

# Προγραμματισμός σε C++ & Python & Εφαρμογές στη Ναυπηγική & Ναυτική Μηχανολογία

ΣΝΜΜ 2019

## Μάθημα 5Α: Errors-Exceptions

**Γεώργιος Παπαλάμπρου**

Επίκουρος Καθηγητής ΕΜΠ

`george.papalambrou@lme.ntua.gr`

Εργαστήριο Ναυτικής Μηχανολογίας (Κτίριο Λ)  
Σχολή Ναυπηγών Μηχανολόγων Μηχανικών  
Εθνικό Μετσοβίο Πολυτεχνείο

March 29, 2019

# Περιεχόμενα

## 1 Classes

# Περιεχόμενο Μαθήματος

- Εβδομάδα 1. A. Εισαγωγή. Η γλώσσα. Το περιβάλλον Linux. Command line. Python interpreter. Ιστοσελίδα μαθήματος. Βιβλιογραφία. Editors: Sublime, Spyder. B. Εισαγωγή στην γλώσσα Python. Hello World.
- Εβδομάδα 2. A. Data types. Loops. Control. B. Παραδείγματα
- Εβδομάδα 3. Functions. Modules
- Εβδομάδα 4. OOP. Classes
- Εβδομάδα 5. A. Παραδείγματα: Μέτρηση και επεξεργασία δεδομένων. B. **Errors-Exceptions.**
- Εβδομάδα 6. Εφαρμογή: Hardware. Πλατφόρμες. Πρωτόκολλα. Βασικό I/O Εφαρμογή: Neural Networks. Machine Learning Βιβλιοθήκες NymPy, SciPy. Παραδείγματα: Γραμμική άλγεβρα, Γραφικά

# Εισαγωγή

Για σήμερα η παράδοση προέρχεται από:

- **Core Python Programming, Wesley Chun**  
Chapter 10. Errors-Exceptions

# Errors-Exceptions: - [Βιβλίο: Wesley Chun]

▼ Chapter 10. Errors and Exceptions	397
Section 10.1. What Are Exceptions?	398
Section 10.2. Exceptions in Python	400
Section 10.3. Detecting and Handling Exceptions	403
Section 10.4. Context Management	420
Section 10.5. *Exceptions as Strings	423
Section 10.6. Raising Exceptions	424
Section 10.7. Assertions	427
Section 10.8. Standard Exceptions	429
Section 10.9. *Creating Exceptions	434
Section 10.10. Why Exceptions (Now)?	440
Section 10.11. Why Exceptions at All?	441
Section 10.12. Exceptions and the sys Module	442
Section 10.13. Related Modules	443

# Errors-Exceptions: - <https://docs.python.org>

8. Errors and Exceptions x +

Python Software Foundation (US) 90% Search

Python » English 3.7.3 Documentation » The Python Tutorial » Quick search [Go] | previous | next | more

## Table of Contents

- 8. Errors and Exceptions
  - 8.1. Syntax Errors
  - 8.2. Exceptions
  - 8.3. Handling Exceptions
  - 8.4. Raising Exceptions
  - 8.5. User-defined Exceptions
  - 8.6. Defining Clean-up Actions
  - 8.7. Predefined Clean-up Actions

## Previous topic

7. Input and Output

## Next topic

9. Classes

## This Page

Report a Bug  
Show Source

## 8. Errors and Exceptions

Until now error messages haven't been more than mentioned, but if you have tried out the examples you have probably seen some. There are (at least) two distinguishable kinds of errors: *syntax errors* and *exceptions*.

### 8.1. Syntax Errors

Syntax errors, also known as parsing errors, are perhaps the most common kind of complaint you get while you are still learning Python:

```
>>> while True print('Hello world')
File "<stdin>", line 1
    while True print('Hello world')
                ^
```

```
SyntaxError: invalid syntax
```

