Avrasya Sosyal ve Ekonomi Araştırmaları Dergisi (ASEAD)
Eurasian Journal of Researches in Social and Economics (EJRSE)
ISSN:2148-9963 **www.asead.com**

GRAPHICS CAPABILITIES IN PYTHON

Gulnur ABDİMANAPOVA[1]
Lazzat ZHAİDAKBAYEVA[2]
Sapargali ALDESHOV[3]

ABSTRACT

The article discusses the main features and advantages of graphics in the Python programming language. The article begins with an overview of popular libraries for creating graphs in Python, such as Matplotlib and Plotly, and also compares their features and advantages.

The article goes on to discuss in detail the various types of graphs that can be created in Python, such as linear, columnar, circular, three-dimensional and many others. In addition, the article describes methods for configuring graphs, such as changing coordinate axes, choosing a color scheme, adding labels, and many others.

The article also examines in detail the possibilities of creating interactive graphs using the Plotly library. This can be useful for displaying a large amount of data on a single graph and allows users to scale, rotate and view the graph from different angles.

In general, the article provides a useful overview of the main features of graphics in Python and may be useful for those who are engaged in data analysis, creating dashboards and reports, or are simply interested in data visualization.

Keywords: Graphics, Programming, Library, Module, Graphics Capabilities

PYTHON'DA GRAFİK YETENEKLERİ

ÖZET

Makale, Python programlama dilindeki grafiklerin temel özelliklerini ve faydalarını incelemektedir. Makale, Python'da Matplotlib ve Plotly gibi grafikler oluşturmak için popüler kütüphanelere genel bir bakışla başlar ve bunların özelliklerini ve faydalarını karşılaştırır.

Makalede ayrıca, Python'da doğrusal, sütunlu, dairesel, üç boyutlu ve diğerleri gibi oluşturulabilecek farklı grafik türlerini ayrıntılı olarak ele almaktadır. Ayrıca bu makalede, koordinat eksenlerini değiştirmek, renk şemasını seçmek gibi grafikleri ayarlamanın yolları açıklanmaktadır.

Makale ayrıca Plotly kütüphanesini kullanarak etkileşimli grafikler oluşturma olanaklarına da ayrıntılı olarak bakmaktadır. Bu, tek bir grafikte çok miktarda veriyi görüntülemek için yararlı olabilir ve kullanıcıların grafiği çeşitli açılardan ölçeklendirmelerine, döndürmelerine ve görüntülemelerine olanak tanır.

Genel olarak, bu makale Python'daki grafiklerin temel yeteneklerine genel bir bakış sunmakta ve veri analizi, gösterge tabloları ve raporlar oluşturma veya sadece verileri görselleştirmekle ilgilenenler için yararlıdır.

Anahtar Kelimeler: Grafik, Programlama, Kütüphane, Modül, Grafik Yetenekleri

[1] South Kazakhstan State Pedagogical Un., Faculty of Physics and Mathematics, ORCID ID: 0000-0001-5709-
[2] M. Auezov South Kazakhstan Un., Faculty of Natural Sciences and Pedagogy, ORCID: 0000-0002-3621-6810
[3] South Kazakhstan State Pedagogical Un., Faculty of Physics and Mathematics, ORCID ID: 0000-0001-7735-2299
Araştırma Makalesi/Research Article, Geliş Tarihi/Received: 14/04/2023–Kabul Tarihi/Accepted: 10/05/2023

INTRODUCTION

A person in the modern world is increasingly faced with the need for a graphic representation of one or another aspect of life. Whether it's a mathematical formula and its graph or a nice user interface that is intuitive to everyone. To implement this need, a lot of software has been created, but we will tell you about one of them in more detail.

Python is a high–level general-purpose programming language with dynamic strict typing and automatic memory management, focused on improving developer productivity, code readability and quality, as well as ensuring the portability of programs written in it. Due to the simplicity of the code, it is easy to learn, which is ideal for beginners. Cross-platform, interpretability and a huge number of libraries provide wide application. It is also impossible not to note that Python is an object–oriented language, which is an undoubted plus (Wikipedia).

