Лекция 8

Язык программирования Python.

Хайрулин Сергей Сергеевич

email: s.khairulin@g.nsu.ru, s.khayrulin@gmail.com

Ссылка на материалы

План

- Лекции/практические занятия
  - Тест
- Дифференцированный зачет в конце семестра
  - Защита задания

Литература

Начальный уровень

- Mark Pilgrim. Dive into Python - http://www.diveintopython.net/
- Марк Лутц. Изучаем Python, 4-е издание // Символ-Плюс 2011.
- ...

Стандарт/Документация

- PEP-8 - https://www.python.org/dev/peps/pep-0008/
- https://www.python.org/
- https://github.com/python/cpython

Экспертный уровень

- Лучано Рамальо: Python. К вершинам мастерства
- Mitchell L. Model. Bioinformatics Programming Using Python // O'Reilly 2010.

Версии Python

- Python 2 вышел 2010 году последняя версия 2.7.16 - исправлялись только баги(ошибки) с января 2020 года поддержка прекращена.
- Python 3 в появился в 2008, является актуальной версией языка. Текущая стабильная версия 3.9, в разработке 3.10
  - Python 3 не гарантирует совместимости кода с Python 2

Summary

- Организация кода
  - Модули
  - Пакеты, файл __init__.py
  - Зависимости:
    - Ключевое слово import
    - Конструкция from … import …
  - PYTHONPATH
  - Точка входа в приложение
- работа с файлами

Организация кода

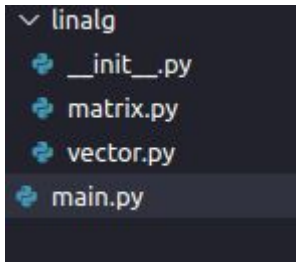
С ростом сложности приложения возникает необходимость упорядочивания кода. Уровни абстракции кода:

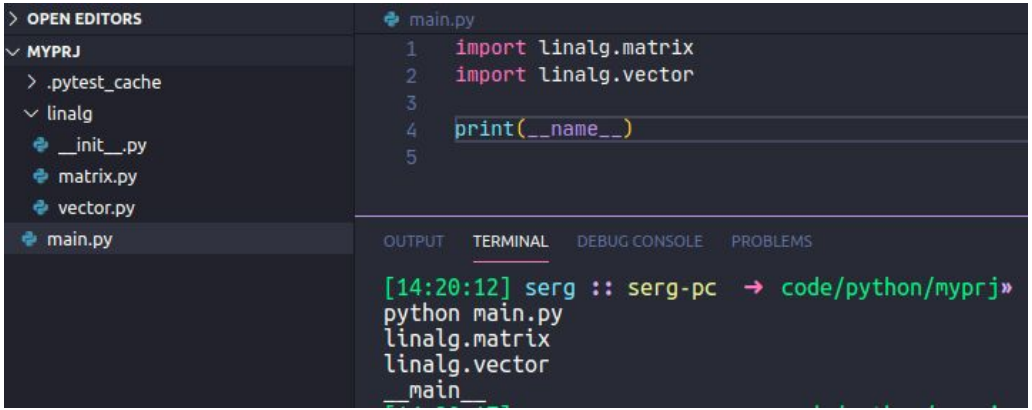
1. Выделение повторяющиеся куски кода в циклы
2. Выделение функций, определенный набор операция реализующий некоторую последовательность вычислений или часть алгоритма, зависящая от входных данных
3. Организация функций в отдельные модули (обычно файлы.)
4. Определение абстрактных структур данных.

