08 Exercises

September 13, 2020

Import the NumPy and Matplotlib modules:

[0]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

0.1 Exercise 08.1 (function plotting)

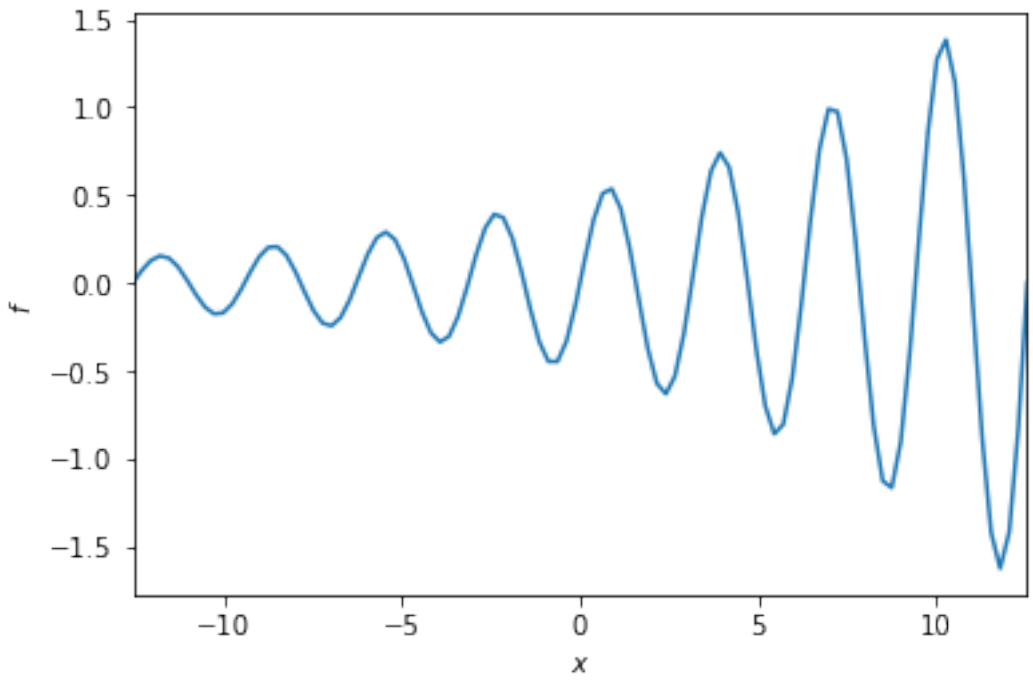
Consider the function

$$f(x) = e^{x/10}\sin(\omega_1 x)\cos(\omega_0 x)$$

from $x = -4\pi$ to $x = 4\pi$.

(1) Plot the function when $\omega_0 = \omega_1 = 1$. Label the axes.

[8]:
```
num_points = 100
x = np.linspace(-4*np.pi, 4*np.pi, num=num_points)

f = np.exp(x/10)*np.sin(x)*np.cos(x)
# Plot the graph
plt.plot(x, f)

# Label the axis
plt.xlabel('$x$')
plt.ylabel('$f$')

# Use the start and end values in x as x limits (recall that x[-1]
# returns the last value in x)
plt.xlim(x[0], x[-1])

plt.show()
```



(2) Create an interactive plot with sliders for ω_0 and ω_1, varying from 0 to 2.

[49]:
```
from ipywidgets import *

@interact( 0 = (0, 2, 0.1),  1 = (0, 2, 0.1))
def plot( 0=1,  1=1):

    num_points = 100
    x = np.linspace(-4*np.pi, 4*np.pi, num=num_points)
    f = np.exp(x/10)*np.sin( 1*x)*np.cos( 0*x)

    plt.xlabel('$x$')
    plt.ylabel('$f$')
    plt.title("$\omega0$ = {}, $\omega1$ = {}".format( 0,  1))
    plt.plot(x, f)
    plt.show()
```

```
  File "<ipython-input-49-af5ea55ddb13>", line 1
    from ipywidgets import
                          ^
SyntaxError: invalid syntax
```

