу, чтобы посмотреть, что такое конкатенация сама по себе. Конкатенация - это у нас объединение строк. Давайте теперь в нашей программе будет еще и представление фамилии пользователя:

<code>

print("привет, User!")

name = input("Введите ваше имя, пожалуйста!")

surname = input("Введите вашу фамилию, пожалуйста!")

print("добрый день, "+ name.title()+" "+ surname.title())

</code>

Кроме того, вы можете попробовать добавить другие методы работы со строками, чтобы, к примеру, очистить пробелы можно с помощью

strip()

<code>

print("привет, User!")

name = input("Введите ваше имя, пожалуйста!")

name = name.strip()

surname = input("Введите вашу фамилию, пожалуйста!")

surname = surname.strip()

print("добрый день, "+ name.title()+" "+ surname.title())

</code>

Отлично, теперь у нас программа будет достаточно неплохо здороваться с нами.

Давайте оставшееся у нас на занятии время потратим на том, чтобы попробовать поработать ещё и с цифрами, и наша программа могла например считать бюджет троих человек:

<code>

print("привет, User!")

name = input("Введите ваше имя, пожалуйста!")

surname = input("Введите вашу фамилию, пожалуйста!")

full\_name = name.title()+" "+ surname.title();

person1 = input("Введите данные первого человека")

person2 = input("Введите данные второго человека")

person3 = input("Введите данные третьего человека")

total\_count = int(person1) + int(person2) + int(person3)

print("Отлично " + full\_name + "итоговая сумма: " + str(total\_count) )

print("Отлично " + full\_name + "итоговая сумма: " + total\_count )

</code>

Можем немного поиграть со студентами, дать им поошибаться в вводе, потому

У нас получилась неплохая программа, которая допустим считает общий доход всей семьи.

**Какие книги вы советуете почитать?**

Изучаем Python. Программирование игр, визуализация данных, веб-приложения | Мэтиз Эрик

<https://www.ozon.ru/context/detail/id/166375074/>

Создаем нейронную сеть | Рашид Тарик

<https://www.ozon.ru/context/detail/id/141796497/>

Машинное обучение с использованием Python. Сборник рецептов | Элбон Крис

<https://www.ozon.ru/context/detail/id/155177800/>

В продолжении курса у нас будет множество других книг, но пока можете начать читать эти, потому что я знаю, что среди вас есть люди, которым как-то хочется учиться, так сказать, вместе с учебниками.