# File I/O

SURP 2022 Python Bootcamp

Ohio State Astronomy

Slides by: James W. Johnson

# Objectives

An example of how to write to a text file with standard library

An example of reading that same text file with standard library

How NumPy simplifies things

# An Example Output File

Write the integers 0 through 9 and their squares separated by tabs, with each integer-square pair on separate lines

```
In [1]: with open("example.txt", 'w') as f:
   ...:     for i in range(10):
   ...:         f.write(str(i))
   ...:         f.write("\t")
   ...:         f.write(str(i**2))
   ...:         f.write("\n")
   ...:     f.close()
```

```
(base) Cosmo:OSUAstroSURP2020 astrobeard$ more example.txt
0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
```

Note: file I/O is typically done with a *with* statement – this functionally is the same as using *f = open("example.txt", 'w')*

# An Example Output File: A Better Code

Makes use of *string formatting*: *%d* allows integers to be substituted in their place

```
In [1]: with open("example.txt", 'w') as f:
   ...:     for i in range(10):
   ...:         f.write("%d\t%d\n" % (i, i**2))
   ...:     f.close()
   ...:
```

```
(base) Cosmo:OSUAstroSURP2020 astrobeard$ more example.txt
0       0
1       1
2       4
3       9
4       16
5       25
6       36
7       49
8       64
9       81
```

There are many ways to format string in python: The %-style notation will look familiar if you have experience with C/C++

https://docs.python.org/3.10/library/string.html

# An Example Input File

Read in the same file and store it in a 2D-list

```
(base) Cosmo:OSUAstroSURP2020 astrobeard$ more example.txt
0       0
1       1
2       4
3       9
4       16
5       25
6       36
7       49
8       64
9       81
```

```
In [1]: x = []

In [2]: with open("example.txt", 'r') as f:
   ...:     line = f.readline()
   ...:     while line != "":
   ...:         x.append([int(i) for i in line.split()])
   ...:         line = f.readline()
   ...:     f.close()
   ...:

In [3]: x
Out[3]:
[[0, 0],
 [1, 1],
 [2, 4],
 [3, 9],
 [4, 16],
 [5, 25],
 [6, 36],
 [7, 49],
 [8, 64],
 [9, 81]]
```

Note: Some argue that *append* is a bad idea when lists are large, but others argue that the performance issues that arose are largely resolved

# A Much Simpler Approach

File I/O is made easy with NumPy's *genfromtxt*, *savetxt*, etc. functions

These functions read and write 1-D and 2-D data to files.

- Caveat: The data must be square. Chances are you'll get non-square data at some point

In the previous example, the entire *with* block can be replaced by:

*import numpy as np*

*x = np.genfromtxt("example.txt")*

(Note however this returns a NumPy array, not a list)