0.2 Exercise 08.2 (multiple function plotting)

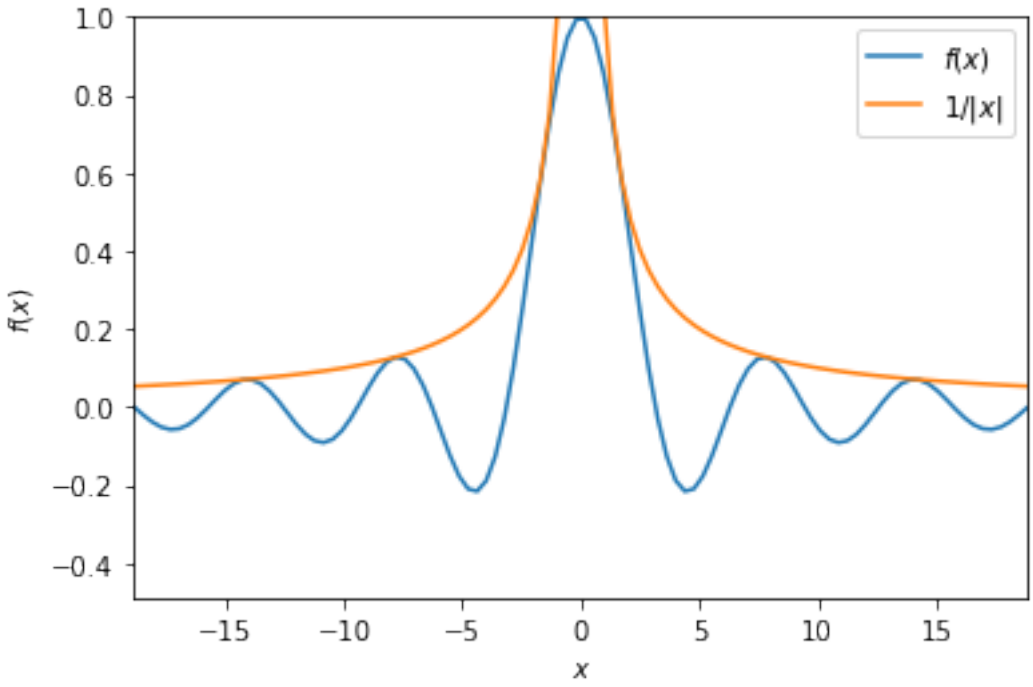
Plot the function

$$f(x) = \frac{\sin(x)}{x}$$

from $x = -6\pi$ to $x = 6\pi$. Think carefully about which x values you use when x is close to zero.

Add to the above plot the graph of $1/|x|$, and limit the range of the y axis to 1 using `plt.ylim`. (Hint: use `np.abs(x)` to return the absolute values of each component of a NumPy array `x`.)

[10]:
```
num_points = 100
x = np.linspace(-6*np.pi, 6*np.pi, num=num_points)

f = np.sin(x)/x
# Plot the graph
plt.plot(x, f, label='$f(x)$')
plt.plot(x, 1/np.abs(x), label="$1/|x|$")
plt.ylim(top=1)

# Label the axis
plt.xlabel('$x$')
plt.ylabel('$f(x)$')

# Add legend
plt.legend()

# Use the start and end values in x as x limits (recall that x[-1]
# returns the last value in x)
plt.xlim(x[0], x[-1])

plt.show()
```



0.3 Exercise 08.3 (demographics and interactive plotting)

A county planning body has requested an interactive tool to visualise the population distribution in Cambridgeshire (by district) from 2011 to 2021 for different population growth rate scenarios in each district. It can be assumed that:

- the growth rates are constant in each district;
- the growth rate will not be negative in any district; and
- the annual growth rate in any one district will not exceed 10%.

Building on the pie chart example with population data in the Activity notebook, create an interactive plot with:

1. A slider for the year (from 2011 to 2021); and
2. Sliders for the annual population growth for each district (in percentage), with an initial value of zero for each district.

