UNIVERSITY of WISCONSIN
LA CROSSE
COMPUTER SCIENCE

CS 224 Introduction to Python

Plotting with Python

1

Introduction

Python is commonly used by scientists for data analysis and visualization.

As programming languages go, the "cost of entry" is low -- it may be easier for non-programmers to pickup than many other languages

Examples of modules for sciences:

- astropy – astronomy
- biopython – bioinformatics
- deap – evolutionary computation
- mlpy – machine learning
- nilearn – neuro imaging
- psychopy -- psychology

2

Tools for general purpose visualization

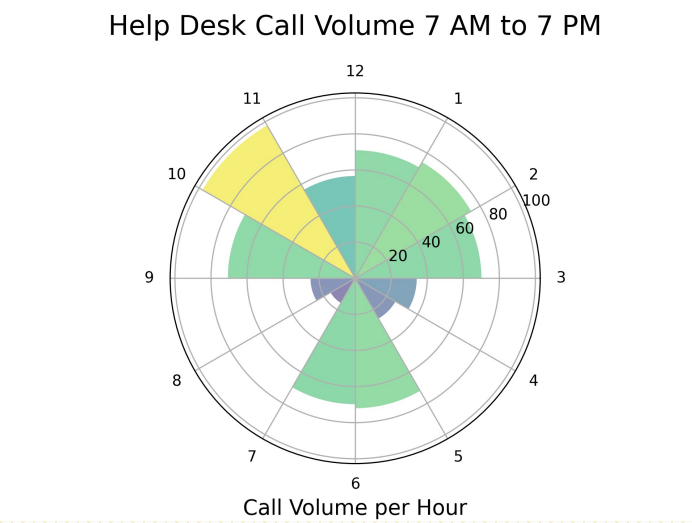
`matplotlib.pyplot:` provides an interface to `matplotlib`

`seaborn:` Python module built on matplotlib that provides a number of predefined plot types

