Самостоятельная работа №1

Повторим и выполним самостоятельно задания из лекции по теме 1.4

Визуализация с помощью библиотеки Matplotlib

1) Импортируем библиотеки pandas, numpy, matplotlib и модуль matplotlib.pyplot. Для вывода рисунков на экран воспользуемся встроенной командой `%matplotlib inline`. Выполнить следующий код:

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

1. Создадим простые графики

2) Для создания рисунка используем оболочку pyplot.subplots(). Затем – Axes.plot() для рисования графика:

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
```

3) Предыдущий код можно записать короче:

```
plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
```

4) Создать рисунок (figure) также позволяет метод plt.figure()
Функция `scatter` просто рисует точки, не соединяя их линиями.
Для отражения рисунка на экране, можно воспользоваться командой plt.show():

```
# Создать рисунок
plt.figure()
# Нарисовать отдельные точки с координатами (x, y) с помощью фун
кции scatter
plt.scatter([0, 0.25, 1], [0, 1, 0.5])
# Отобразить рисунок на экране
plt.show()
```

5) Постройте несколько множеств точек на одном графике

```
plt.scatter([0, 1, 2, 3, 4 , 5], [0, 1, 2, 3, 4 , 5])
plt.scatter([1, 2, 3, 1, 2 , 1], [2, 3, 4, 3, 4 , 4])
plt.scatter([2, 3, 4, 3, 4 , 4], [1, 2, 3, 1, 2 , 1])
plt.show()
```

6) Постройте замкнутый многоугольник:

```
plt.figure()
plt.plot([0, 0.25, 1, 0], [0, 1, 0.5, 0])
plt.show()
```

2. Использование массивов библиотеки NumPy

7) Поработайте с функцией `linspace()` библиотеки NumPy (np), которая возвращает (задает) одномерный массив из указанного количества элементов, значения которых равномерно распределены внутри заданного интервала.

7.1. По умолчанию количество элементов последовательности выводится 50.

```
# Создаем массив координат – по умолчанию 50 точек,
# равнораспределенных в диапазоне от 0 до 1
np.linspace(0, 1)
```

7.2. Выведем количество элементов в интервале, равное 5.

```
np.linspace(0, 1, num = 5)
```

7.3. Зададим парметр endpoint = *False*. В этом случае значение stop не входит в интервал. По умолчанию endpoint = True.

```
np.linspace(0, 1, num = 5, endpoint = False)
```

7.4. Если параметр retstep = True, то возвращается значение шага между элементами

```
np.linspace(0, 1, num = 5, endpoint = False, retstep = True)
```

8) Постройте ломаную с использованием фунукции linspace(), когда много точек

```
x = np.linspace(0, 4*np.pi, 100)
plt.figure()
plt.plot(x, np.sin(x))
plt.show()
```

9) Постройте параметрическую линию $x=x(t)$, $y=y(t)$

```
t = np.linspace(0, 2*np.pi, 100)
plt.figure(figsize=(4, 4))
plt.plot(np.cos(t), np.sin(t))
plt.show()
```

10) Постройте фигуры Лиссажу:

10.1. a=2, в=3

```
plt.figure(figsize=(4, 4))
plt.plot(np.sin(2*t), np.cos(3*t))
plt.show()
```

10.2. a=8, в=9

```
plt.figure(figsize=(4, 4))
plt.plot(np.sin(8*t), np.cos(9*t))
plt.show()
```

Замечание: `t = np.linspace(0, 2*np.pi, 100)` сохраняется из примера 9).

==========

3. Несколько кривых на одном графике

11. Постройте две кривые на одном графике:

а) линиями

```
x = np.linspace(0, 4*np.pi, 100)
plt.figure()
plt.plot(x, np.sin(x), 'r-')
plt.plot(x, np.cos(x), 'b--')
plt.show()
```

б) кружками и квадратиками

```
x = np.linspace(0, 1, 11)
plt.figure()
plt.plot(x, x**2, 'ro')
plt.plot(x, 1 - x, 'gs')
plt.show()
```

4. Настройка графиков

12. Подписи к рисунку, осям и графикам

а) Скопируйте код в блокнот и выполните:

```
x = np.linspace(0, 2*np.pi, 100)

plt.figure(figsize=(10, 5))
plt.plot(x, np.sin(x), linewidth=2, color='g', dashes=[8, 4], la
bel=r'$\sin x$')
plt.plot(x, np.cos(x), linewidth=2, color='r', dashes=[8, 4, 2,
4], label=r'$\cos x$')
plt.axis([0, 2*np.pi, -1, 1])
plt.xticks(np.linspace(0, 2*np.pi, 9),   # Где сделать отметки
           ('0',r'$\frac{1}{4}\pi$',r'$\frac{1}{2}\pi$',# Как подписать
            r'$\frac{3}{4}\pi$',r'$\pi$',r'$\frac{5}{4}\pi$',
            r'$\frac{3}{2}\pi$',r'$\frac{7}{4}\pi$',r'$2\pi$'),
           fontsize=20)
plt.yticks(fontsize=12)
plt.xlabel(r'$x$', fontsize=20)
plt.ylabel(r'$y$', fontsize=20)
plt.title(r'$\sin x$, $\cos x$', fontsize=20)
plt.legend(fontsize=20, loc=0)
plt.show()
```

б) Постройте пунктирный график функции $y=x^3$. Подпишите оси, заголовок и создайте легенду. Нанести сетку.

```
x = np.linspace(-2, 2, 100)

plt.figure(figsize=(8, 8))
plt.plot(x, x**3, linestyle='--', lw=2, label='$y=x^3$')
plt.xlabel('x'), plt.ylabel('y')
plt.legend()
plt.title('График кубической функции')
plt.grid(ls=':')
plt.show()
```

в) Постройте точечные графики, используя маркеры разных типов:

```
x = np.linspace(0, 1, 11)

plt.figure(figsize=(4, 4))
plt.plot(x, x, linestyle='', marker='<', markersize=20, markerfa
cecolor='#FF0000')
plt.plot(x, x**2, linestyle='', marker='^', markersize=10, marke
rfacecolor='#00FF00')
plt.plot(x, x**(1/2), linestyle='', marker='v', markersize=10, m
arkerfacecolor='#0000FF')
plt.plot(x, 1 - x, linestyle='', marker='+', markersize=10, mark
erfacecolor='#0F0F00')
plt.plot(x, 1 - x**2, linestyle='', marker='x', markersize=10, m
arkerfacecolor='#0F000F')
plt.axis([-0.05, 1.05, -0.05, 1.05])
plt.show()
```

13. Использование логарифмического масштаба осей:
а) по оси *y*

```
x = np.linspace(-5, 5, 100)

plt.figure()
plt.plot(x, np.exp(x) + np.exp(-x))
plt.yscale('log')
plt.yticks(fontsize=15)
plt.show()
```

б) по обеим осям

```
x = np.logspace(-2, 2, 100)
plt.figure()
plt.plot(x, x + x**3)
plt.xscale('log'), plt.xticks(fontsize=15)
plt.yscale('log'), plt.yticks(fontsize=15)
plt.show()
```

5. Сложные графики

14. а) спираль

```
t = np.linspace(0, 4*np.pi, 100)
plt.figure()
plt.polar(t, t)
plt.show()
```

б) Угловое распределение пионов

```
phi = np.linspace(0, 2*np.pi, 100)
plt.figure()
plt.polar(phi, np.sin(phi)**2)
plt.show()
```

15. Контурные графики

Построить линии уровня функции.

а)

```
x = np.linspace(-1, 1, 50)
y = x
z = np.outer(x, y)
plt.figure(figsize=(5,5))
plt.contour(x, y, z)
plt.show()
```

Комментарий: Функция numpy.outer(a, b, out=None) вычисляет произведение двух векторов: a, b – числа (массивы NumPy); out – *массив* NumPy*, необязательный параметр.*

б) Увеличим количество линий уровней и подпишем их

```
plt.figure(figsize=(5,5))
curves = plt.contour(x, y, z, np.linspace(-1, 1, 11))
plt.clabel(curves)
plt.title(r'$z=xy$', fontsize=20)
plt.show()
```

в) заменим функцию `contour` на `contour`

```
plt.figure(figsize=(5, 5))
plt.contourf(x, y, z, np.linspace(-1, 1, 11))
plt.show()
```

г) Зададим собственную шкалу с помощью функции `colorbar()`.