0.3.1 Solution

There are number of ways this exercise could be done, from very simple but not very robust with respect to changes in the order of the input data, to quite techical but robust with repsect to the order in which data is entered.

[11]:
```
population_data = (('Cambridge City', 123900),
                   ('East Cambridgeshire', 83800),
                   ('Fenland', 95300),
                   ('Huntingdonshire', 169500),
```

```
                    ('South Cambridgeshire', 148800))
data=[]
label=[]
for i in range(len(population_data)):
    data.append(population_data[i][1])
    label.append(population_data[i][0])
print(data)
print(label)
```

```
[123900, 83800, 95300, 169500, 148800]
['Cambridge City', 'East Cambridgeshire', 'Fenland', 'Huntingdonshire', 'South
Cambridgeshire']
```

[12]:
```
@interact(year = (2011, 2021, 1),
          G_C = (0, 10, 0.1),
          G_E = (0, 10, 0.1),
          G_F = (0, 10, 0.1),
          G_H = (0, 10, 0.1),
          G_S = (0, 10, 0.1))
def plot(year=2011, G_C=0, G_E=0, G_F=0, G_H=0, G_S=0):
    # List of (district, population) data (in 2011)
    population_data = (('Cambridge City', 123900),
                       ('East Cambridgeshire', 83800),
                       ('Fenland', 95300),
                       ('Huntingdonshire', 169500),
                       ('South Cambridgeshire', 148800))
    data=[]
    label=[]
    for i in range(len(population_data)):
        data.append(population_data[i][1])
        label.append(population_data[i][0])

    for i in range(year-2011):
        data[0]+=G_C/100*data[0]
        data[1]+=G_E/100*data[1]
        data[2]+=G_F/100*data[2]
        data[3]+=G_H/100*data[3]
        data[4]+=G_S/100*data[4]

    # Specify slice colours
    colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral', 'red']

    # Produce pie chart. The slices will be ordered and plotted␣
 ↪counter-clockwise.
    plt.pie(data, labels=label, colors=colors,
            autopct='%1.1f%%', shadow=True, startangle=90)
```

```
    plt.title("2011-{} population distribution".format(year))
    plt.show()
```

```
interactive(children=(IntSlider(value=2011, description='year', max=2021, min=2011), FloatSlide
```

0.4 Exercise 08.4 (crime reports by location)

0.4.1 Background

Your task is to produce a crime report data plot in the neighborhood of your college, by reported crime category.

We can get crime data in the UK from the police data systems using what is known as a *REST API*, and turn the data into a list of Python dictionaries. Each entry in the list is a police report (an entry is a Python dictionary detailing the report).

The first step is to import the modules we will be using:

[0]:
```
import json
import requests
```

The service https://data.police.uk has an interface where we can add specific strings to the URL (web address) to define what data we are intersted in, and the police server will return our requested data. The format is

`https://data.police.uk/api/crimes-street/all-crime?lat=52.629729&lng=-1.131592&date=2017-01`

This return crimes reports within a mile radius of the geographic coordinate point for the month `2017-01`.

Below we create this URL string to include a part of the Cambridge city centre. You can modify this for your own college or other areas of interest (Google Maps is a handy way to get the geographic coordinates).

[0]:
```
# A point in the Cambridge city centre
long, lat = 52.205277, 0.119117

# year-month of interest
year_month = '2019-05'

# Construct request URL string
url = "https://data.police.uk/api/crimes-street/all-crime?
 ↪lat={}&lng={}&date={}".format(long, lat, year_month)

# Fetch data from https://data.police.uk
r = requests.get(url)
```

The following converts the fetched data into a list of dictionaries:

[0]:
```
crime_data = r.json()
```

To get an idea of how the data is arranged, we can look at the first report in the list. To make the displayed data easier to read, we use the 'pretty print' module `pprint`.