`numpy:` Python numerical computing package known for array operations

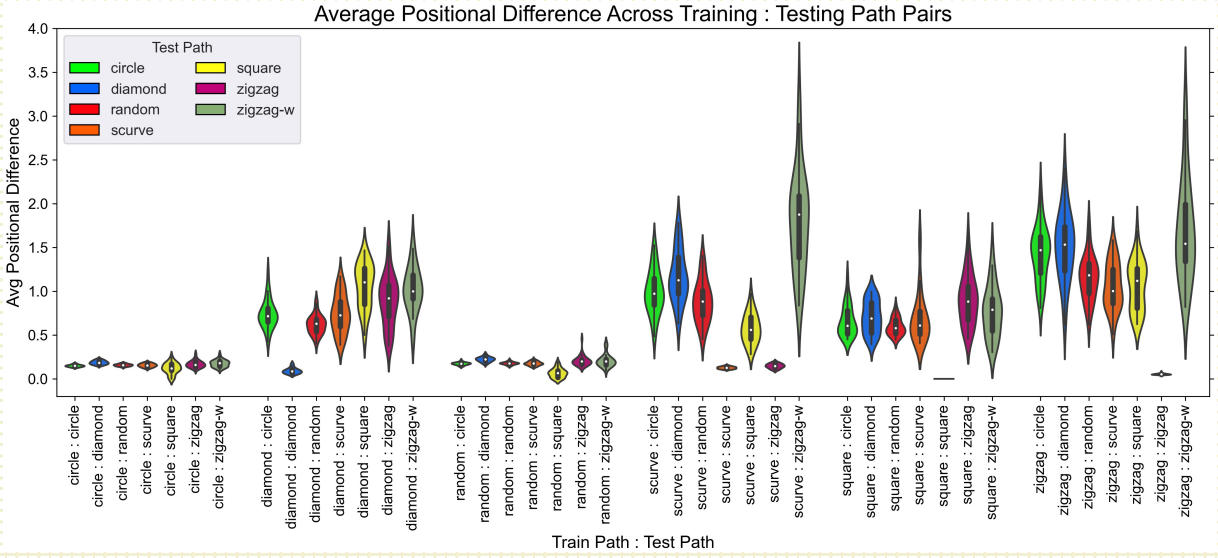
3

Inspiration



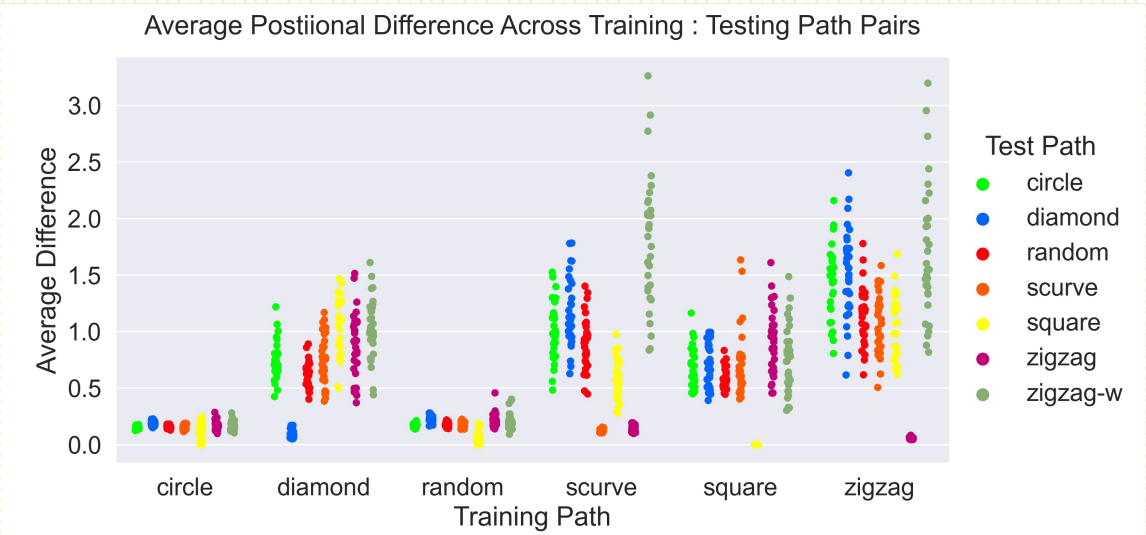
4

Inspiration



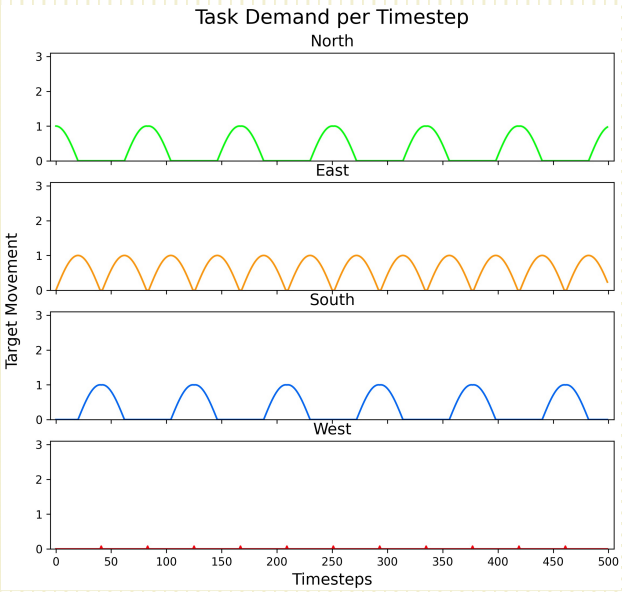
5

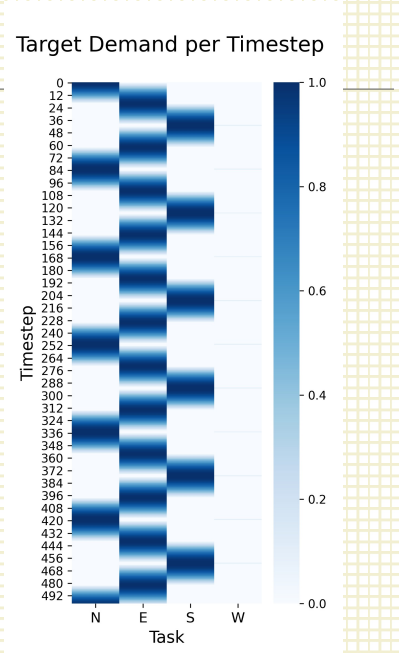
Inspiration



6

Inspiration



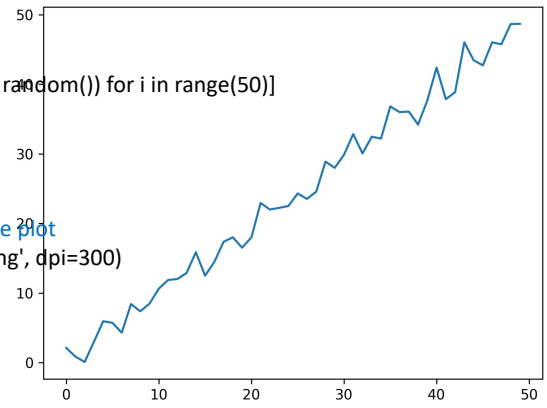


7

We need to walk before we run

A simple line plot:

```
# generate random data
y_data = [i + (randint(-5, 5) * random()) for i in range(50)]

# create a pline plot
lp = plt.plot(y_data)

# save to a file and display the plot
plt.savefig('sample_plot_1.png', dpi=300)
plt.show()
```

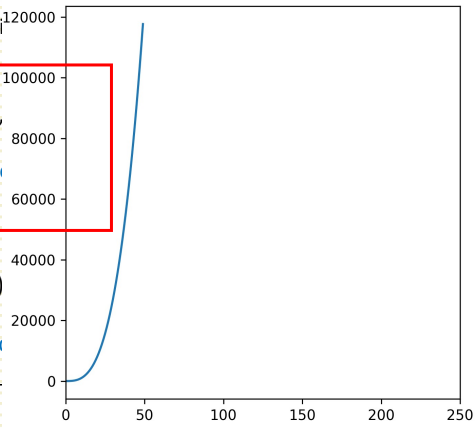


8

A line plot with new “features”

A very slightly less simple line plot:

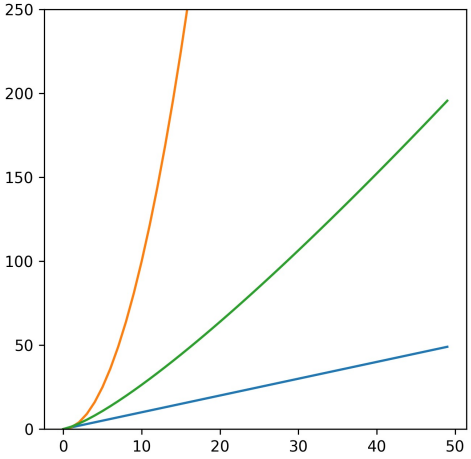
```
# generate random
y_data = [i**3 for i i

# set the figure size
plt.figure(figsize=(5,

# set the x axis valu
plt.xlim(0, 250)

# create a line plot
lp = plt.plot(y_data)

# save to a file and
plt.savefig('sample_
plt.show()
```



9

Plotting multiple data sets

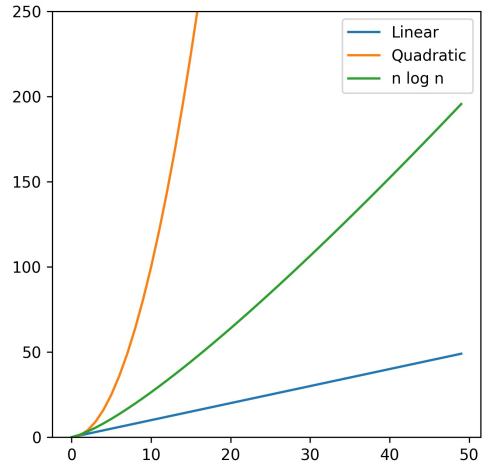
```
# generate three data sets
y_data_1 = y_data = [i for i
y_data_2 = [i**2 for i in ra
y_data_3 = [(i+1) * math.lo

# set the figure size
plt.figure(figsize=(5,5))

# set the y axis value limits
plt.ylim(0, 250)

# each call plots one of the
lp1 = plt.plot(y_data_1)
lp2 = plt.plot(y_data_2)
lp3 = plt.plot(y_data_3)

# save and display the plot
plt.savefig('sample_plot_3
plt.show()
```