Модуль

Модуль - неделимую сущность объединяющий в себе некоторый набор переменных, функций, классов…

A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended. Within a module, the module's name (as a string) is available as the value of the global variable __name__. (https://docs.python.org/3/tutorial/modules.html)

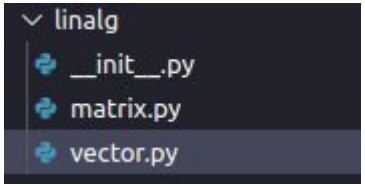




Пакеты

Компиляция модулей в пределах одной папки называется пакетом

Packages are a way of structuring Python's module namespace by using "dotted module names". For example, the module name `A.B` designates a submodule named `B` in a package named `A`. Just like the use of modules saves the authors of different modules from having to worry about each other's global variable names, the use of dotted module names saves the authors of multi-module packages like NumPy or Pillow from having to worry about each other's module names. (https://docs.python.org/3/tutorial/modules.html#packages)

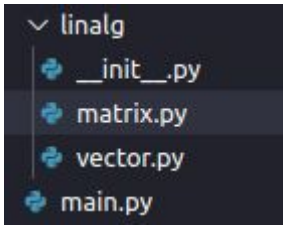


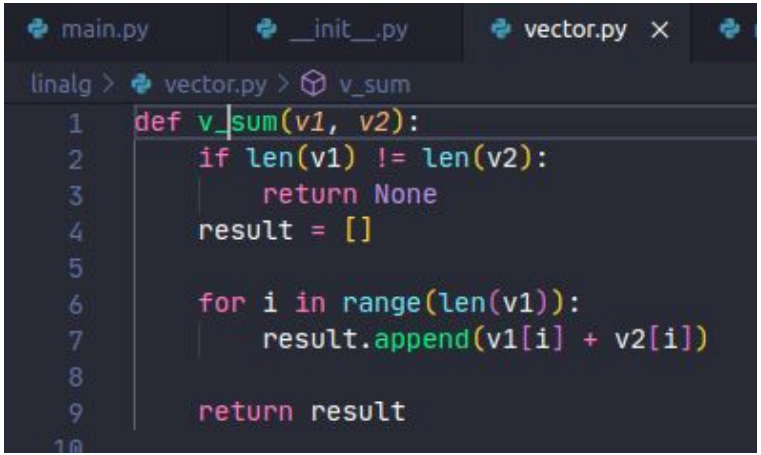
Пакеты (__init__.py)

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

Зависимости: import

Для того чтобы воспользоваться функциями определенными в некотором пакете можно импортировать их в модуль где вы собираетесь их использовать.





Зависимости: from … import ...

Запись импорта from package_name....module_name import functions, class, var,....

linalg > vector.py > ...

```python
def v_sum(v1, v2):
    if len(v1) != len(v2):
        return None
    result = []

    for i in range(len(v1)):
        result.append(v1[i] + v2[i])

    return result


def scalar_mult(scalar, v):
    result = []
    for i in range(len(v)):
        result.append(scalar * v[i])

    return result
```

main.py > ...

```python
import linalg.matrix

from linalg.vector import v_sum, scalar_mult

v1 = [1, 2, 3]
v2 = [4, 5, 6]
print(v_sum(v1, v2))
print(scalar_mult(2, v1))
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[16:14:48] serg :: serg-pc  → code/python/myprj»
python main.py
[5, 7, 9]
[2, 4, 6]
```

PYTHONPATH

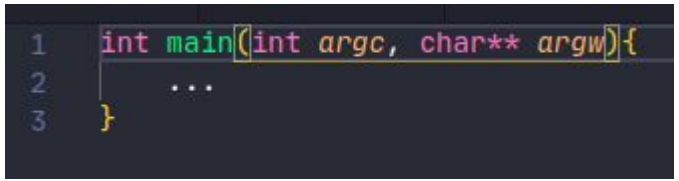
Разрешение зависимостей происходит следующим образом:

- The directory containing the input script (or the current directory when no file is specified).
- `PYTHONPATH` (a list of directory names, with the same syntax as the shell variable `PATH`).
    a. PYTHONPATH=/usr/local/lib/python/lib/:/home/sergey/python/lib

