Язык Python. Часть 2

Лекция 1

Планы на семестр

- Библиотеки
  - Пользуемся написанным
  - И немного пишем свои

А пока...

Вспомним про модули и пакеты

Простейший вариант

- Модуль, определенный по месту использования
- Отдельный файл
- В нем - код

Код модуля (fibo.py)

```
 1 def fibo(n):
 2     if n == 0:
 3         return 0
 4
 5     curr, previous = 1, 0
 6
 7     for i in range(1, n):
 8         curr, previous = curr + previous, curr
 9
10     return curr
```

Импортирующий код (fibo.py)

```
1 import fibo
2
3 for i in range(10):
4     print(i, fibo.fibo(i))
```

- Хорошо внутри одного проекта
- Плохо - когда хотим использовать в разных проектах
  - Копировать замучаемся
  - И поддерживать изменения

А если все-таки разные проекты...

- Переменные окружения
- Стандартный модуль sys

Переменная окружения/environment variable

- Механизм операционной системы
- Напоминает Python-переменную
- Именованная строка
- Есть в Windows, MacOS, Linux

Какие переменные окружения интересны нам

- PYTHONHOME
- PYTHONPATH

PYTHONHOME

- Путь к папке, куда установлен Python
- Заходим в что-то типа lib/python3.10
- Ищем там
- Более точно: https://docs.python.org/3/using/cmdline.html#envv PYTHONHOME

PYTHONPATH

- Список путей, где хотим хранить библиотеки
- Список - не в питоновском смысле
- Элементы разделяются
  - Windows - точкой с запятой
  - Linux, MacOS - двоеточием
- Более точно: https://docs.python.org/3/using/cmdline.html#envv PYTHONPATH

Внутри Python

- Есть модуль sys
- В нем - список sys.path
- Тут - уже в питоновском смысле
- Более точно: https://docs.python.org/3/library/sys.html#sys.path

Как все работает вместе

- При запуске проверяется наличие PYTHONHOME
- Если есть - берется его значение
- Если нет - вместо него берется путь к интерпретатору
- Из этого строятся несколько путей
- И добавляются в sys.path

Как все работает вместе

- Потом проверяется PYTHONPATH
- Если есть - значение разбивается по разделителю
- И полученные элементы добавляются в sys.path
- В начало

Как все работает вместе

- В самое начало добавляется путь, по которому лежит запущенный скрипт
- И это все - в первом приближении
- Более точно: https://docs.python.org/3/library/sys_path_init.htm

В итоге...

- В начале работы получаем sys.path
- Содержимое которого полностью определяет поиск модулей
  - При необходимости загрузить модуль идем по списку
  - Ищем файл в каждом из каталогов
  - Если не нашли нигде - бросаем исключение
- Можем по ходу действия управлять

содержимым sys.path

- Тем самым - управляя поиском модулей

Схема установки сторонней библиотеки

- Находим
- Скачиваем архив
- Распаковываем
- Копируем туда, где ее найдет python
- Используем

Хотелось бы автоматизировать

- Заведем служебную папку
- В нее будем скачивать
- В ней же распакуем
- Скопируем в то место, где ее найдет python
- Но есть проблема в нахождении - о ней чуть позже

А если одна библиотека зависит от других...

- Надо скачивать другие
- Изначально - руками
- Но это как-то грустно
- Можно автоматизировать
- И здесь - тоже проблема в нахождении

А еще бывают версии библиотек

- Более ранние и более поздние
- Более поздние - более надежные и развитые
- Но постоянно гнаться за совсем последними - не всегда оправданно
- Надо как-то ориентироваться
- Принято нумеровать

Нумерация версий

- Популярный подход - семантическое версионирование
- MAJOR.MINOR.PATCH
- Например, 2.4.12
- MAJOR - крупные изменения, иногда несовместимые с прошлыми версиями
- MINOR - совместимые изменения, новая функциональность
- PATCH - мелкие правки

Обработка версий

- Простор для автоматизации
- Отслеживаем, какие версии библиотек установлены
- Выясняем, версии зависимостей - при установке новой библиотеки
- Автоматически их скачиваем и устанавливаем

Автоматизация

- Можно представить себе программу, которая берет на себя рутинную работу
- И она сама может быть модулем, сторонней библиотекой
- Мы ее руками установим один раз
- А дальше - она сама пускай все устанавливает
- В том числе - свои же обновления

Остался вопрос - как находить

- Можно по URL
- Это удобно и гибко
- Но есть проблема - это слишком гибко
- Произвольный ресурс может перехать, или даже исчезнуть
- А от него другие библиотеки зависят

Нужен какой-то реестр

- Чтобы библиотека называлась именем
- И в ней хранилось знание о всех версиях
- И где их скачать
- И ресурсы для хранения
- Итого: нужен установщик и инфраструктура хранения

Итак, нужны

- Установщик библиотек
- Инфрастуктура хранения
- Популярность инфраструктуры
- И такой популярный установщик с инфраструктурой есть

PIP

- Package Installer for Python
- Package Installer Package
- Инфраструктура: https://pypi.org/
  - 427 595 проектов - на 11.01.2022
- Когда он есть - он сам себя умеет обновить
- Когда его нет - его надо поставить
  - Установка зависит от системы - просто надо погуглить

Первая библиотека

- requests
  - В первом приближении - имитирует браузер
  - Позволяет скачать страничку
  - Или файл
  - Или отправить комментарий, загрузить файл

Можно так:

- `pip install requests`

Или так:

- `python3 -m pip install requests`

Отвлечемся от Python-а

- Поговорим о сетях
  - О протоколах
  - О сайтах
  - Об HTTP и HTML
  - О json и REST API

Адреса, порты, соединения

- IP-адрес: 4 байта
  - Вообще-то, это маловато
  - Но есть IPv6
  - Есть другие решения
- Они трудно запоминаются
  - Какой поисковик вам больше нравится: 216.58.207.228 или 87.250.250.242 ?

DNS

- Есть специальный сервис
  - Domain name system
- Он дает нормальные имена
- И знание о том, каким адресам они соотвествуют
  - Какой поисковик вам больше нравится: www.google.com или www.ya.ru ?

Порты

- Достаточно или IP-адреса ?
- А если на одной машинке одновременно работают
  - Web-сервер
  - Почтовый сервер
  - ssh - чтобы можно было залогиниться
  - git-сервер
  - Парочка специфичных серверов с бизнес-логикой

Порты

- Вводим понятие "порт"
- Это что-то вроде номера квартиры в доме
- Или абонентского ящика
- Многим стандартным сервисам назначены выделенные номера портов
- Веб-сервера работают обычно на порту 443

Адрес, имя и порт

- Чтобы начать общаться с каким-то приложением нужно
  - Знать его IP-адрес или DNS-имя
  - Знать его порт
  - Приложение должно быть запущено на машинке с таким адресом
  - И слушать этот порт