The graphical capabilities of the programming language are of particular interest to students. And it is the graphics that allows you to clearly demonstrate the work of each block of the program, to creatively approach the task.

In this article, we will learn how to build function graphs in python using the matplotlib module.

The graph of a function is the set of all points in the coordinate plane whose abscissae are equal to the values of the argument, and the ordinates are the corresponding values of the function.

Matplotlib is a library in the Python programming language for visualizing data with two—dimensional and three-dimensional graphics. The resulting images can be used as illustrations in publications (Toby Segaran, 2007).

Matplotlib was written and maintained mainly by John Hunter and is distributed under the terms of a BSD-like license. Images generated in various formats can be used in interactive graphics, scientific publications, graphical user interface, web applications where plotting is required (Sandro Tosi, 2009). In the documentation, the author admits that Matplotlib began with imitation of MATLAB graphics commands, but is an independent project from it.

The matplotlib library is a two-dimensional graphics library for the python programming language with which you can create high-quality drawings in various formats. Matplotlib is a module package for python (Shabanov).

Matplotlib is a flexible, easily configurable package that, together with NumPy, SciPy and IPython, provides MATLAB-like capabilities. The package currently works with several graphics libraries, including wxWindows and PyGTK.

The package supports many types of graphs and charts and is shown in Table 1.

Table 1: Types of Graphs and Charts

S/n	Graphs and charts	Description
1	Line plots	Such a graph is a sequence of data points on a line. Each point consists of a pair of values (x, y), which are transferred to the graph in accordance with the scales of the axes (x and y).
2	Scatter plots	A classical and fundamental type of diagram used to study the relationship between two variables
3	Bar charts and histograms	Bar charts and histograms are different things. The main difference is that the histogram shows the frequency distribution — we set a set of values of the Ox axis, and the frequency is always postponed by Oy. In a bar chart, we set both the values of the abscissa axis and the values of the ordinate axis.
4	Pie charts	This is a visual way to show the proportions of components in a set
5	Stem-leaf charts	This is a chart that displays data by dividing each value in a dataset into a base and a sheet.
6	Contour plots	This is a type of graph that allows us to visualize three-dimensional data in two dimensions using contours
7	Spectrogram	It can be defined as a visual representation of frequencies as a function of time, which shows the signal strength at a certain point in time.

The user can specify coordinate axes, a grid, add inscriptions and explanations, use a logarithmic scale or polar coordinates.

Simple three-dimensional graphs can be built using the mplot3d toolkit. There are other sets of tools: for cartography, for working with Excel, utilities for GTK and others.

With the help of Matplotlib, you can also make animated images.

A set of supported image formats, vector and raster, can be obtained from the FigureCanvasBase dictionary.filetypes (Table 2).

Table 2: Typical Supported Formats

Formats	Description
Encapsulated PostScript	A file format based on a subset of the PostScript language and intended for the exchange of graphic data between various applications
Enhanced Metafile	Vector, graphical format of data representation in Windows OS, adapted to work with numerous image editors
JPEG	One of the popular raster graphic formats used for storing photos and similar images
PDF	Cross-platform open format of electronic documents
PNG	A raster format for storing graphic information using lossless compression using the Deflate algorithm.
RGBA ("raw" format)	This is a three-channel RGB color model supplemented with a fourth alpha channel
SVG	Vector format describing images in the form of shapes, lines, text and filters
SVGZ	The compressed file format of scalable vector graphics is a vector image SVG (Scalable Vector Graphics). The letter "z" in the extension name means "archived" and indicates that the file is compressed.
TIFF	Format for storing raster graphic images

In addition, other modules can be created based on the package classes. For example, to generate sparkgraphs.

Plotly is an open source Python module that is used for data visualization and supports various graphs, such as line charts, scatter charts, histograms, area charts, etc. In this article we will see how to build a basic chart using plotly, as well as how to make the graph interactive.

Plotly uses javascript and is used to create interactive graphs where we can zoom in on the graph or add additional information such as mouse hover data, and much more (Shabanov).

