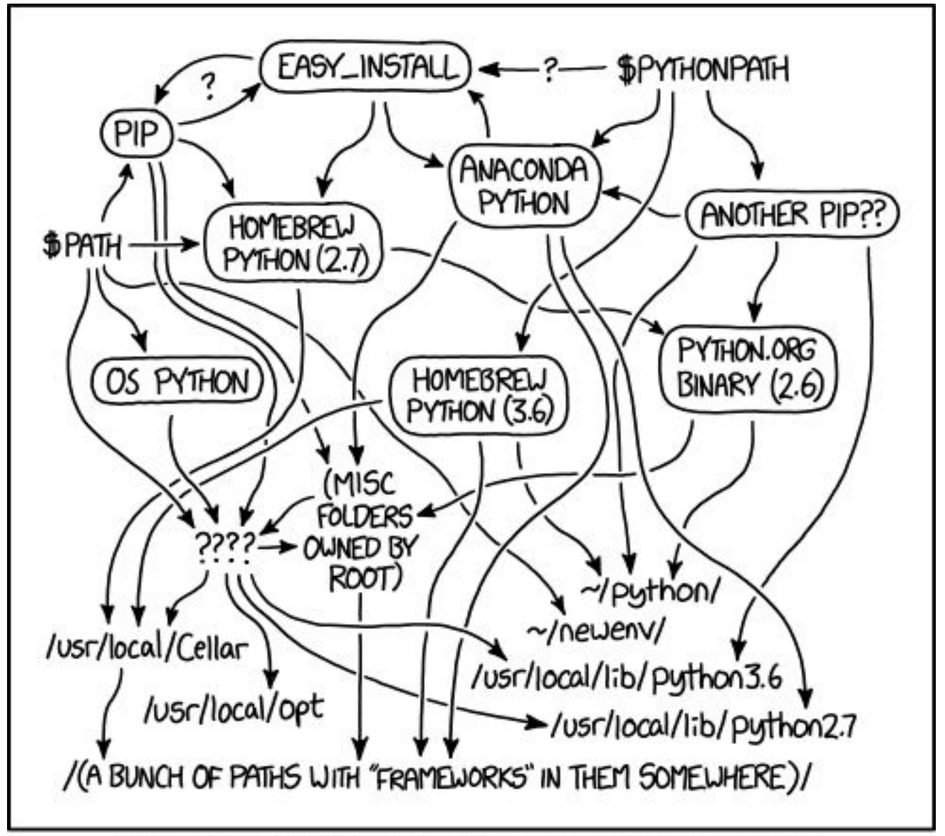
Python Part 4

Spring 2023
PCfB Class 7
March 3, 2023



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Outline

- Stand-alone python scripts
- **`argparse`** module
- Functions
- Sets

Stand-alone python scripts

To make a text file a Python script

`argparse` module

1. argparse generates help documentation

```
(base) jtl-macbook-pro:Demos jtladner$ ./calc.py -h
usage: calc.py [-h] [-o {+,-,*,/,**,%}] a b

positional arguments:
  a                     First number to use in calculation
  b                     Second number to use in calculation

optional arguments:
  -h, --help            show this help message and exit
  -o {+,-,*,/,**,%}, --operation {+,-,*,/,**,%}
                        Operation to perform
```

```
import argparse

parser = argparse.ArgumentParser()

args = parser.parse_args()
```

```
(base) jtl-macbook-pro:Demos jtladner$ ./test.py -h
usage: test.py [-h]

optional arguments:
  -h, --help  show this help message and exit
```

