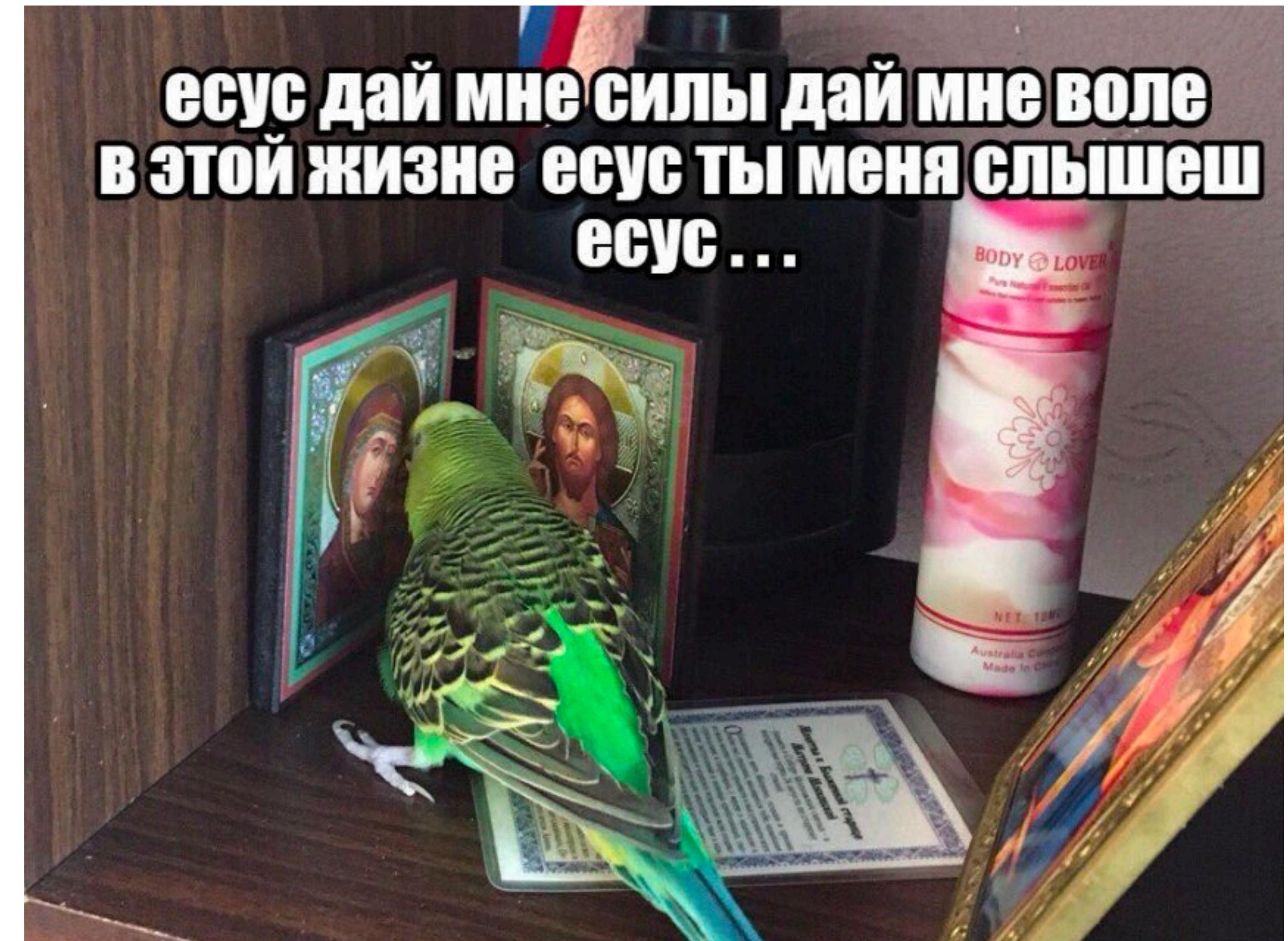


Продвинутый Python на ПМИ

05.09.2022

О себе

- Тимур (отчество говорить не буду)
- Закончил ПМИ, учусь в магистратуре ФТИАД (и еще на ФКН преподаю)
- Работал в Тинькофф, Озоне, Яндекс.Лавке
- На данный момент работаю в Яндекс.Плюс аналитиком девайсов по подписке
- (Люблю попугаев)



