



Модули и пакеты

Технологии и языки программирования

Юдинцев В. В.

Кафедра теоретической механики
Самарский университет

17 декабря 2017 г.

- 1 Модули
- 2 Пакеты (Packages)
- 3 Полезные модули
 - math
 - numpy
 - matplotlib
 - scipy
 - pandas
 - sympy

Модули

Модульное программирование

- Модульное программирование — это организация программы как совокупности небольших независимых блоков — модулей.
- Использование модульного программирования позволяет упростить тестирование программы и поиск ошибок.

Модули Python

- В языке Питон модуль — это файл с python-кодом.
- Модуль может содержать переменные, объявления функций и классов.
- Модуль может содержать и исполняемый код.

Импорт модуля

Файл **my_module.py** содержит определение двух функций:

```
1 def my_function_x2 (x) :  
2     return x**2  
3  
4 def my_function_x3 (x) :  
5     return x**3
```

Программа **main.py** использует функции, объявленные в модуле **my_module.py**

```
1 import my_module  
2  
3 x = 3.0  
4  
5 res1 = my_module.my_function_x2 (x)  
6 res2 = my_module.my_function_x3 (x)
```

Импорт модуля

Импорт имён из модулей `module1.py`, `module2.py`, `module3.py`

import `module1`, `module2`, `module3`

Пример: импорт математических функций из стандартного модуля **math**:

```
1 import math
2
3 a = math.sin(math.pi/4.0)
4 b = math.tan(math.radians(30.0))
```

При таком способе импорта модуля с именем функции необходимо указать и имя модуля (префикс): **math.sin**, **math.cos**

Импорт выбранных имён из модуля

Импорт функций `sin`, `cos`, `tan` и константы π из модуля `math`

from `math` **import** `sin`, `cos`, `tan`, `pi`

```
1 from math import sin , cos , tan , pi
2
3 a = sin(pi/4.0)
```

Имена из модуля используются без префикса — имени модуля.

Импорт всех имён из модуля

Импорт определений всех функций и констант из модуля `math`

from math import *

Все определения модуля `math`:

```
1 import math
2
3 a = sin(pi/4.0)
```

Импортированные имена из модуля используются без префикса — имени модуля. При таком способе импорта возможно перекрытие имён, если два модуля предоставляют для импорта одно и то же имя для функции или объекта.

Псевдонимы модулей

При импорте модулей им можно давать новые имена-псевдонимы:

```
1 import numpy as np
2 import scipy.linalg as ls
3
4 A = np.matrix( [[1, 2, 3], [4, 5, 6], [3, 4, 1]] )
5 B = np.matrix( [[1], [2], [3]] )
6
7 x = ls.solve(A,B)
8 print(x)
```

Решение матричного уравнения $Ax = B$:

```
[[ -0.66666667]
 [  1.33333333]
 [ -0.33333333]]
```

Пакеты (Packages)

- Пакеты позволяют структурировать коллекции модулей, составляющих большие библиотеки.
- Пакеты формируются в виде иерархий каталогов с исходными кодами

Структура пакета

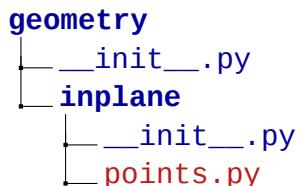
Пример пакета функций аналитической геометрии на плоскости и в пространстве:

```
geometry
├── __init__.py
├── inplane
│   ├── __init__.py
│   ├── points.py
│   ├── lines.py
└── spatial
    ├── __init__.py
    ├── points.py
    ├── lines.py
    └── planes.py
```

- Для того, чтобы транслятор Python “понял”, что просматриваемый им каталог является каталогом с модулями в корень такого каталога помещается файл `__init__.py`
- Файл `__init__.py` может быть пустым или содержать код инициализации пакета — код, который будет выполняться при импорте пакета

Импорт модулей пакета: `import`

Импорт модуля `points.py` пакета `geometry.inplane`



```
1 | import geometry.inplane.points
```

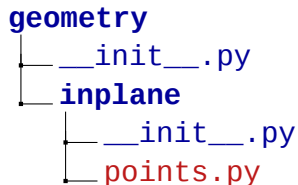
```
>> geometry.inplane.points.distance( (1,2) ,(7,3) )
>> 6.082762530298219
```

Фрагмент файла `points.py`

```
def distance(p1, p2):
    ...
    ...
```

Импорт модулей пакета: `from ... import`

Импорт модуля `points.py` пакета `geometry.inplane`

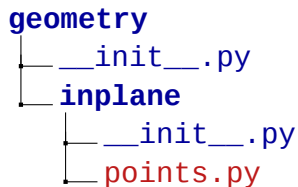


```
1 from geometry.inplane import points
2 print( points.distance( (1,2) ,(7,3) ) )
```

```
>> 6.082762530298219
```