Positional arguments

Optional arguments

```
parser = argparse.ArgumentParser()


parser.add_argument("num",
help="Number of hits to report")

args = parser.parse_args()
```

```
usage: test.py [-h]

optional arguments:
  -h, --help  show this help message and exit
```

```
usage: test.py [-h] num

positional arguments:
  num         Number of hits to report

optional arguments:
  -h, --help  show this help message and exit
```

```
parser = argparse.ArgumentParser()


parser.add_argument("-n", "--num",
help="Number of hits to report")


args = parser.parse_args()
```

```
usage: test.py [-h]

optional arguments:
  -h, --help  show this help message and exit
```

```
usage: test.py [-h] [-n NUM]

optional arguments:
  -h, --help            show this help message and exit
  -n NUM, --num NUM     Number of hits to report
```

Functions

```
def sqrt(num):
    squareroot = float(num)**(0.5)
    return squareroot
```

```
def sqrt(num):
    squareroot = float(num)**(0.5)
    return squareroot
```

```
numstring="123456789"
sqroots = [sqrt(x) for x in numstring]
```

```
def sqrt(num):
    squareroot = float(num)**(0.5)
    return squareroot
```

```
numstring="123456789"
sqroots = [sqrt(x) for x in numstring]
```

```
sqroots = [float(x)**(0.5) for x in numstring]
```

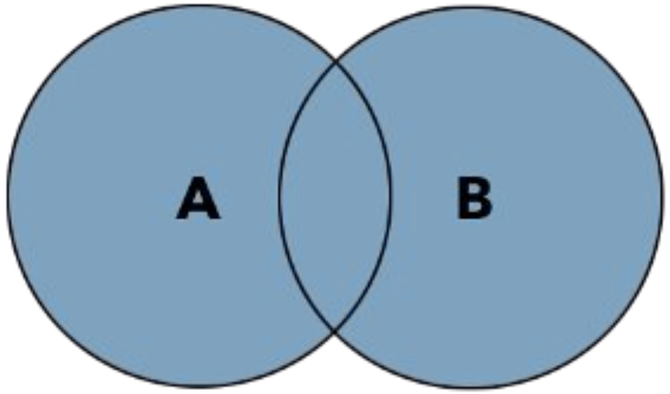
```
def mergeClusts(clusts, k, indivThresh, clustThresh):
    pairs2merge = []
    for i in range(len(clusts)):
        for j in range(i+1, len(clusts)):
            hits=0
            comps=0
            for n1,s1 in clusts[i].items():
                for n2,s2 in clusts[j].items():
                    comps+=1
                    if kmerOvlp(s1, s2, k)>=indivThresh:
                        hits+=1
            if hits/comps >= clustThresh:
                pairs2merge.append([i,j])
    
    groups2merge = combPairs(pairs2merge)
    
    newClusts = []
    singles = []
    merged = []
    for each in groups2merge:
        newClusts.append({})
        for a in each:
            merged.append(a)
            for k,v in clusts[a].items():
                newClusts[-1][k] = v
    for i,info in enumerate(clusts):
        if i not in merged:
            if len(info) == 1:
                singles.append(info)
            else:
                newClusts.append(info)
    return newClusts, singles
```