List of features that make Plotly one of the best JavaScript graph libraries:

• Using Plotly.js can easily create interactive charts. Any chart you create in the library is equipped with functions such as zooming, panning, autoscaling, etc. These functions are very useful when you want to study diagrams with a large number of plotted points. All these events are displayed in the API, so you can write your own code to perform your own actions when any of these events are triggered.

• High performance when plotting a large number of points makes Plotly.js is a great choice when you need to draw a large amount of data. Since most charts are created using SVG, you get decent compatibility between browsers and the ability to export high-quality images of any chart. However, involving a large number of SVG elements in the DOM can negatively affect performance. The library uses stack.gl to create high-performance 2D and 3D diagrams.

• All 3D diagrams created using WebGL will take full advantage of all the features offered by the GPU.

• All charts are Plotly.js is fully configurable. Everything from colors and labels to grid lines and legends can be customized using a set of JSON attributes.

METHODS AND RESULTS

In this study, we will analyze how to build graphs of functions using the Matplotlib module in Python Python.

Matplotlib is a Python library designed for data visualization. In this article analyzes the construction of graphs of functions in Python on a plane and the construction of a surface in three-dimensional space. Often, it is Matplotlib that is used in scientific research and conferences to demonstrate the data obtained. To plot graphs, you need to import the Pyplot module. Pyplot is a module for working with graphs in Python. Pyplot is a set of commands created for plotting functions and equations. The main element of the image that pyplot builds is a Figure, one or more fields with graphs, coordinate axes, text labels, etc. are superimposed on it. The plot() function is used to plot the graph (Abdrakhmanov, 2018). For convenient plotting, you also need to use the NumPy library.

Matplotlib, like NumPy, is built into the Spyder development environment, so they can be imported without pre-installation.

```
import numpy as np
import matplotlib.pyplot as plt
```

as np and as plt means that when we call functions and procedures from modules, we will use np and plt instead of module names.

To plot a function graph in Python, you need to specify the function itself. It can be set using a lambda function. A lambda function is a short way to write a regular function in one line. In this lesson, we will look at the construction of a sine wave on Python. The sine wave is given by the function Th

f(x) = sin(x).

y = lambda x: np.sin(x)

y is the function designation (we will use y(x) to call it), lambda is the keyword indicating the beginning of the lambda function assignment, x is the argument used in the function, after the colon the function is set. Since there is no function in standard Python that returns the sine of x, we set it using NumPy, we imported it under the name np.

All actions in Pyplot are performed in figures. To plot a function in Python, you first need to set a grid of coordinates (Lemeshevsky, 2020). The coordinate grid in python is set using the plot.subplots() command.

fig = plt.subplots()

We need to define the range of values on which we will plot the function in Python. This is done using linspace.

x = np.linspace(-3, 3, 100)

linspace creates an array with a lower bound of -3 and an upper bound of 3, there will be 100 elements in the created array. The larger the last number, the more function values will be calculated, the more accurately the graph will be displayed in Python.

After we have created a coordinate system, a plotting area, we can plot a graph in Python. To plot a function in Python, you need to use the plt.plot(x, y(x)) command, where x is an argument, y(x) is a function from x, specified using a lambda expression (Ozerova, 2022).

plt.plot(x, y(x))

After we have built a graph in Python, we need to show it in the figure. To do this, use plt.show().

Full python program code for drawing a graph of a function

```
# import modules
import numpy as np
import matplotlib.pyplot as plt
# function
y = lambda x: np.sin(x)
# creating a drawing with a coordinate plane
fig = plt.subplots()
# creating an area in which there will be
# - graph
x = np.linspace(-3, 3,100) is displayed
# x values to be displayed
# number of elements in the created array
# - the quality of drawing the graph
```

```
# draw a graph
plt.plot(x, y(x))
# showing the graph
plt.show()
```

Figure 1 shows a graph of a sine wave in python in a separate window.