Контакты

- Группа в Telegram: <https://t.me/+-AWfNN2gvTg0ZmM6> (всегда поможем)
- Канал в Telegram: <https://t.me/+nJT3cEX3NmMzMJi>
- Мой контакт: @Palladain
- Куда сдавать домашки: ТВА

О курсе

- Каждый раз, когда вы приходите на работу, вам так или иначе потребуется Python. В зависимости от направления вам нужны разные инструменты, о которых вы либо не знаете ничего, либо так мельком смотрели
- И вот мы подумали: а что, если дать эти знания изначально? Можно все потрогать и даже понять, а чем вы хотите заниматься
- Так и родился этот курс :)



О курсе

- Курс идет 1-2 модуль
- Лекция + семинар каждую неделю (было бы славно, чтобы ходили)
- 4 домашки, 1 проект (экзамена не существует)
- Оценка: $0,1 * \text{ДЗ1} + 0,1 * \text{ДЗ2} + 0,15 * \text{ДЗ3} + 0,15 * \text{ДЗ4} + 0,5 * \text{Проект}$
- Округление арифметическое, блокирующих нет

О курсе

- ДЗ:
 - Ноутбук с 8-10 заданиями, часть - технические, часть - творческие (больше упор на исследование), бонусы
- Проект:
 - Задание в командах (3-4 человека, можно меньше при желании), идея - использовать те или иные инструменты, которые будут в курсе для создания продукта (бот в Телеге, сайт etc)
 - Список будет примерно в середине сентября, следите за анонсами (также можно предложить свою тему, если знаете, что хотите сделать)

Что вас ждет

- 4 блока:
 - Вычисления и визуализация (numpy, pandas, jax, matplotlib, plotly etc)
 - Веб-разработка и API
 - Многопоточность, тестирование, модули
 - Прочие полезные вещи

А что сегодня?



Сегодня в программе

- Shell + bash
- Git
- Среды разработки на Python
- Docker (на семинаре)

Начнем со скачиваний

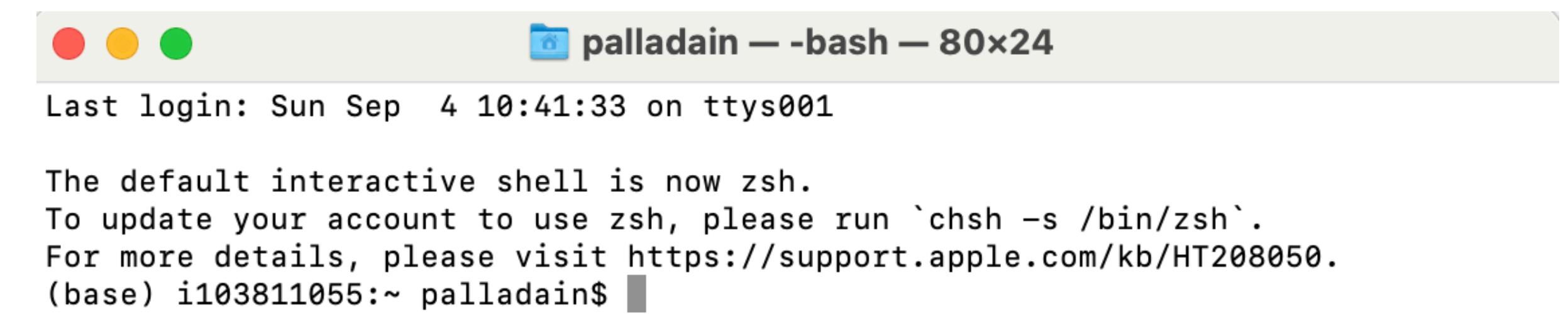
- Что установить:
 - Homebrew - отличная утилита для установления библиотек, потребуется примерно всегда (<https://brew.sh/>)
 - Docker (<https://docs.docker.com/engine/install/>)
 - Git (brew install git) - как вариант можно установить десктопную версию (<https://github.com/git-guides/install-git>)

Shell и как с ним жить

- Каждый так или иначе сталкивался с командной строкой (терминалом). Внутри терминала можно писать различные команды, с помощью которых вы можете работать с файлами (создать папку, удалить файл etc) или же посмотреть процессы, которые активны. По сути это и есть shell
- Shell - это командный интерпретатор внутри юникс системах (сори, с Виндой все сложнее, ребята, не стоит вскрывать эту тему, вы еще молоды)
- В нашем случае мы будем говорить про bash. По существу, это модернизированная версия Shell, которая на данный момент используется по дефолту