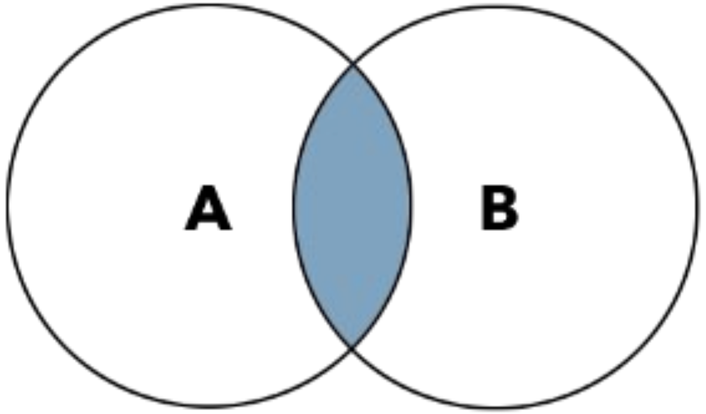
Sets

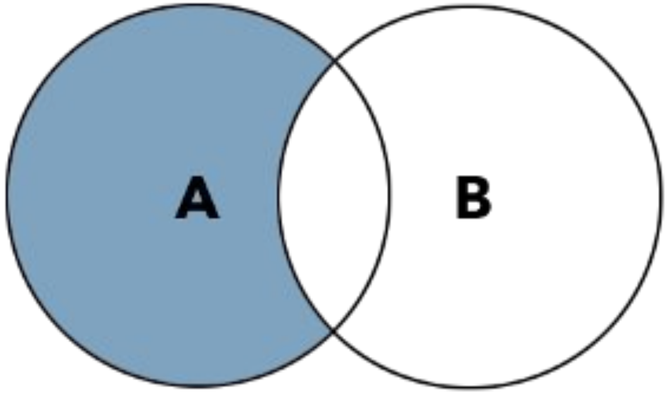
Sets

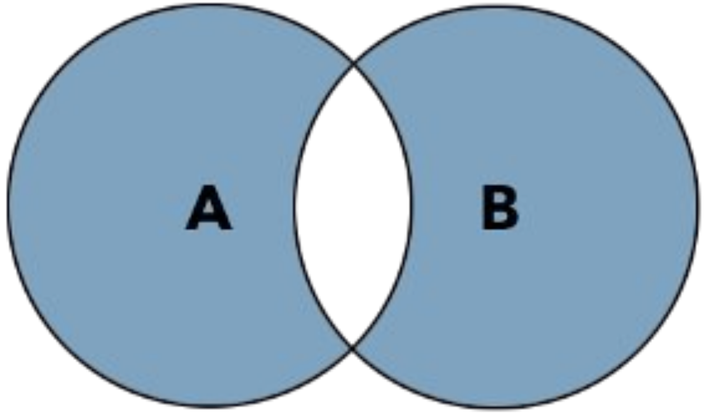
Compare collections of items

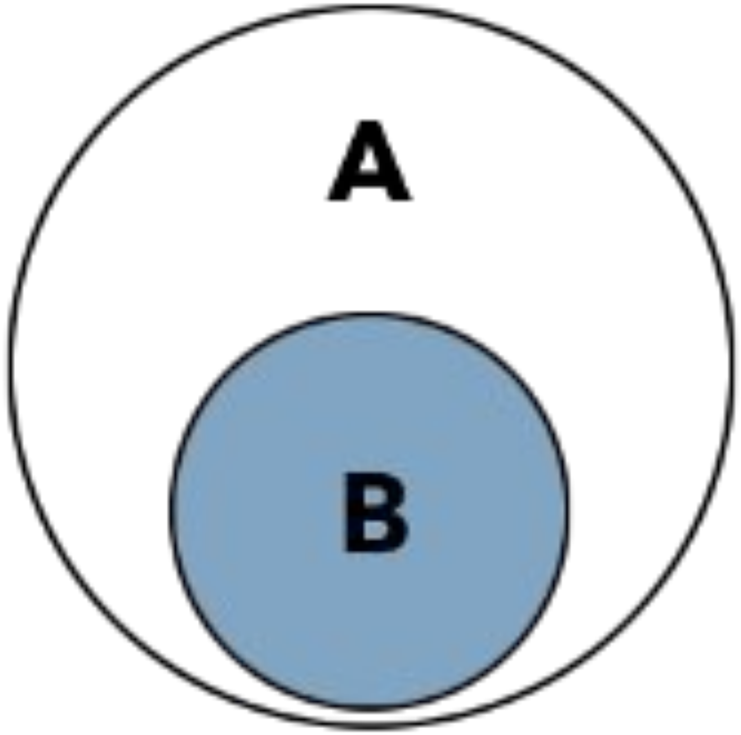
```
A.intersection(B)

A.union(B)

A.difference(B)

A.symmetric_difference(B)

A.issubset(B)

A.issuperset(B)
```


A
B


A
B


A
B


A
B


A
B