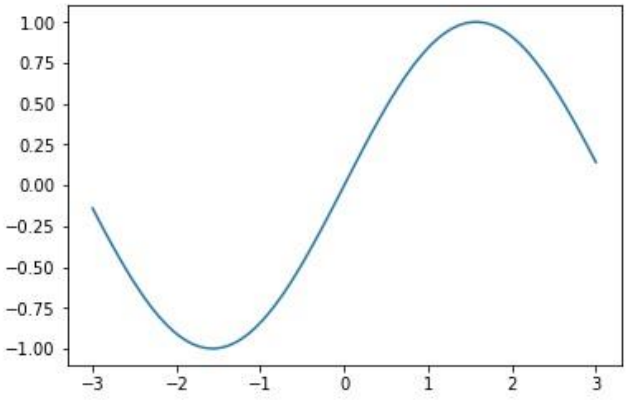


Figure 1: The Graph of The Sine Wave

Displaying multiple graphs in a single figure in Python

In one area of python, graphs of several functions are displayed. To do this, add the function y= x and draw it together with the sine wave (Figure 2).

To do this, we introduce another function using lambda

```
y1=lambda x: x
```

Plot this function plt.plot(x,y1(x))

As a result, a Python program for plotting two functions in one window

```
# import modules
import numpy as np
import matplotlib.pyplot as plt
# function
y = lambda x: np.sin(x)
y1=lambda x: x
# creating a drawing with a coordinate plane
fig = plt.subplots()
# creating an area in which there will be
# - graph
x = np.linspace(-3, 3,100) is displayed
# x values to be displayed
# number of elements in the created array
# - the quality of drawing the graph
# draw a graph
plt.plot(x, y(x))
```

```
plt.plot(x,y1(x))
# show a graph
plt.show()
```

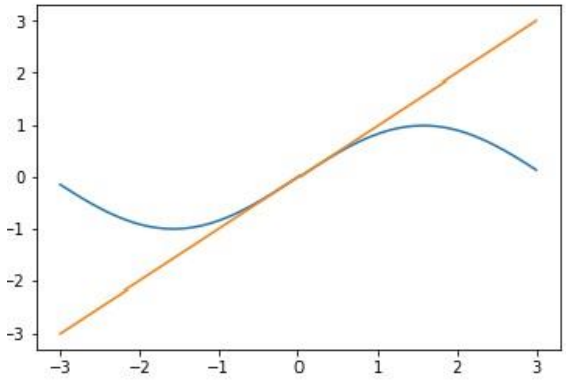


Figure 2: Graph of The Function y= x with a Sine Wave

In three-dimensional space, each point is defined by three coordinates, therefore, in three-dimensional space, two arguments are needed to set the function (Jake VanderPlas, 2017). In this Python tutorial, we will set the function

f(x,y)=x^2-y^2

from two arguments.

Arguments x and y, function z.

f = lambda x, y: x ** 2 - y ** 2

To start drawing three–dimensional surfaces in Python, you first need to set the plot area using the plt function.figure takes the figsize(x, y) parameter, where x and y are the width and height of the drawing in inches. Let's create a drawing in Python with a size of 12×6 inches to display graphs

fig = plt.figure(figsize = (12, 6))

In the constructed area, we will create a drawing in which a three-dimensional space with coordinate axes and the surface itself will be displayed.

Python uses fig.add_subplot() for this.

ax = fig.add_subplot(1, 1, 1, projection = '3d')

The function in Python fig.add_subplot() splits the plot area into cells and sets in which cell to draw a three-dimensional graph. So the command

ax = fig.add_subplot(1, 1, 1, projection = '3d')

splits the plot area into two cells and a three-dimensional garfic will be displayed in the first cell, thanks to the projection = ‘3d’ argument.

Let's introduce the function display areas for each argument in Python.

xval = np.linspace(-5, 5, 100)

yval = np.linspace(-5, 5, 100)

You need to create a surface that will be displayed in the drawing in Python. To do this,

surf = ax.plot_surface(x, y, z, rstride = 4, cstride = 4, cmap = cm.plasma)

Where x and y are the accepted arguments, z is the received function, rstride and cstride are responsible for the step of drawing the surface in Python, the smaller these values are, the smoother the gradient on the surface will look. Using cmap.plasma, the surface will be displayed with the plasma color scheme (Dawson, 2012). For example, there are color schemes such as viridis and magma.