Импорт модулей пакета: `from ... import`

Импорт **имени** (например, функции) из модуля `points.py` пакета `geometry.inplane` для использования имени без префикса модуля:



```
1 from geometry.inplane.points import distance
2
3 print( distance( (1,2), (7,3) ) )
```

```
>> 6.082762530298219
```


from ... import vs import

from package **import** item

item — пакет, модуль или имя, определённые в модуле **package**

import item.subitem. subsubitem

subsubitem — пакет или модуль, но **не имя** функции или переменной, определённые в subitem

from ... import *

Импорт всех имён, определённых в модуле

```
1 from mymodule import *
```

Если в модуле задана переменная `__all__`, которая определяет список импортируемых имен, то будет импортированы только имена из этого списка

```
1 __all__ = ( 'var1' , 'var2' )
2
3 var1 = 10
4 var2 = 15
5
6 var3 = 30
```

Если переменная `__all__` не определена, то импортируются **все** имена, не начинающиеся с нижнего подчёркивания

Поиск модулей транслятором

При импорте модуля транслятор Python ищет модуль (файл):

- среди встроенных модулей, таких как `math` (каталоги Python)
- в каталоге файла, который импортирует модуль
- в переменной окружения `PYTHONPATH`

Все каталоги поиска указаны в переменной `sys.path`:

```
>> import sys
>> print(sys.path)
>> [ '', '/home/user/Programs/anaconda3/lib/python3.5/site-packages/spyder/utils/site', '/home/user/Programs/anaconda3/lib/python3.5', '/home/user/Programs/anaconda3/lib/python3.5/platform-linux', '/home/user/Programs/anaconda3/lib/python3.5/lib-dynload', ... ]
```

Выполнение модуля как программы

- При импорте модуля код, находящийся вне определений функций, выполнится.
- Если необходимо разрешить выполнение такого кода только при запуске модуля как программы:

```
1 | > python.exe my_module.py
```

необходимо внутри модуля проверять значение переменной `__main__`:

```
1 | def my_function_B(a):  
2 |     ...  
3 | def my_function_A(a):  
4 |     ...  
5 | if __name__ == '__main__':  
6 |     print('Standalone running ...')
```

Полезные модули

Модуль math

- Математические функции для работы с вещественными числами (`float`)
- Модуль для работы с комплексными числами — `cmath`

Импорт модуля `math`

Импорт модуля

```
1 import math
```

```
2
```

```
3 a = math.sin(1.0)
```

```
4 print(a)
```

```
>> 0.8414709848078965
```

Импорт отдельных имён модуля

```
1 from math import sin, cos, tan, pi
```

```
2 a = sin(1.0)
```

```
3 b = cos(1.0)
```

Импорт всех имён модуля

```
1 from math import *
```

```
2 a = sin(1.0)
```

Импорт всех имён

```
1 from cmath import *
2 from math import *
3
4 a = sqrt(-0.5)
```

ValueError: math domain error

Имена модуля `cmath` переопределились модулем `math`, который импортируется последним

```
1 import math
2 import cmath
3
4 b = cmath.sqrt(-0.5)
```


Преобразование числа

<code>math.ceil(x)</code>	Округление до большего целого
<code>math.floor(x)</code>	Округление до меньшего целого
<code>math.trunc(x)</code>	Округление к нулю
<code>math.fabs(x)</code>	Модуль числа
<code>math.copysign(x, y)</code>	Модуль x со знаком числа y

Сравнение вещественных чисел

Функция `isclose`

```
math.isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
```

используется для проверки на равенство с заданной точностью двух вещественных чисел.

- Результат работы функции `True` или `False`
- Если относительная или абсолютная погрешность чисел меньше заданной величины, то числа считаются равными, т.е. результат работы функции `True`, если выполняется неравенство:

$$|a - b| \leq \max(\text{rel_tol} \cdot \max(|a|, |b|), \text{abs_tol})$$

Возведение в степень

<code>math.exp(x)</code>	e^x
<code>math.log(x)</code>	$\ln x$
<code>math.log(x, n)</code>	$\lg_n x$
<code>math.log10(x)</code>	$\lg_{10} x$
<code>math.log2(x)</code>	$\lg_2 x$
<code>math.pow(x, y)</code>	x^y
<code>math.sqrt(x)</code>	\sqrt{x}