[16]:
```
import pprint
if crime_data:
    pprint.pprint(crime_data[0])
```

```
{'category': 'anti-social-behaviour',
 'context': '',
 'id': 74130763,
 'location': {'latitude': '52.212138',
              'longitude': '0.136055',
              'street': {'id': 560631,
                         'name': 'On or near Cutter Ferry Close'}},
 'location_subtype': '',
 'location_type': 'Force',
 'month': '2019-05',
 'outcome_status': None,
 'persistent_id': ''}
```

Each dictionary item corresponds to a reported crime.

0.4.2 Task

Produce a bar chart of the number of reports in different categories. Run your program for different parts of Cambridge, starting with the area around your college, and for different months and years.

Hints Create an empty dictionary, which will eventually map the report category to the number of incidents:

[0]:
```
categories_freq = {}
```

Iterate over all reports in the list, and extract the category string from each report. If the category string (the 'key') is already in the dictionary increment the associated counter. Otherwise add the key to the dictionary, and associate the value 1.

[0]:
```
# Iterate over all reports
for report in crime_data:
    # Get category type
    category = report['category']

    if category in categories_freq:
        # Increment counter here
        pass  # This can be removed once the 'if' block has a body
    else:
        # Add category to dictionary here
        pass  # This can be removed once the 'else' block has a body
```

The crime categories are the dictionary keys, which can be extracted using

`list(categories_freq.keys())`

When adding the tick labels (crime categories), it may be necessary to rotate the labels, e.g.:

`plt.xticks(x_pos, categories, rotation='vertical')`

0.4.3 Extensions (optional)

1. Probe the retrieved data to build a set of all crime categories in the data set.
2. Explore the temporal (time) aspect of the data. Thinks of ways to represent the change in reported incident types over time.
3. Explore what other data you can retriev from https://data.police.uk/docs/.

0.4.4 Solution

There are many ways this exercise could be programmed, and there is some choice in data structures.

Build (a) a list of crime categories and (b) the number of reports in each category:

Solution from https://stackoverflow.com/questions/28819830/count-occurrences-of-item-in-json-element

[19]:
```
import json
import requests
from collections import Counter

# A point in the Cambridge city centre
long, lat = 52.205277, 0.119117

# year-month of interest
year_month = '2019-05'

# Construct request URL string
url = "https://data.police.uk/api/crimes-street/all-crime?
 ↪lat={}&lng={}&date={}".format(long, lat, year_month)

# Fetch data from https://data.police.uk
crime_data = requests.get(url).json()

import pprint
if crime_data:
    pprint.pprint(crime_data[0])
```

```
{'category': 'anti-social-behaviour',
 'context': '',
 'id': 74130763,
 'location': {'latitude': '52.212138',
              'longitude': '0.136055',
              'street': {'id': 560631,
                         'name': 'On or near Cutter Ferry Close'}},
 'location_subtype': '',
```