Example of a Python program building a surface in three-dimensional space

```
# importing modules
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from matplotlib import cm
import matplotlib.pyplot as plt
# surface equation
f = lambda x, y: x ** 2 - y ** 2
# creating a canvas for a drawing
fig = plt.figure(figsize = (10, 10))
# creating a drawing of a space with a surface
ax =fig.add_subplot(1, 1, 1, projection = '3d')
# marking the axis boundaries for arguments
xval = np.linspace(-4, 4, 100)
yval = np.linspace(-4, 4, 100)
# creating an array with xval columns and yval rows
# - z values will be stored in this array
x, y = np.meshgrid(xval, yval)
# equating z to a function of x and y
z = f(x, y)
# creating a surface
surf = ax.plot_surface(
# note the arguments and the equation of the surface
x, y, z,
# grid drawing step
# - the smaller the value, the smoother
# - there will be a gradient on the surface
rstride = 10,
cstride = 10,
# color scheme plasma
cmap = cm.plasma)
```

Figure 3 shows a graph of a three-dimensional surface in a color scheme in a special window.

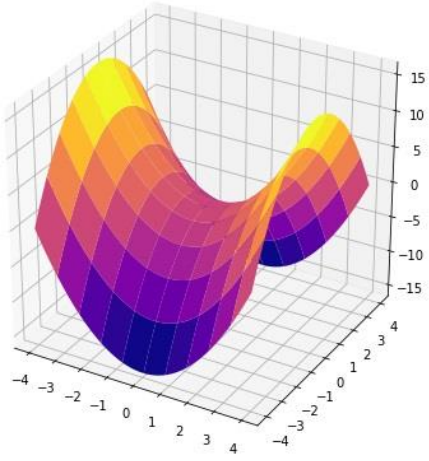


Figure 3: Graph of a Three-Dimensional Surface in a Color Scheme

Let's change the parameters for constructing a three-dimensional surface, reduce the size of the septic tank, make the surface smoother and more accurate for this, reduce the parameters and change the color scheme to viridis

stride = 2,

cstrike = 2,

cmap = cm.viridis)

The graph of a three-dimensional surface in Python is more accurate and in a different color scheme is shown in Figure 4.

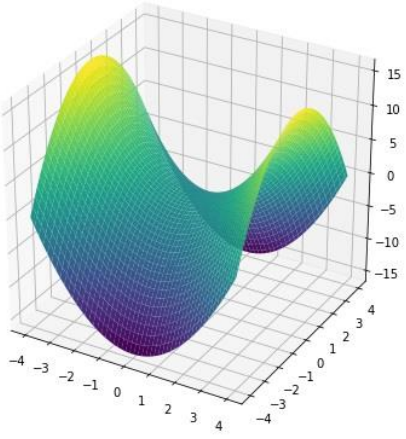


Figure 4. Graph of a three-dimensional surface in a different color scheme

Plotly can also be successfully used in jupyter notebooks.

Before starting work, we import all the necessary modules and initialize plotly using the init_notebook_mode command.

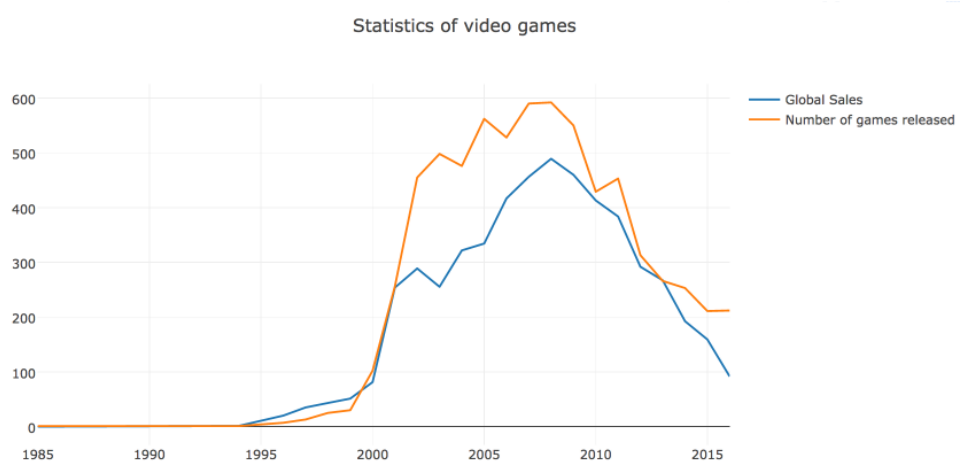
```
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly
import plotly.graph_objs as go
init_notebook_mode(connected=True)
```

