Модули (Modules)

Модуль – это файл .py с набором функций

mymodule.py:

```
def say_hello():
    print('Hello! It's module mymodule.py')

__version__ = '0.1'
# End of module mymodule.py
```

test.py:

```
import mymodule
mymodule.say_hello()
print ('Version', mymodule.__version__)
```

Подключение модулей

- Импорт всех функций и переменных модуля

```
import mymodule # import all functions
mymodule.say_hello()

import mymodule as mm # use alias
mm.say_hello()

from mymodule import * # bad style
say_hello()
```

- Импорт отдельных функций и переменных

```
from mymodule import say_hello,say_goodbye
say_hello()
```

Магический атрибут `__all__`

Список имен модуля, загружаемых через `from mymodule import *`, может быть определен в магическом атрибуте `__all__` модуля

`mymodule.py`:

```
__all__ = ['say_hello']
```

`test.py`:

```
from mymodule import *
say_hello() # ok
say_goodbye() # error: undefined name
```

Если атрибут `__all__` не определен, то подгружаются все методы и переменные модуля, кроме тех, которые начанаются с символа подчеркивания (_)

Подключение модулей. Замечания

- В момент начала работы программы Python загружает в память модуль `__builtins__`, содержащий базовые функции и переменные
- Список подгружаемых имен модуля возвращает функция **`dir`**:

```
import mymodule
print(dir(mymodule))
```

- Загрузка модуля возможна только один раз, при первом вызове **`import`**. Для повторной загрузки модуля используется библиотека `importlib`

```
import importlib
importlib.reload(mymodule)
mymodule.say_hello()
```

Исполняемый код в модулях

Модуль может содержать исполняемый код

`fibo.py`:

```
def fib(n):
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result

print("It's module fibo") # executable statement
```

Код модуля выполняется только один раз при первом подключении модуля. При повторной загрузке модуля (с помощью `importlib.reload`) код модуля выполняется повторно

Выполнение модуля как скрипта

У каждого модуля (файла) имеется магический атрибут `__name__`, которому присваивается имя модуля при импорте

```
import math
print(math.__name__) # math
```

Когда файл исполняется как скрипт, значению `__name__` присваивается строка `'__main__'`

```
def sq(x)
    return x*x

if __name__ == "__main__":
    # do if called as a script
    print(sq(5))
```

Пакеты (Packages)

Пакеты – это каталоги с модулями и специальным файлом `__init__.py`, который показывает Python, что этот каталог особый, так как содержит модули Python

Пакеты – удобный способ иерархической организации модулей

```
| — <some folder in sys.path>/
| |——— world/
|       |——— __init__.py
|       |——— asia/
|       |     |——— __init__.py
|       |     |——— india/
|       |           |——— __init__.py
|       |           |——— foo.py
|       |——— africa/
|             |——— __init__.py
|             |——— madagascar/
```

Пакеты. Пример

Например, пакет sound содержит в себе три модуля:
wavread.py, wfilter.py и surround.py

```
sound
|-- __init__.py
|-- wavread.py
|-- wfilter.py
|-- surround.py
```

```
# import module wavread.py from package sound
import sound.wavread
w = sound.wavread.read(fname) # usage
import sound.wfilter as filt
res = filt.equalizer(w, n)
from sound import surround
res = surround.process(w)
```

Файл __init__.py

Файл __init__.py может быть пустым или может содержать переменную __all__, хранящую список модулей, который ся при загрузке через конструкцию

```
import *
```

ла __init__.py

```
__all__ = ["wavread", "wfilter", "surround"]
```

Файл __init__.py позволяет инициализировать переменные пакета, подключить модули и пр.

Пример файла __init__.py

```
import sys
if sys.version_info[0] >= 3:
    ...
```