Bash

- Для того, чтобы работать с bash, достаточно открыть терминал
- Что мы сразу видим:
 - i103811055 - это машина, на которой мы находимся
 - Palladain - кто мы
 - ~ - где находимся (тильда означает корневую папку)
 - \$ - обозначение, что мы не боги (пока что)



```
Last login: Sun Sep 4 10:41:33 on ttys001
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) i103811055:~ palladain$
```

Что можно делать

- Самое базовое - выполнять команды непосредственно в терминале ->
- ssh - попасть на машинку
- cd - навигация по папкам
- cp/mv/mkdir/rm - манипуляции с файлами
- ls - посмотреть, что вокруг
- cat - превью содержимого
- grep - поиск по файлам
- echo - вывести содержимое
- sudo - стать богом
- htop - понять, что с ресурсами
- kill - убить процесс

Что можно делать

- \$PATH - путь поиска программ
- \$HOSTNAME - ваш hostname
- \$HOSTTYPE - архитектура машины
- \$OSTYPE - тип OS
- \$SECONDS - время работы скрипта (в секундах)
- \$HOME - домашний каталог пользователя
- Внутри bash есть также переменные, которые можно как задавать, так и использовать (обозначаются с помощью \$) ->

Что можно делать

- И вишенка на торте: можно писать скрипты и их выполнять!
- На семинаре мы разберем, как писать скрипты, комбинировать команды и как эффективно писать использовать этот инструмент



Git

- Каждый из вас явно сталкивался с GitHub (или Bitbucket/GitLab), и каждый из этих вещей - это репозиторий, где можно хранить свой код и чего угодно
- В чем же удобство репозитория? Допустим, что вы работаете в команде над одним общим проектом. Так или иначе, каждый вносит разные изменения в код, и для того, чтобы спокойно писать код и не иметь проблем, у каждого должна быть актуальная версия кода + хочется, чтобы внесенные изменения одного не рушили изменения другого. В этом как раз и заключается плюс таких репозиториев и с ними нужно уметь работать
- На всякий случай: **git != GitHub**

Git

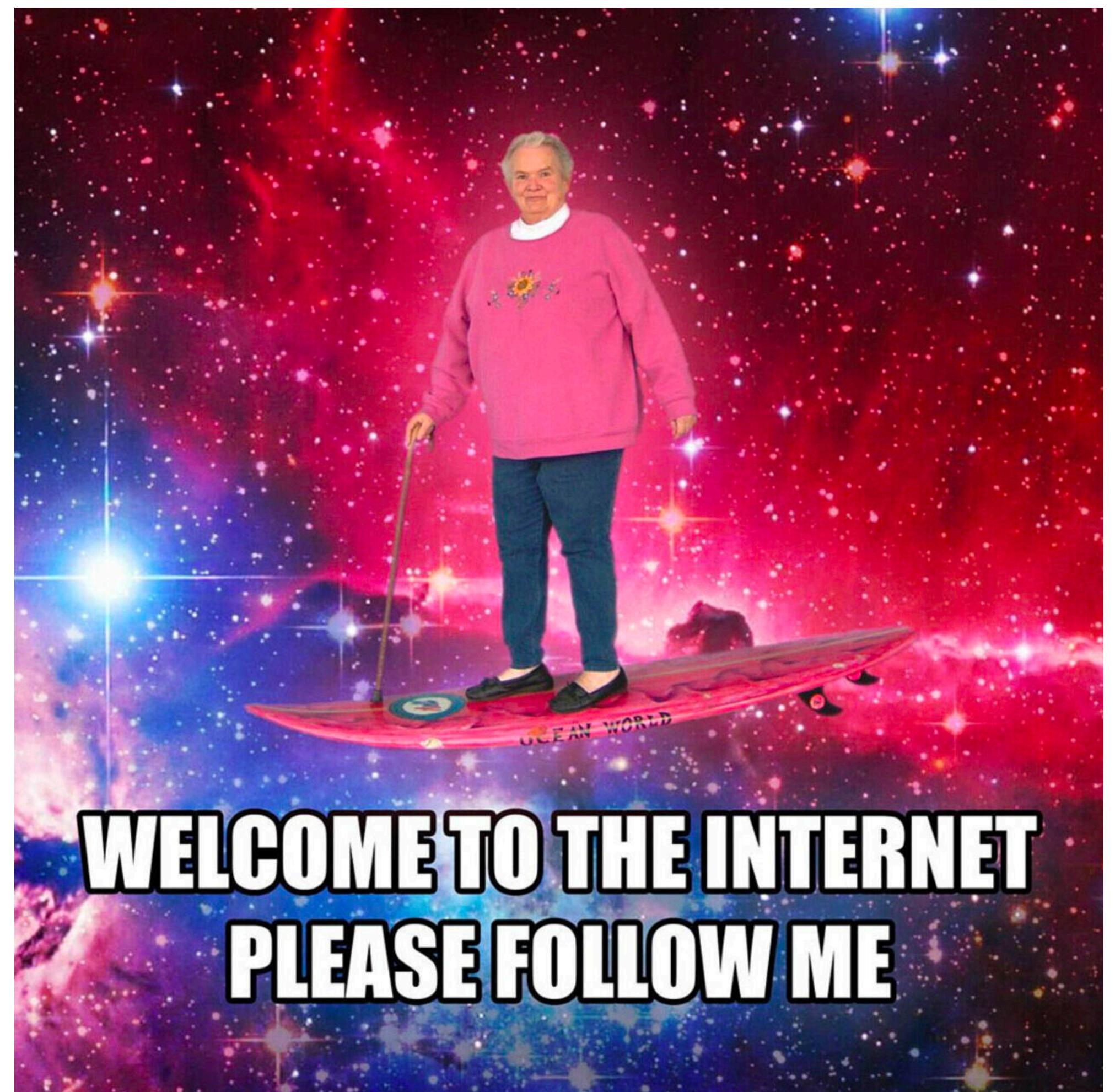
- Не будем вдаваться в глубину, как по сути своей работает git (разберем на семинаре), обсудим, как им пользоваться и что мы можем делать:
 - `git init` - создай репозиторий
 - `git status` - что происходит, где мы
 - `git add <file>` - добавь файл в коммит
 - `git commit -m "Hi"` - запушь коммит с комментарием (коммит - это запись изменений, которые случились по сравнению с исходной версией)
 - `git log` - посмотреть историю
 - `git diff version_1 version_2` - сравни две версии
 - `git checkout version_1` - вернись на другую версию