To begin with, let's build a line plot with the dynamics of the number of released games and their sales by year.

```
# let's count the number of games released and copies sold by year
years_df = df.groupby('Year_of_Release')[['Global_Sales']].sum().join(
df.groupby('Year_of_Release')[['Name']].count()
)
years_df.columns = ['Global_Sales', 'Number_of_Games']
# creating a line for the number of copies sold
trace0 = go.Scatter(
x=years_df.index,
y=years_df.Global_Sales,
name='Global Sales'
)
# creating a line for the number of games released
trace1 = go.Scatter(
x=years_df.index,
y=years_df.Number_of_Games,
name='Number of games released'
)
# define the data array and set the title of the graph in layout
data = [trace0, trace1]
layout = {'title': 'Statistics of video games'}
# creating a Figure object and visualizing it
fig = go.Figure(data=data, layout=layout)
iplot(fig, show_link=False)
```

Plotly builds a visualization of the Figure object, which consists of data (an array of lines called traces in the library) and the layout/style for which the layout object is responsible. In simple cases, call the iplot function from the traces array.

The show_link parameter is responsible for links to the online platform plot.ly on the charts (Suzi, 2006). Since this functionality is not usually required, but it is possible to hide it to prevent accidental clicks.

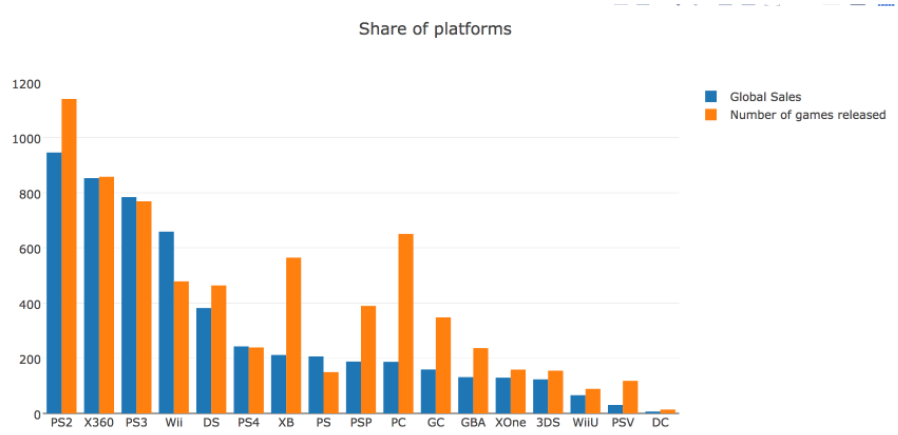


Save the graph as an html file.

```
plotly.offline.plot(fig, filename='years_stats.html ', show_link=False)
```

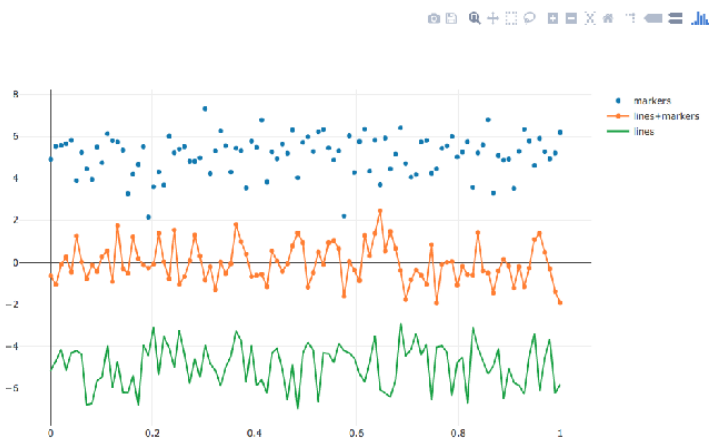
Let's also look at the market share of gaming platforms, calculated by the number of games released and by total revenue. To do this, we will build a bar chart.

```
# counting the number of games sold and released by platforms
platforms_df = df.groupby('Platform')[['Global_Sales']].sum().join(
df.groupby('Platform')[['Name']].count()
)
platforms_df.columns = ['Global_Sales', 'Number_of_Games']
platforms_df.sort_values('Global_Sales', ascending=False, inplace=True)
# creating traces for visualization
trace0 = go.Bar(
x=platforms_df.index,
y=platforms_df.Global_Sales,
name='Global Sales'
)
trace1 = go.Bar(
x=platforms_df.index,
y=platforms_df.Number_of_Games,
name='Number of games released'
)
# create an array with data and set the title for the graph and the x axis in layout
data = [trace0, trace1]
layout = {'title': 'Share of platforms', 'xaxis': {'title': 'platform'}}
# creating a Figure object and visualizing it
fig = go.Figure(data=data, layout=layout)
iplot(fig, show_link=False)
```