```
plt.figure(figsize=(5, 5))
plt.contourf(x, y, z, np.linspace(-1, 1, 11))
plt.colorbar()
plt.show()
```

16. Создание пиксельных картинок

а)

```
n = 256
u = np.linspace(0, 1, n)
x, y = np.meshgrid(u, u)
z = np.zeros((n, n, 3))
z[:, :, 0] = x
z[:, :, 2] = y

plt.figure()
plt.imshow(z)
plt.show()
```

б) Отобразить с помощью функции `np.meshgrid` прямоугольную сетку в диапазоне значений x и y

```
n = 28
u = np.linspace(0, 1, n)
x, y = np.meshgrid(u, u)
plt.plot(x, y, marker='.', color='r', linestyle='none')
plt.show()
```

б) Увеличим n = 256 и цвет сделаем зеленым ('g')

```
n = 256
u = np.linspace(0, 1, n)
x, y = np.meshgrid(u, u)
plt.plot(x, y, marker='.', color='g', linestyle='none')
plt.show()
```

в) Постройте график трехмерной функции $z = f(x, y)$, на котором каждому значению z соответствует определенный цвет

```
x, y = np.mgrid[-5*np.pi:5*np.pi:1000j,
                -5*np.pi:5*np.pi:1000j]
z = np.sin(x) + np.cos(y)
fig, ax = plt.subplots()
ax.imshow(z)
fig.set_figwidth(8)    #  ширина и
fig.set_figheight(8)   #  высота "Figure"
plt.show()
```

6. Трехмерная графика

17. а) Скопируйте код в блокнот:

```
t = np.linspace(0, 4*np.pi, 100)
x = np.cos(t)
y = np.sin(t)
z = t/(4*np.pi)
fig = plt.figure()
from mpl_toolkits.mplot3d import Axes3D
```

```
ax = Axes3D(fig)
ax.plot(x, y, z)
plt.show()
```

б) Зададим ракурс, с которого смотреть на картинку

```
fig = plt.figure()
ax = Axes3D(fig)
ax.elev, ax.azim = 45, 30
ax.plot(x, y, z)
plt.show()
```

18. Поверхности

а)

```
X = 10
N = 50
u = np.linspace(-X, X, N)
x, y = np.meshgrid(u, u)
r = np.sqrt(x**2 + y**2)
z = np.sin(r)/r

fig = plt.figure()
ax = Axes3D(fig)
ax.plot_surface(x, y, z, rstride=10, cstride=10)
plt.show()
```

б) Раскрасить поверхность. Воспользуемся встроенным способом раскраски поверхностей – методом `gnuplot`

```
fig = plt.figure()
ax = Axes3D(fig)
ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap='gnuplot')
plt.show()
```

в) Постройте бублик

```
t = np.linspace(0, 2*np.pi, 50)
th, ph = np.meshgrid(t, t)
r = 0.2
x, y, z = (1 + r*np.cos(ph))*np.cos(th), (1 + r*np.cos(ph))*np.s
in(th), r*np.sin(ph)

fig = plt.figure()
ax = Axes3D(fig)
ax.elev = 60
ax.plot_surface(x, y, z, rstride=2, cstride=1)
plt.show()
```

https://matplotlib.org/stable/gallery/index.html – эта галерея содержит примеры многих действий, которые вы можете делать с Matplotlib. Щелкните любое изображение, чтобы увидеть полное изображение и исходный код.

Галерея демонстрирует разнообразие способов создания фигур. Просмотрите галерею, щелкните любой рисунок, на котором есть фрагменты того, что вы хотите увидеть, и код, который его сгенерировал. Вскоре вы, как шеф-повар, будете смешивать и сочетать компоненты для создания своего шедевра!