10

Adding a legend

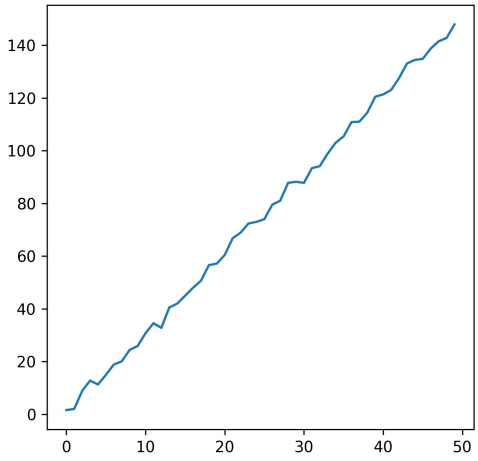
```
# create three data sets
  y_data_1 = y_data = [i for
  y_data_2 = [i**2 for i in r
  y_data_3 = [(i+1) * math.

  # set the figure size
  plt.figure(figsize=(5,5))

  # set the y axis value limi
  plt.ylim(0, 250)

  # each call plots one of th
  # label arguments for use
  lp = plt.plot(y_data_1, lab
  lp = plt.plot(y_data_2, lab
  lp = plt.plot(y_data_3, lab

  # create the legend
  plt.legend()
```



11

Something's not right here

The x values used to calculate the y values are 0 to 147 with an increment of 3 but the x-axis shows 0 to 49.

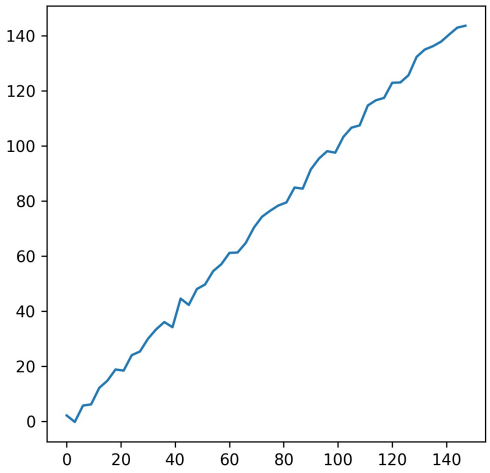


12

We can fix that…

Creating values for the x-axis:

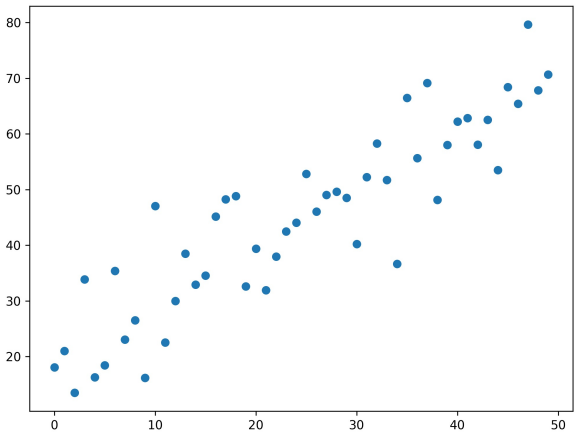
```
# fix: create a separate list for x-values and tell plt to use them
x_data = [3 * i for i in range(50)]
y_data = [i + (random.randint(-5, 5) * random.random())
                                                    for i in x_data]

# set the figure size
plt.figure(figsize=(5,5))

# for the fixed version we need to use this plot call
lp = plt.plot(x_data, y_data)
```



13

Are you tired of line plots?

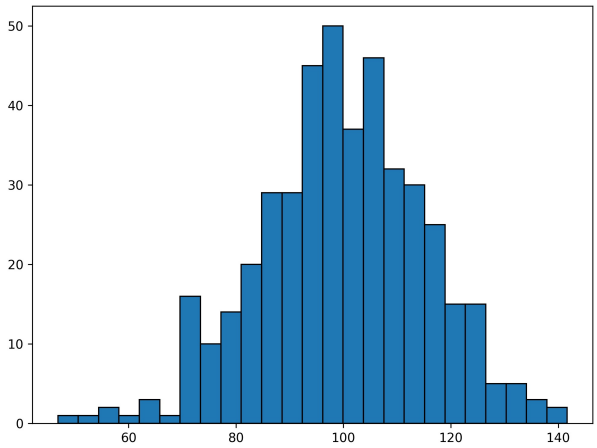
```
# this y_data has a litt
x_data = [i for i in rang
y_data = [20 + i + (ran

# set the figure size
plt.figure(figsize=(8,6)

# for the fixed version
lp = plt.scatter(x_data

# save and display the
plt.savefig('sample_pl
plt.show()
```



14

How about a histogram?