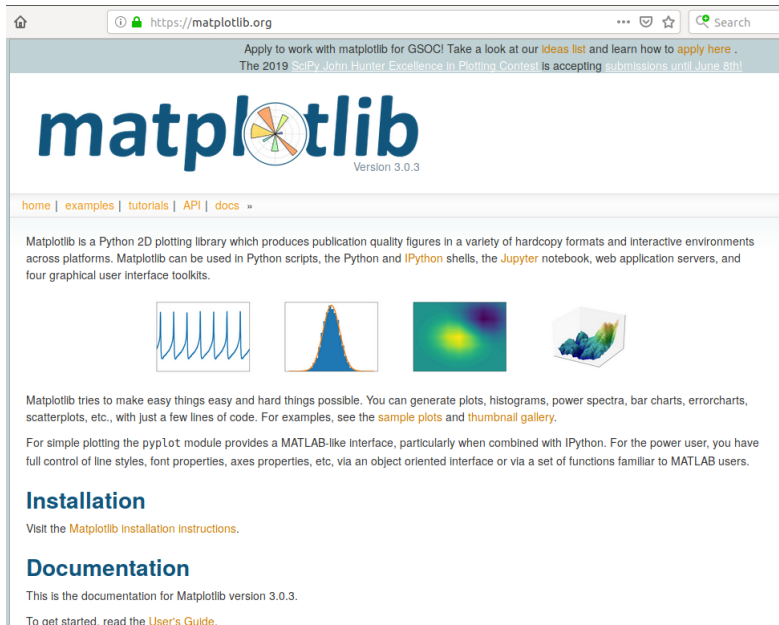
The parser repeats the offending line and displays a little 'arrow' pointing at the earliest point in the line where the error was detected. The error is caused by (or at least detected at) the token *preceding* the arrow: in the example, the error is detected at the function `print()`, since a colon (':') is missing before it. File name and line number are printed so you know where to look in case the input came from a script.

### 8.2. Exceptions

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called *exceptions* and are not unconditionally fatal: you will soon learn how to handle them in Python programs. Most exceptions are not handled by programs, however, and result in error messages as shown here:

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

# Γραφικά: Matplotlib



The screenshot shows the Matplotlib website homepage. At the top, there is a navigation bar with a home icon, a search bar, and a banner for the 2019 SciPy John Hunter Excellence in Plotting Contest. The main header features the Matplotlib logo and the version number 3.0.3. Below the header is a navigation menu with links for home, examples, tutorials, API, and docs. The main content area contains a paragraph describing Matplotlib as a Python 2D plotting library, followed by four small images illustrating different plot types: a line plot, a histogram, a heatmap, and a 3D surface plot. Below these images is a paragraph about the library's ease of use and a link to sample plots. Another paragraph mentions the pyplot module and the IPython interface. The page concludes with sections for Installation and Documentation, each with a link to the respective resources.


Apply to work with matplotlib for GSOC! Take a look at our [ideas list](#) and learn how to [apply here](#) .  
The 2019 SciPy John Hunter Excellence in Plotting Contest is accepting submissions until June 8th!

# matplotlib

Version 3.0.3

[home](#) | [examples](#) | [tutorials](#) | [API](#) | [docs](#) »

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](#) shells, the [Jupyter](#) notebook, web application servers, and four graphical user interface toolkits.



Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the [sample plots](#) and [thumbnail gallery](#).

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with [IPython](#). For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

## Installation

Visit the [Matplotlib Installation Instructions](#).

## Documentation

This is the documentation for Matplotlib version 3.0.3.

To get started, read the [User's Guide](#).

# Μαθηματικά: SciPy, Numpy, ...

https://www.scipy.org/install.html



SciPy.org

## Installing packages

These are general instructions for installing packages in the SciPy ecosystem.

### Scientific Python distributions

For many users, especially on Windows, the easiest way to begin is to download one of these Python distributions, which include all the key packages:

- **Anaconda:** A free distribution of Python with scientific packages. Supports Linux, Windows and Mac.
- **Enthought Canopy:** The free and commercial versions include the core scientific packages. Supports Linux, Windows and Mac.
- **Python(x,y):** A free distribution including scientific packages, based around the **Spyder IDE**. Windows and Ubuntu; Py2 only.
- **WinPython:** Another free distribution including scientific packages and the Spyder IDE. Windows only, but more actively maintained and supports the latest Python 3 versions.
- **Pyzo:** A free distribution based on Anaconda and the IEP interactive development environment. Supports Linux, Windows and Mac.

### Installing via pip

Most major projects upload official packages to the **Python Package index**. They can be installed on most operating systems using Python's standard **pip** package manager.

*Note that you need to have Python and pip already installed on your system.*

You can install packages via commands such as:

```
python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose
```

We recommend using an *user* install, using the `--user` flag to pip (note: do not use `sudo pip`, which can cause problems). This

[About SciPy](#)

[Getting Started](#)

[Documentation](#)

**[Install](#)**

[Bug Reports](#)

[Codes of Conduct](#)

[SciPy Conferences](#)

[Topical Software](#)

[Citing](#)

[Cookbook](#)

[Blogs](#)

[NumFOCUS](#)

CORE PACKAGES:

[Numpy](#)

[SciPy library](#)

[Matplotlib](#)

[IPython](#)

[SymPy](#)

[Pandas](#)