

# Introduction to python

– with Application to Bioinformatics

Syntax

||

Programming

Best movie  
per category



step1.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Find and print the best movie per category
# =====

|
```

U: step1.py All L8 (Python +2 MM) 19:23 1.63



step2.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Find and print the best movie per category
# =====

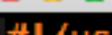
with open('250.imdb', 'r', encoding='utf-8') as f:

    for line in f:

        # I have a line
        # What now?
```

U:--- step2.py All L1 (Python +2 MM) 19:23 1.63  
Use +,-,0 for further adjustment





step3.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Find and print the best movie per category
# =====

with open('250.imdb', 'r', encoding='utf-8') as f:

    for line in f:

        if line.startswith('#'): # Not interested
            continue

        # Get the fields as a list of strings
        fields = line.split('|')

        # Rename the fields, cuz I prefer, and convert them
        rating = float(fields[1])
        genres = fields[-2].lower().split(',') # List of strings also

        # Now what?
        # I need a global data structure to remember those values
```

U: step3.py All Li (Python +2 MM) 19:23 1.63  
Use +,-,0 for further adjustment



```
step4.py  
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
# =====  
# Find and print the best movie per category  
# =====  
  
with open('250.imdb', 'r', encoding='utf-8') as f:  
  
    # For each category, I keep the best rating  
    # Mapping { key: value } where key = string  
    #                                     value = int  
    categories = {} # Nothing at the start  
  
    for line in f:  
  
        if line.startswith('#'): # Not interested  
            continue  
  
        # Get the fields as a list of strings  
        fields = line.split('|')  
  
        # Rename the fields, cuz I prefer, and convert them  
        rating = float(fields[1])  
        genres = fields[-2].lower().split(',') # List of strings also  
  
        for genre in genres:  
            old_rating = categories.get(genre, 0)  
  
            if rating > old_rating: # found a better one  
                categories[genre] = rating  
  
    # Print the categories  
    for genre, rating in categories.items():  
        print("The best movie for", genre, "has rating:", rating)
```

U:--- step4.py All L1 (Python +2 MM) 19:22 1.20  
Use +,-,0 for further adjustment



tmux new-session -A -s main

```
[~/tmp] (vt17) $ python step4.py
Traceback (most recent call last):
  File "step4.py", line 28, in <module>
    old_rating = categories[genre]
KeyError: 'drama'
[~/tmp] (vt17) $
```

step5.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Find and print the best movie per category
# =====

with open('250.imdb', 'r', encoding='utf-8') as f:

    # For each category, I keep the best rating
    # Mapping { key: value } where   key = string
    #                                     value = int
    categories = {} # Nothing at the start

    for line in f:

        if line.startswith('#'): # Not interested
            continue

        # Get the fields as a list of strings
        fields = line.split('|')

        # Rename the fields, cuz I prefer, and convert them
        rating = float(fields[1])
        genres = fields[-2].lower().split(',') # List of strings also

        for genre in genres:
            old_rating = categories.get(genre, 0.0) # No KeyError

            if rating > old_rating: # found a better one
                categories[genre] = rating

    # Print the categories
    for genre, rating in categories.items():
        print("The best movie for", genre, "has rating:", rating)
```

U:--- step5.py All L1 (Python +2 MM) 19:21 1.08  
Use +,-,0 for further adjustment



```
tmux new-session -A -s main
[~/tmp] (vt17) $ python step5.py
The best movie for war has rating: 8.6
The best movie for thriller has rating: 9.0
The best movie for music has rating: 8.5
The best movie for horror has rating: 8.5
The best movie for biography has rating: 9.0
The best movie for history has rating: 8.9
The best movie for fantasy has rating: 8.9
The best movie for sport has rating: 9.0
The best movie for comedy has rating: 8.8
The best movie for mystery has rating: 8.6
The best movie for western has rating: 8.9
The best movie for drama has rating: 9.3
The best movie for sci-fi has rating: 8.8
The best movie for film-noir has rating: 8.5
The best movie for family has rating: 8.6
The best movie for musical has rating: 8.6
The best movie for crime has rating: 9.3
The best movie for historical has rating: 8.1
The best movie for romance has rating: 8.6
The best movie for adventure has rating: 8.9
The best movie for action has rating: 9.0
The best movie for animation has rating: 8.6
[~/tmp] (vt17) $
```



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Find and print the best movie per category
# =====

with open('250.imdb', 'r', encoding='utf-8') as f:

    # For each category, I keep the best rating
    # Mapping { key: value } where   key = string
    #                                     value = int
    categories = {} # Nothing at the start

    for line in f:

        if line.startswith('#'): # Not interested
            continue

        # Get the fields as a list of strings
        fields = line.split('|')

        # Rename the fields, cuz I prefer, and convert them
        rating = float(fields[1])
        genres = fields[-2].lower().split(',') # List of strings also

        for genre in genres:
            genre = genre[:6] # Cheating
            old_rating = categories.get(genre, 0.0) # No KeyError

            if rating > old_rating: # found a better one
                categories[genre] = rating