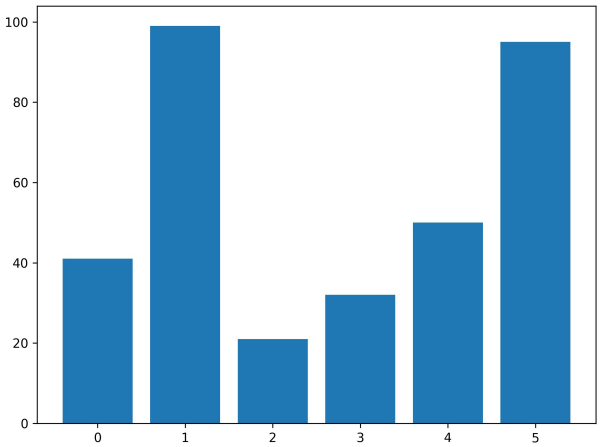
```
# mean and std deviatic
mu = 100
sigma = 15
# generate a numpy arr
# distributed random va
y_data = mu + sigma * r

# number of bins in the
bins = 25

# set the figure size
plt.figure(figsize=(8,6))

# pass the data and nur
hp = plt.hist(y_data, bin
```



15

Bar chart anyone?

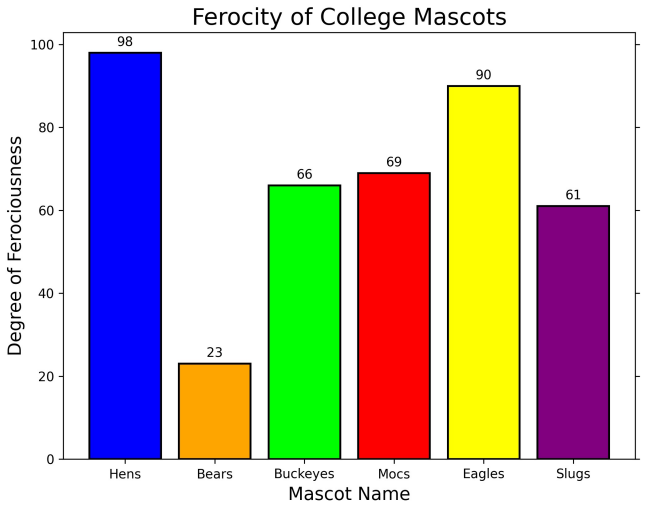
```
# create the data
x_data = [i for i in range(6)]
y_data = [random.randint(20,

# set the figure size
plt.figure(figsize=(8,6))

# pass the x locations and bar
rects = plt.bar(x_data, y_data

# save and display plot
plt.savefig('sample_plot_8.pn
plt.show()
```



16

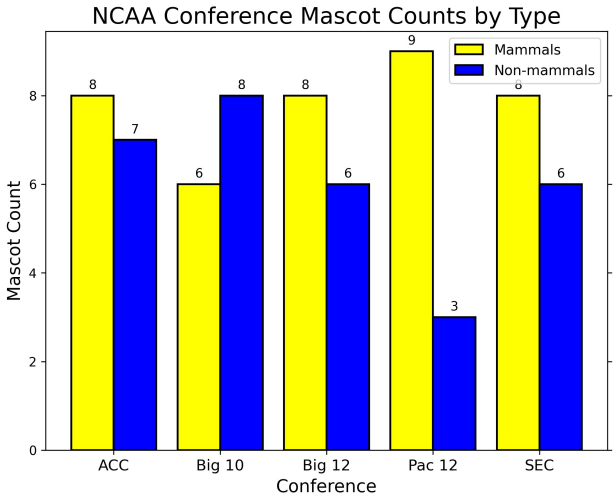
Let's make the bar chart pretty

```
# labels for the bars
x_labels = ['Hens', 'Bears',
            'Eagles', 'Slugs']

# colors for the bars
bar_colors = ['blue', 'orang
              'yellow', 'pur

#create the plot
rects = plt.bar(x_data, y_d
    color=bar_colors, edge

# axis labels and title
plt.xlabel('Mascot Name',
plt.ylabel('Degree of Feroc
plt.title('Ferocity of College
```