Тригонометрические функции

<code>math.sin(x)</code>	$= \sin x$
<code>math.cos(x)</code>	$= \cos x$
<code>math.tan(x)</code>	$= \tan x$
<code>math.acos(x)</code>	$= \arccos x$
<code>math.asin(x)</code>	$= \arcsin x$
<code>math.atan(x)</code>	$= \arctan x$
<code>math.atan2(y, x)</code>	$= \arctan(y/x) \in (-\pi \text{ до } \pi]$
<code>math.hypot(x, y)</code>	$= \sqrt{x^2 + y^2}$

Градусы и радианы

$$\begin{aligned}\text{math.degrees}(x) &= x \times 180/\pi \\ \text{math.radians}(x) &= x \times \pi/180\end{aligned}$$

Константы

<code>math.pi</code>	$\pi = 3.141592\dots$
<code>math.e</code>	$e = 2.718281\dots$
<code>math.inf</code>	бесконечность
<code>math.nan</code>	не число

Базовый пакет для численных методов

- определяет типы данных: векторы, матрицы, многомерные массивы
- Функции для эффективной работы с матрицами
- функции линейной алгебры
- генераторы случайных чисел различных типов
- преобразование Фурье

Пример использование модуля numpy

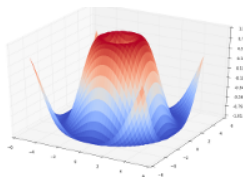
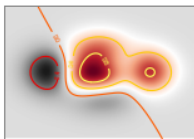
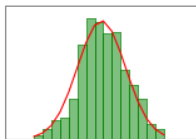
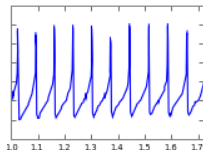
Умножение матриц:

```
1 import numpy as np
2
3 m1 = np.matrix([[1, 2, 3],[5, 5, 6]])
4 m2 = np.matrix([[3, 2],[4, 2],[1, 1]])
5
6 m12 = m1*m2
7
8 print(m12)
```

```
[[14  9]
 [41 26]]
```


Пакет matplotlib

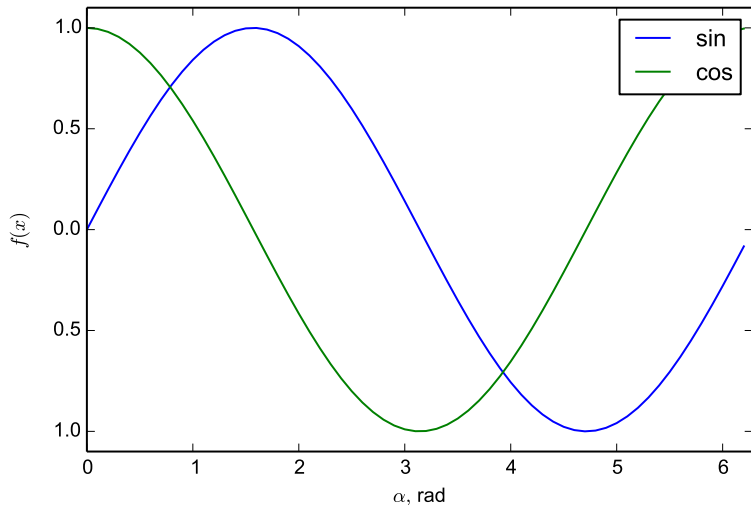
- **matplotlib** — библиотека для построения графиков по массивам (таблицам) данных с возможностью экспорта их в различных форматы файлов
- Вместе с NumPy, SciPy и IPython предоставляет возможности, подобные MATLAB



Пакет matplotlib

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 x = np.arange(0.0, np.pi*2.0, 0.1)
5 y1 = np.sin(x)
6 y2 = np.cos(x)
7
8 plt.plot(x, y1, x, y2)
9
10 plt.axis([0, 2*np.pi, -1.1, 1.1])
11 plt.xlabel('$\\alpha$, rad')
12 plt.ylabel('f(x)')
13 plt.legend(["sin", "cos"])
```

Пакет matplotlib



- `scipy.special`
специальные функции
- `scipy.integration`
численное интегрирование
- `scipy.optimize`
оптимизация, решение нелинейных уравнений
- `scipy.interpolate`
интерполирование
- `scipy.linalg`
линейная алгебра
- `scipy.stats`
статистика

Пример использования модулей scipy

Численное решение нелинейного уравнения:

$$x + 2 \cos x = 0$$

```
1 import numpy as np
2 from scipy.optimize import root
3
4 def func(x):
5     return x + 2 * np.cos(x)
6
7 sol = root(func, 0.3)
8 print(sol.x)
```