We can create even more complex and informative graphs, such as a linear scatter plot, in a similar way as above:

```
import plotly
import plotly.graph_objs as go
# Create random data with numpy
import numpy as np
N = 100
random_x = np.linspace(0, 1, N)
random_y0 = np.random.randn(N)+5
random_y1 = np.random.randn(N)
random_y2 = np.random.randn(N)-5
# Create traces
trace0 = go.Scatter(
x = random_x,
y = random_y0,
mode = 'markers',
name = 'markers' )
trace1 = go.Scatter(
x = random_x,
y = random_y1,
mode = 'lines+markers',
name = 'lines+markers' )
trace2 = go.Scatter(
x = random_x,
y = random_y2,
mode = 'lines',
name = 'lines' )
data = [trace0, trace1, trace2]
plotly.offline.plot(data, filename='scatter-mode')
```



CONCLUSION

The article provides an overview of various libraries and tools that can be used to visualize data in Python.

The choice of a specific library depends on the specific task and preferences of the programmer. They provide an overview of the following libraries: Matplotlib, Plotly.

Each of these libraries has its own features and advantages. Matplotlib, for example, is one of the most popular libraries for data visualization in Python and has a wide range of possibilities for creating various types of graphs. Plotly allows you to create interactive graphs.

The article also provides code examples that help how to use each library. This can be useful for beginners who are just starting to learn graphics in Python.

In general, the article gives an excellent overview of the basics of the Python graphics module and can serve as a starting material for anyone who wants to start creating professional graphics in Python.

REFERENCES

Wikipedia. Python: Free Encyclopedia. Electronic data. [electronic resource]. URL: https://ru.wikipedia.org/wiki/Python.

Toby Segaran. (2007) Programming Collective Intelligence: Building Smart Web 2.0 Applications. — O'Reilly Media, Inc., 308 p. ISBN: 9780596529321

Sandro Tosi. (2009) Matplotlib for Python Developers. Packt Publishing. 308 p. ISBN: 978-1847197900

Shabanov P.A. Scientific graphics in python [Electronic resource]. URL: https://github.com/whitehorn/Scientific_graphics_in_python

Abdrakhmanov M.I. (2018) Python. Data visualization: Matplotlib, Seaborn, Mayavi. Competition for research papers: technological innovations and scientific discoveries international research competition. SIC Bulletin of science. 59. [Electronic resource]. URL: https://devpractice.ru/book-pythondata-viz

Lemeshevsky S.V. (2020) Graphical visualization of data. Institute of Mathematics of the National Academy of Sciences of Belarus. 25 p. https://slemeshevsky.github.io/python-course/visual/pdf/visual.pdf

Ozerova G.P. (2022) Fundamentals of Python programming in examples and tasks. Vladivostok: FEFU Publishing House. 128 p.
Jake VanderPlas. (2017) Python Data Science Handbook. Published by O'Reilly Media. 548 p.
Dawson M. (2012) Programming in Python. - St. Petersburg: Peter. p. 79-83
Suzi R.A. (2006) Python programming language: Textbook. M.: INTUIT, BINOM. Laboratory of Knowledge. 206 p.