17

A grouped bar chart



18

Grouped bar chart code

```
# labels for the bars
conferences = ['ACC', 'Big 10', 'Big 12', 'Pac 12', 'SEC']
# mapping conference names to number of teams
size_map = {'ACC':15, 'Big 10':14, 'Big 12':14, 'Pac 12':12, 'SEC':14 }

x_data = [i for i in range(5)]

# randomly generate number of mammals
mammals = [random.randint(2, 10) for _ in range(5)]
# number of non-mammals = conference size - number of mammals
other = [size_map[conferences[i]] - mammals[i] for i in range(5)]

rects1 = plt.bar(x - width/2, mammals, width, edgecolor='black', linewidth=1.5, color='yellow', label='Mammals')
rects2 = plt.bar(x + width/2, other, width, edgecolor='black', linewidth=1.5, color='blue', label='Non-mammals')

# place tick marks on both sides of plot but tick labels on left side only
plt.tick_params(axis='y', left=True, right=True, labelleft=True, labelright=False)
# set tick labels
plt.xticks(ticks=x_data, labels=conferences, fontsize=TICKSIZE)
```

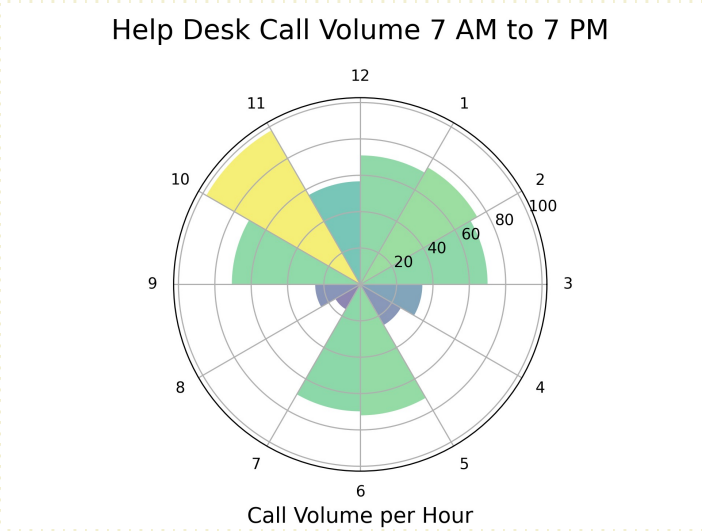
19

HowTo: Values at top of bars

```
# function to label the bars with numeric values
def rect_num_labels(bars):
    for rect in bars:
        height = rect.get_height()
        plt.annotate('{}'.format(height),
                xy=(rect.get_x() + rect.get_width() / 2, height),
                xytext=(0, 3),                  # 3 points vertical offset
                textcoords='offset points',     # set offset unit to points (rather than pixels)
                ha='center', va='bottom')

def main():

    ...
    rects1 = plt.bar(x - width/2, mammals, width, edgecolor='black', linewidth=1.5, color='yellow', label='Mammals')
    rects2 = plt.bar(x + width/2, other, width, edgecolor='black', linewidth=1.5, color='blue', label='Non-mammals')

    ...
    # call function to label the rectangles
    rect_num_labels(rects1)
    rect_num_labels(rects2)
```

