


# Модули





## Зачем?

- упростить задачи проектирования программы и распределения процесса разработки между группами разработчиков;
  - предоставить возможность обновления (замены) модуля, без необходимости изменения остальной системы;
  - упростить тестирование программы;
  - упростить обнаружение ошибок.
- 

# Модуль, Пакет

- Модуль (англ. Module) - специальное средство языка программирования, позволяющее объединить вместе данные и функции и использовать их как одну функционально-законченную единицу (например, математический модуль, содержащий тригонометрические и прочие функции, константы , и т.д.).
- Пакеты (англ. Package) являются еще более крупной единицей и представляют собой набор взаимосвязанных модулей, предназначенных для решения задач определенного класса некоторой предметной области (например, пакет для решения систем уравнений, который может включать математический модуль, модуль со специальными типами данных и т.д.).

# Модуль, Пакет

- Модуль - отдельный файл с кодом на Python, содержащий функции и данные:
- имеет расширение \*.py (имя файла является именем модуля);
- может быть импортирован (подключен) (директива import ...);
- может быть многократно использован.
- Пакеты в Python - это способ структуризации модулей. Пакет представляет собой папку, в которой содержатся модули и другие пакеты и обязательный файл `__init__.py`, отвечающий за инициализацию пакета.

# 1. Встроенные

Встроенные (англ. Built-in).

1. Модули, встроенные в язык и предоставляющие базовые возможности языка (написаны на языке Си).
1. К встроенным относятся как модули общего назначения (например, `math` или `random`), так и платформозависимые модули (например, модуль `winget`, предназначенный для работы с реестром ОС Windows, устанавливается только на соответствующей ОС).



# Стандартная библиотека

- Стандартная библиотека (англ. Standard Library).
- Модули и пакеты, написанные на Python, предоставляющие расширенные возможности, например, json или os.



## Сторонние

- Сторонние (англ. 3rd Party).
- Модули и пакеты, которые не входят в дистрибутив Python, и могут быть установлены из каталога пакетов Python (англ. PyPI - the Python Package Index, более 90.000 пакетов) с помощью утилиты pip.



# Пользовательские

- Пользовательские (собственные).
- Модули и пакеты, создаваемые разработчиком.



# Подключение и использование

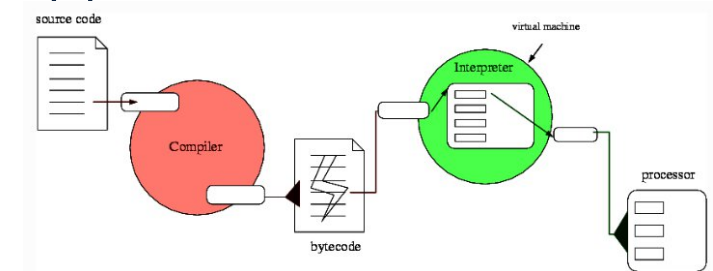
- Import
- При импорте модуля или пакета Python выполняет его поиск в следующем порядке:
- 1. Встроенный модуль?
- 2. Файл `module.py` / `module.pyc`, или пакет `package` есть в списке путей переменной `sys.path`?
- `sys.path` - список, при инициализации включающий:
- рабочую директорию скрипта (основного модуля);
- переменную окружения `PYTHONPATH` и пути инсталляции Python.

# Аттрибуты

- Каждый модуль имеет специальные и дополнительные атрибуты.
- Специальные атрибуты содержат системную информацию о модуле (путь запуска, имя модуля и др.) и доступны всегда. Некоторые из них:
  - `__name__`
  - Полное имя модуля.
  - Пример: "math" или "os.path".
  - `__doc__`
  - Строка документации.
  - `__file__`
  - Полный путь к файлу, из которого модуль был создан (загружен).
- Дополнительные (необязательные) атрибуты могут содержать справочную информацию об авторе, версии модуля и т.д. и имеют следующие обозначения: `__author__`, `__copyright__`, `__credits__`, `__license__`, `__version__`, `__maintainer__`, `__email__`, `__status__`.

# Кэширование («компиляция») модулей

- Python считается интерпретируемым языком, однако имеет гибридную составляющую: при запуске программы происходит компиляция в промежуточный байт-код (бинарный файл с расширением \*.рус) всех импортированных модулей, после чего код выполняется виртуальной машиной



- Особенности \*.рус-файлов:
- имеют бинарный формат;
- загружаются быстрее, чем исходные \*.py-файлы; при этом скорость выполнения одинакова в обоих случаях;
- создаются в папке `__pycache__` и имеют имя `имя_модуля.cpython-35.рус` (для реализации CPython), где 35 - версия Python;
- «привязаны» к версии интерпретатора (обязательно в пределах основной версии - 2 или 3) и платформе компиляции.

# Модули

- Любой файл с расширением .py является модулем
- PEP8 – соглашение об именовании и структуризации кода:
- змеиный\_регистр для наименования модулей: first\_module;
- строки документации.

```
def printing():  
    '''Ptinting of simple imformation'''  
    print("Hello from the first module")
```

```
import first_module.print as fp  
import second_module.print  
  
if __name__ == '__main__':  
    print(fp.printing.__doc__)  
    fp.printing()  
    second_module.print.printing()
```

```
PS C:\Users\pixpl\Downloads\practice_4> py -m module_example  
Ptinting of simple imformation  
Hello from the first module  
Hello from the second module
```

# Запуск модуля

- Любой модуль в Python может быть:

```
PS C:\Users\pixpl\Downloads\practice_4> py -m module_example > results.txt
PS C:\Users\pixpl\Downloads\practice_4> █
```

- запущен автономно (как скрипт, например, в командной строке или через IDE);
- импортирован (через import).

```
PS C:\Users\pixpl\Downloads\practice_4> py -m module_example
```

```
import first_module.print as fp
import second_module.print

if __name__ == '__main__':
    print(fp.printing.__doc__)
    fp.printing()
    second_module.print.printing()
```

- Чтобы выполнить различный код в зависимости от того, запущен модуль или импортирован, достаточно использовать специальный идентификатор `__name__`, который содержит:
- имя модуля, если он был импортирован (например, "first\_module" или "math");
- специальное наименование `"__main__"`, если модуль был запущен автономно

# Список доступных модулей в пакете

```
__init__.py × print.py first_module × module
list_processing > __init__.py > ...
1
2  #__all__ = ['print', 'processing']
3  __all__ = ['print']
```

```
#from list_processing import *
```