```
>> array([-1.02986653])
```

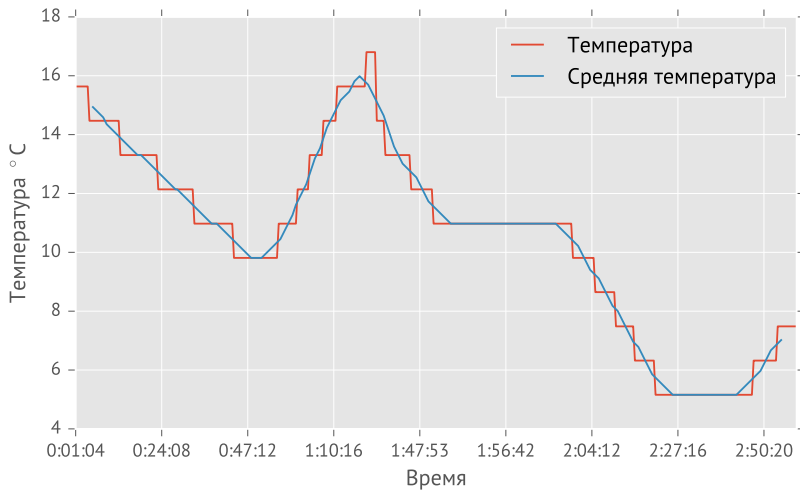
<http://pandas.pydata.org>

Пакет для эффективного анализа данных, первоначально разработанный для анализа финансовой информации (2008).

Возможности **pandas**:

- загрузка данных из текстовых файлов, таблиц XLS, баз данных;
- обработка таблиц и временных рядов, группировка данных, преобразование данных, создание сводных таблиц;
- объединение наборов данных;
- работа с данными большой размерности;
- построение графиков.

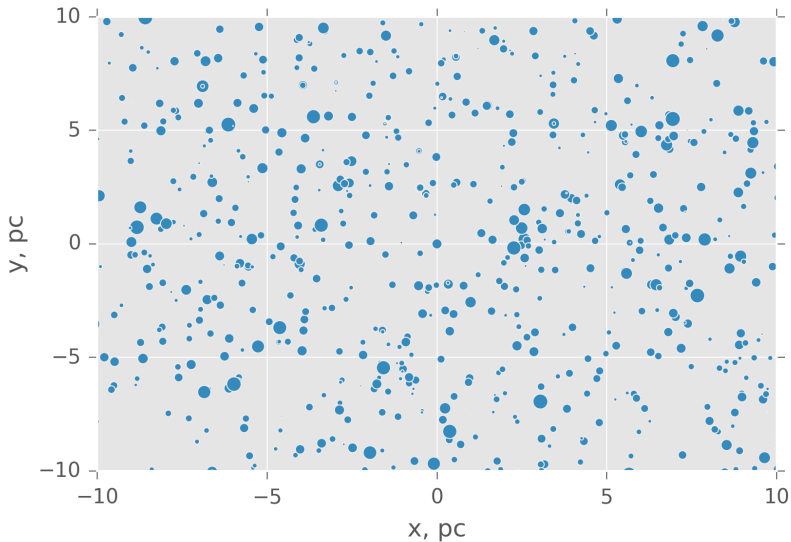
```
1 import pandas as pd
2 import matplotlib
3
4 data = pd.read_csv("temp.dat")
5
6 rolling = pd.rolling_mean(data, 20, center = True)
7 rolling.columns=["Время", "Средняя температура"]
8 ax_data.set_ylabel('Температура  $\circ C$ ')
9
10 ax_data = data.plot()
11 rolling.plot(ax = ax_data)
```



Загрузка данных из Сети

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import matplotlib
4 matplotlib.style.use('ggplot')
5
6 data = pd.read_csv(
7     "http://www.astronexus.com/files/downloads/hygdata_v3.csv.gz",
8     compression="gzip")
9
10 data[data["dist"]<20].plot.scatter(x="x", y="y",
11 s=5*data["absmag"], xlim=(-10,10), ylim=(-10,10))
12
13 plt.xlabel("x, pc")
14 plt.ylabel("y, pc")
15 plt.savefig('stars.png', dpi=300)
```

Карта ближайших звёзд



<http://www.sympy.org>

Пакет для аналитических преобразований:

- Решение уравнений
- Дифференцирование функций
- Интегрирование функций

Решение уравнения

Решение квадратного уравнения

$$x^2 + 10x + 3 = 0$$

```
1 import sympy as sp
2
3 x = sp.symbols('x')
4
5 f = x**2 + 10*x + 3
6
7 print(sp.solve(f))
```

```
>> [-5 - sqrt(22), -5 + sqrt(22)]
```

Дифференцирование функции

Производная функции:

$$f = \cos(x) \sin^2(x)$$

```
1 import sympy as sp
2
3 x = sp.symbols('x')
4
5 f = sp.cos(x)*sp.sin(x)**2
6
7 print(sp.diff(f, x))
```

```
>> -sin(x)**3 + 2*sin(x)*cos(x)**2
```