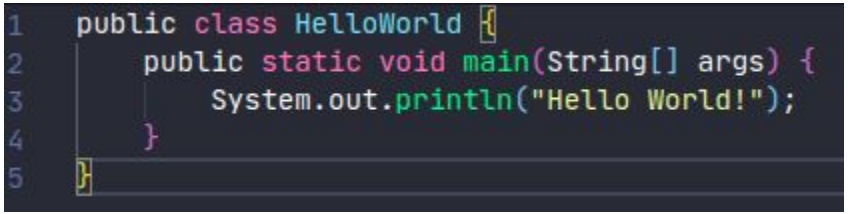
https://docs.python.org/3/using/cmdline.html?highlight=pythonpath#envvar-PYTHONPATH

Точка входа в приложение

C/C++



Java



Go

Точка входа в приложение

vector.py

```python
def v_sum(v1, v2):
    if len(v1) != len(v2):
        return None
    result = []

    for i in range(len(v1)):
        result.append(v1[i] + v2[i])

    return result


def scalar_mult(scalar, v):
    result = []
    for i in range(len(v)):
        result.append(scalar * v[i])

    return result


v1 = [1, 2, 3]
v2 = [4, 5, 6]
print(v_sum(v1, v2))
print(scalar_mult(2, v1))
```

Python выполняет код последовательно.

main.py | __init__.py | vector.py | matrix.py | func main(){ Untitled-1

```python
import linalg.matrix

from linalg.vector import v_sum, scalar_mult

v = list(map(float, input("provide vector separate with coma: ")))
scalar = input("provide scalar value (should be number): ")
print(scalar_mult(scalar, v))
```

OUTPUT | TERMINAL | DEBUG CONSOLE | PROBLEMS

```
[16:53:50] serg :: serg-pc  → code/python/myprj»
python main.py
[5, 7, 9]
[2, 4, 6]
provide vector separate with coma: 1,2,3
provide scalar value (should be number): 22
[22.0, 44.0, 66.0]
```

Точка входа в приложение

```python
def v_sum(v1, v2):
    if len(v1) != len(v2):
        return None
    result = []

    for i in range(len(v1)):
        result.append(v1[i] + v2[i])

    return result


def scalar_mult(scalar, v):
    result = []
    for i in range(len(v)):
        result.append(scalar * v[i])

    return result


if __name__ == '__main__':
    v1 = [1, 2, 3]
    v2 = [4, 5, 6]
    print(v_sum(v1, v2))
    print(scalar_mult(2, v1))
```

main.py > ...

```python
import linalg.matrix

from linalg.vector import v_sum, scalar_mult

if __name__ == '__main__':
    v = list(map(float, input("provide vector separate with coma: ")))
    scalar = input("provide scalar value (should be number): ")
    print(scalar_mult(scalar, v))

```

Работа с файлами

vector.py

```python
def v_sum(v1, v2):
    if len(v1) != len(v2):
        return None
    result = []

    for i in range(len(v1)):
        result.append(v1[i] + v2[i])

    return result


def scalar_mult(scalar, v):
    result = []
    for i in range(len(v)):
        result.append(scalar * v[i])

    return result

def load_vector_from_file(f_name):
    v = []
    with open(f_name, "r") as f:
        for l in f:
            v.append(float(l))
    return v
```

main.py

```python
import linalg.matrix

from linalg.vector import v_sum, scalar_mult, load_vector_from_file


if __name__ == '__main__':
    f_list = input("Provide file name list: ").split(",")
    v1 = load_vector_from_file(f_list[0])
    v2 = load_vector_from_file(f_list[1])
    print(v_sum(v1, v2))
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[17:18:01] serg :: serg-pc  → code/python/myprj»
python3 main.py
Provide file name list: 1.vec,2.vec
[5.0, 7.0, 9.0]
[17:19:05] serg :: serg-pc  → code/python/myprj»
```

Работа с файлами

Функция open(path_to_file, mode) - открытие файла

path_to_file - путь до файла подходит как полный, так и относительный

mode - может быть:

- **w** - на запись в этом случае содержимое файла обнулиться,
- **r** - на чтение,
- **r+** - открывает файл и для чтения и записи,
- **a** - на запись в конец файла

Задачи

1. Работа с файлами