Сливаем ветки

- Теперь давайте рассмотрим ситуацию с командой. Один отвечает за правку багов, другой за введение новой фичи, при этом они не должны все пушить вместе. Для того, чтобы каждый мог комфортно работать, нужны ветви
- По сути ветви - разветвление от основной версии (`master`), которые мы потом посылаем на сервер и после аппрува изменения сливаются в основную версию
- `git branch <name>` - создаем ветку, после нужно на нее перейти (с помощью `checkout`)
- `git -b checkout <name>` - создание и переход сразу же
- `git merge` - сливаем с `master`

Отправляем на сервер

- А теперь надо все запушить в Github, чтобы это хранилось уже в общем доступе:
 - Git remote add <name> <link> - добавляем репозиторий, где хранить будем
 - Git push - пушим коммит
 - Git pull - забираем и делаем merge



Среды разработки на Python

- Скажем сразу: сред разработки огромное количество (и даже я не знаю всех сред)
- Но можно выделить самые популярные и поговорить про них:
 - Jupyter Notebook
 - PyCharm
 - Visual Studio Code

Jupiter Notebook

- Все, у кого был курс по финалу, уже знают, что это такое и с чем его едят (по крайней мере если вы делали лабы)
- В чём плюсы:
 - В разных ячейках можно запустить разный код и посмотреть, как он работает
 - Много визуализации
 - Достаточно прост в освоении
- Дополнительно: Google Colab (позволяет выполнять код на виртуальной машине, можно подключить GPU)

PyCharm

- Есть в подписке GitHub Student Developer Pack (следовательно, можно прямо использовать полную версию)
- В чём плюсы:
 - Очень мощный инструмент, отлично подходит для разработки
 - Куча дополнительных фич типа системы контроля версий, инспекции кода, дебагер
- Однако требует много ресурсов (8Гб оперативки, не везде пойдет)

Visual Studio Code

- Есть в подписке GitHub Student Developer Pack (следовательно, можно прямо использовать полную версию)
- В чём плюсы:
 - Подходит не только для Python, но и для других языков программирования
 - Имеет встроенный терминал (что очень удобно)
 - Потребляет не особо много памяти