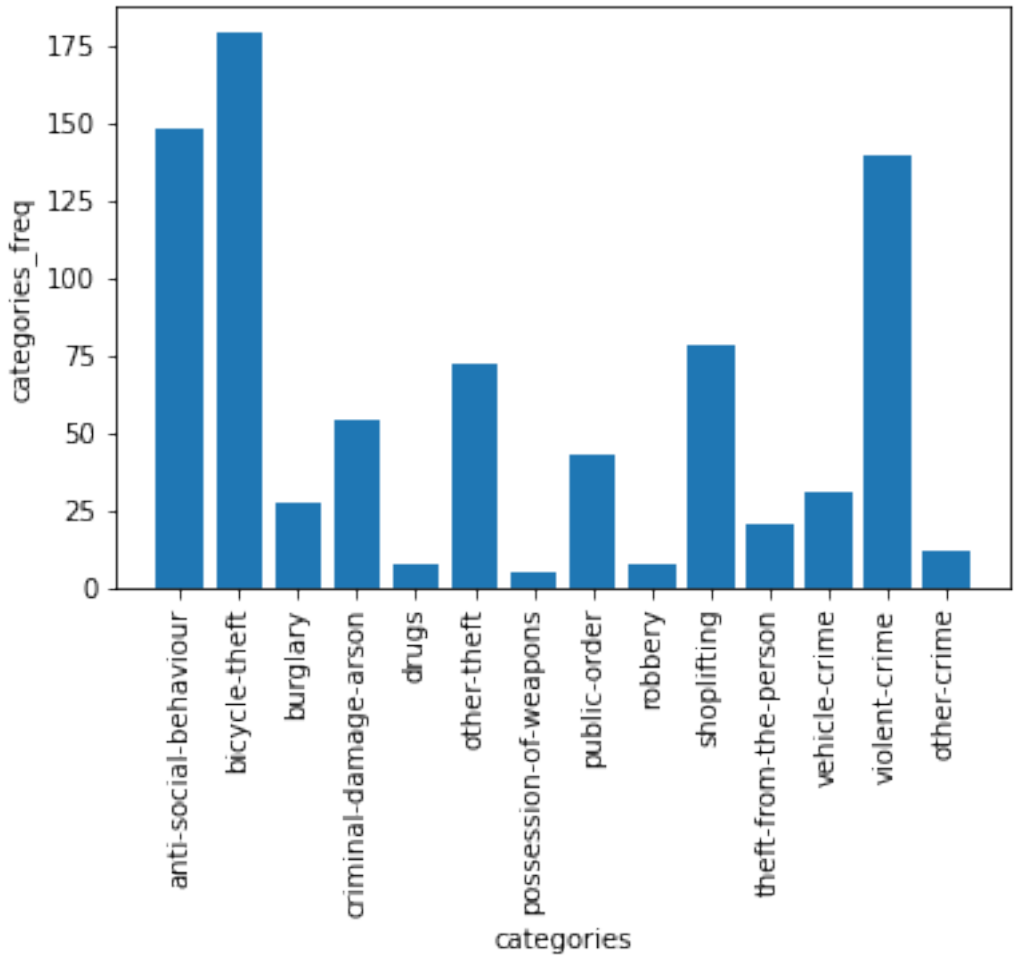
```
 'location_type': 'Force',
 'month': '2019-05',
 'outcome_status': None,
 'persistent_id': ''}
```

[32]:
```
data = Counter(report['category'] for report in crime_data)
data = dict(c)
print(data)

categories=[]
categories_freq=[]

for i, j in data.items():
    categories.append(i)
    categories_freq.append(j)
```

```
{'anti-social-behaviour': 148, 'bicycle-theft': 179, 'burglary': 27, 'criminal-
damage-arson': 54, 'drugs': 7, 'other-theft': 72, 'possession-of-weapons': 5,
'public-order': 43, 'robbery': 7, 'shoplifting': 78, 'theft-from-the-person':
20, 'vehicle-crime': 31, 'violent-crime': 140, 'other-crime': 12}
```

Produce a bar chart:

[43]:
```
# Create an array with the position of each bar along the x-axis
x_pos = np.arange(len(categories_freq))

# Produce bar plot
plt.bar(x_pos, categories_freq, align='center');

# Replace the x ticks with the Tripos name, and rotate labels vertical
plt.xticks(x_pos, categories, rotation='vertical')

# Add axis labels
plt.xlabel('categories')
plt.ylabel('categories_freq')

plt.show()
```



Here is a compact version using dictionary member functions (it will be visible when solutions are released).

[67]:
```
from ipywidgets import *
import json
import requests
from collections import Counter

long, lat = 52.205277, 0.119117
year_month='2019-05'
url = "https://data.police.uk/api/crimes-street/all-crime?
 ↪lat={}&lng={}&date={}".format(long, lat, year_month)
crime_data = requests.get(url).json()
data = Counter(report['category'] for report in crime_data)
data = dict(c)
categories = data.keys()
categories_freq = data.values()
```

```
plt.xticks(np.arange(len(categories)), categories, rotation='vertical')
plt.xlabel('categories')
plt.ylabel('categories_freq')
plt.bar(categories, categories_freq)
```

[67]: <BarContainer object of 14 artists>