20

How to make the polar bar chart



21

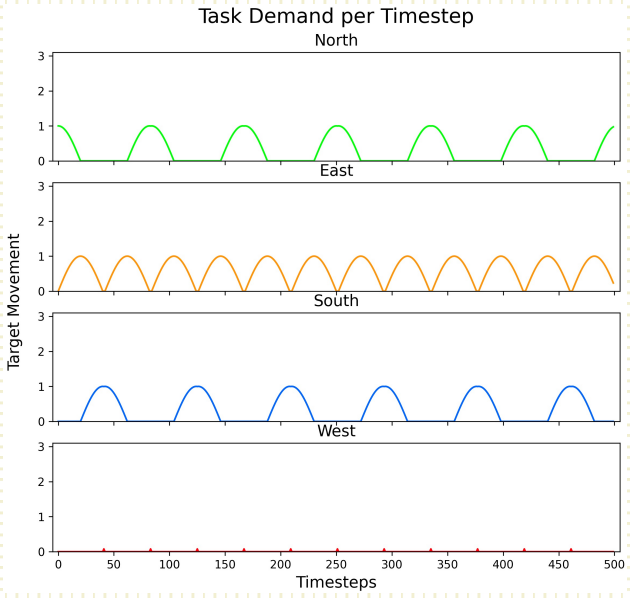
How to make the polar bar chart

```
# Number of wedges; one per hour
N = 12

# creates np array of angles at which wedges are centered -
# linspace creates evenly spaced values
theta = np.linspace(0.0, 2 * np.pi, N, endpoint=False)

# creates np array of radii in range 0.0 -- 99.99...
radii = 100 * np.random.rand(N)

# creates np array of wedge widths
# we want each wedge to represent an hour
width = 2*np.pi/12 * np.ones(12)

# colors come from the specified colormap (cm.viridis)
# are related to the radii of the wedges
colors = plt.cm.viridis(radii / 100.0)

# set for polar axes
ax = plt.subplot(111, projection='polar')
```

```
# create the plot
#   align: align left edge of wedge with ticks
#   color: the colormap used above
#   alpha: degree of opacity
pb = ax.bar(theta, radii, width=width, align='edge',
            color=colors, alpha=0.6)

# we want 12 evenly-spaced positions starting at 0 degrees
# and moving counterclockwise
xtick_pos = [i * (2*np.pi/12) for i in range(N)]
xtick_labels = ['3', '2', '1', '12', '11', '10', '9', '8', '7', '6', '5', '4']
plt.xticks(xtick_pos, labels=xtick_labels)

# set top of plot at (0.82 * height) to make room for title
plt.subplots_adjust(top=0.82)

# create the label and plot title
plt.xlabel('Call Volume per Hour', fontsize=LABELSIZE)
plt.title('Help Desk Call Volume 7 AM to 7 PM\n',
          fontsize=TITLESIZE)
```

22

How to make the multi-plot



23

How to make the multi-plot

```
# create the grid of subplots
fig, axes = plt.subplots(nrows=4, ncols=1, sharex=True, figsize=(10, 8))

# create the subplots
for i in range(len(task_dict)):
    sns.lineplot(ax=axes[i], y=data[i], x=timesteps, color=plot_colors[i])
    axes[i].set_title(task_dict[i], fontsize=LABELSIZE)
    axes[i].set_ylim(0, 3.1)
    axes[i].set_xlim(-5, num_pts + 5)

# create the x labels
x_tick_labels = [str(i) for i in range(num_pts + 5) if i % tick_interval == 0]
axes[0].set_xticks([i for i in range(num_pts + 5) if i % tick_interval == 0])
axes[0].set_xticklabels(x_tick_labels, fontsize=TICKSIZE)

# this allows "global" axis labels (rather than labels for each subplot)
fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none', top=False, bottom=False, left=False, right=False)
plt.xlabel('Timesteps', fontsize=LABELSIZE)
plt.ylabel('Target Movement', fontsize=LABELSIZE)
```

24

Quiz 1

Problem 2:

Write Python code that takes two lists (assume that they already exist) named **list1** and **list2** and prints all combinations of an element from **list1** followed by an element from **list2**. Each combination should be printed on a separate line.

Solution:

```
for e in list1:
    for f in list2:
        print('({}, {})'.format(e, f))
```

Alternate solution 1:

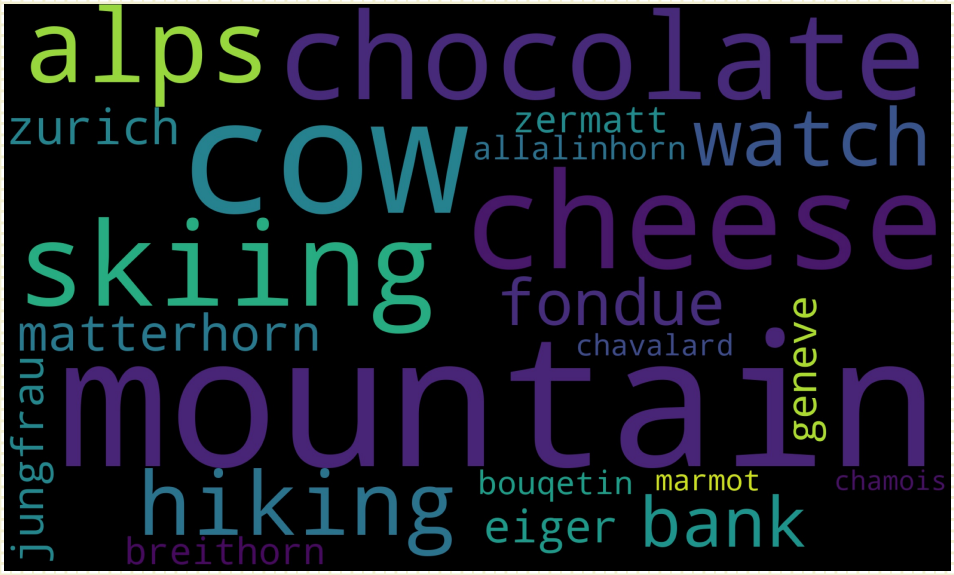
```
pairs = [(e, f) for e in list1 for f in list2]
for p in pairs:
        print(p)
```

Alternate solution 2:

```
pairs = itertools.product(list1, list2)
for p in pairs:
        print(p)
```

25

And now for something completely different…



26

How to make the word cloud

```
from wordcloud import WordCloud

cloud = WordCloud(width=2000, height=1200, background_color='black', collocations=False).generate(' '.join(words))

plt.imshow(cloud)
plt.axis('off')
```

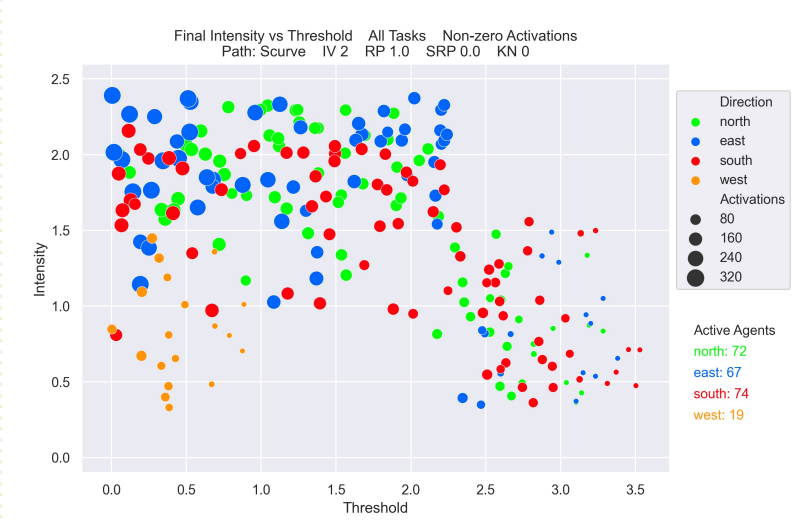
words is a list of the words to display

Tying this to recorded lecture:
join is a function that creates a string from a list of strings

27

Coming attractions

Plotting with Seaborn



28

Quiz 1

Problem 1:

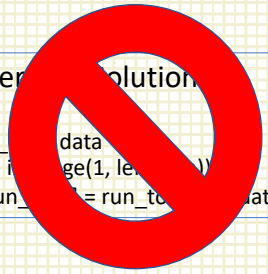
Write Python code that takes a list named **data** and creates a list called **run_tot** that contains a running total of the elements in **data**. In other words, each element **i** of **run_tot** is the sum of all elements with indices **j <= i**. YOU MAY **NOT** USE A LIST COMPREHENSION. Assume that **data** already exists.

Solution:

```
run_tot = [data[0]]
for i in range(1, len(data)):
    run_tot.append(run_tot[i-1] + data[i])
```

Alter olution:

```
run_  data
for i i  ge(1, le  ))
    run_  = run_to  ata[i]
```



Alternate Solution 2:

```
run_tot = deepcopy(data)
for i in range(1, len(data)):
    run_tot[i] = run_tot[i-1] + data[i]
```

29