    # Print the categories
    for genre, rating in categories.items():
        print("The best movie for", genre, "has rating:", rating)
```

U:--- step6.py All L1 (Python +2 MMN) 19:21 1.08  
Use +,-,0 for further adjustment



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Find and print the best movie per category
# =====

with open('250.imdb', 'r', encoding='utf-8') as f:

    # For each category, I keep the best rating
    # Mapping { key: value } where   key = string
    #                                     value = (int,string)
    categories = {} # Nothing at the start

    for line in f:

        if line.startswith('#'): # Not interested
            continue

        # Get the fields as a list of strings
        fields = line.split('|')

        # Rename the fields, cuz I prefer, and convert them
        rating = float(fields[1])
        title = fields[-1].strip() # Clean the title
        genres = fields[-2].lower().split(',') # List of strings also

        for genre in genres:
            genre = genre[:6]
            old_rating,old_title = categories.get(genre, (0.0,'')) # No KeyError

            if rating > old_rating: # found a better one
                categories[genre] = (rating, title)

# Print the categories
for genre,value in categories.items():
    print("The best movie for",genre,'\n\tis"',value[1],"\n\tand has rating:",value[0])
```

U:--- step7.py All L1 (Python +2 MMN) 19:20 1.25  
Use +,-,0 for further adjustment



```
tmux new-session -A -s main
    is " The Dark Knight "
        and has rating: 9.0
The best movie for comedy
    is " Forrest Gump "
        and has rating: 8.8
The best movie for wester
    is " The Good, the Bad and the Ugly "
        and has rating: 8.9
The best movie for sport
    is " Dangal "
        and has rating: 9.0
The best movie for animat
    is " Spirited Away "
        and has rating: 8.6
The best movie for crime
    is " The Shawshank Redemption "
        and has rating: 9.3
The best movie for biogra
    is " Dangal "
        and has rating: 9.0
The best movie for music
    is " Like Stars on Earth "
        and has rating: 8.5
The best movie for fantas
    is " The Lord of the Rings: The Return of the King "
        and has rating: 8.9
[~/tmp] (vt17) $
```

molcellbio161:main

bash \*

19:11 Tue, Feb 7, 2017



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Find and print the best movie per category
# =====

with open('250.imdb', 'r', encoding='utf-8') as f:

    # For each category, I keep the best rating
    # Mapping { key: value } where   key = string
    #                                     value = (int,string,string)
    categories = {} # Nothing at the start

    for line in f:

        if line.startswith('#'): # Not interested
            continue

        # Get the fields as a list of strings
        fields = line.split('|')

        # Rename the fields, cuz I prefer, and convert them
        rating = float(fields[1])
        title = fields[-1].strip() # Clean the title
        genres = fields[-2].lower().split(',') # List of strings also

        for genre in genres:
            key = genre[:6]
            old_rating,old_title,old_genre = categories.get(key, (0.0,'','')) # No KeyError

            if rating > old_rating: # found a better one
                categories[key] = (rating, title, genre.capitalize())

# Print the categories
for (rating,title,category) in categories.values():
    print("The best movie for",category,'is',title,'"and has rating:',rating)
```

```
is " Life Is Beautiful "
and has rating: 8.6
The best movie for Action
is " The Dark Knight "
and has rating: 9.0
The best movie for History
is " Amadeus "
and has rating: 8.3
The best movie for Horror
is " Alien "
and has rating: 8.5
The best movie for Animation
is " Spirited Away "
and has rating: 8.6
The best movie for Musical
is " Sholay "
and has rating: 8.4
The best movie for Fantasy
is " Spirited Away "
and has rating: 8.6
The best movie for Crime
is " The Shawshank Redemption "
and has rating: 9.3
The best movie for Biography
is " Amadeus "
and has rating: 8.3
[~/tmp] (vt17) $
```

molcellbio161:main

bash \*

19:16 Tue, Feb 7, 2017



Find most votes per category

Find longest movie per year

Rewrite the file in this other format

```
> CATEGORY  
movie  
movie  
movie
```

Rewrite the file in this other format

> CATEGORY

movie

movie

movie

← Rating      Name (Year)  
tab

> DRAMA

9.3 Shawshank... (1996)

8.7 Blabla (2017)

> MUSICAL

8.3 Something (2000)

8.7 Blabla (2017)



Recreate or Crash

```
with open('output.txt', 'w', encoding='utf-8') as f:  
  
    f.write('Something')  
    f.write('\n')  
    f.write('Something Else')  
    f.write('\n')  
    f.write('Something Interesting\n')
```

```
with open('input.txt', 'r', encoding='utf-8') as f_input:  
    with open('output.txt', 'w', encoding='utf-8') as f_output:  
  
        f_output.write('# FORMAT:\n')  
        f_output.write('# > CATEGORY\n')  
        f_output.write('# Movie: Rating \t Name (Year)\n')  
  
    for line in f_input:  
        if line.startswith('#'):  
            continue  
  
        # Do something with that line
```

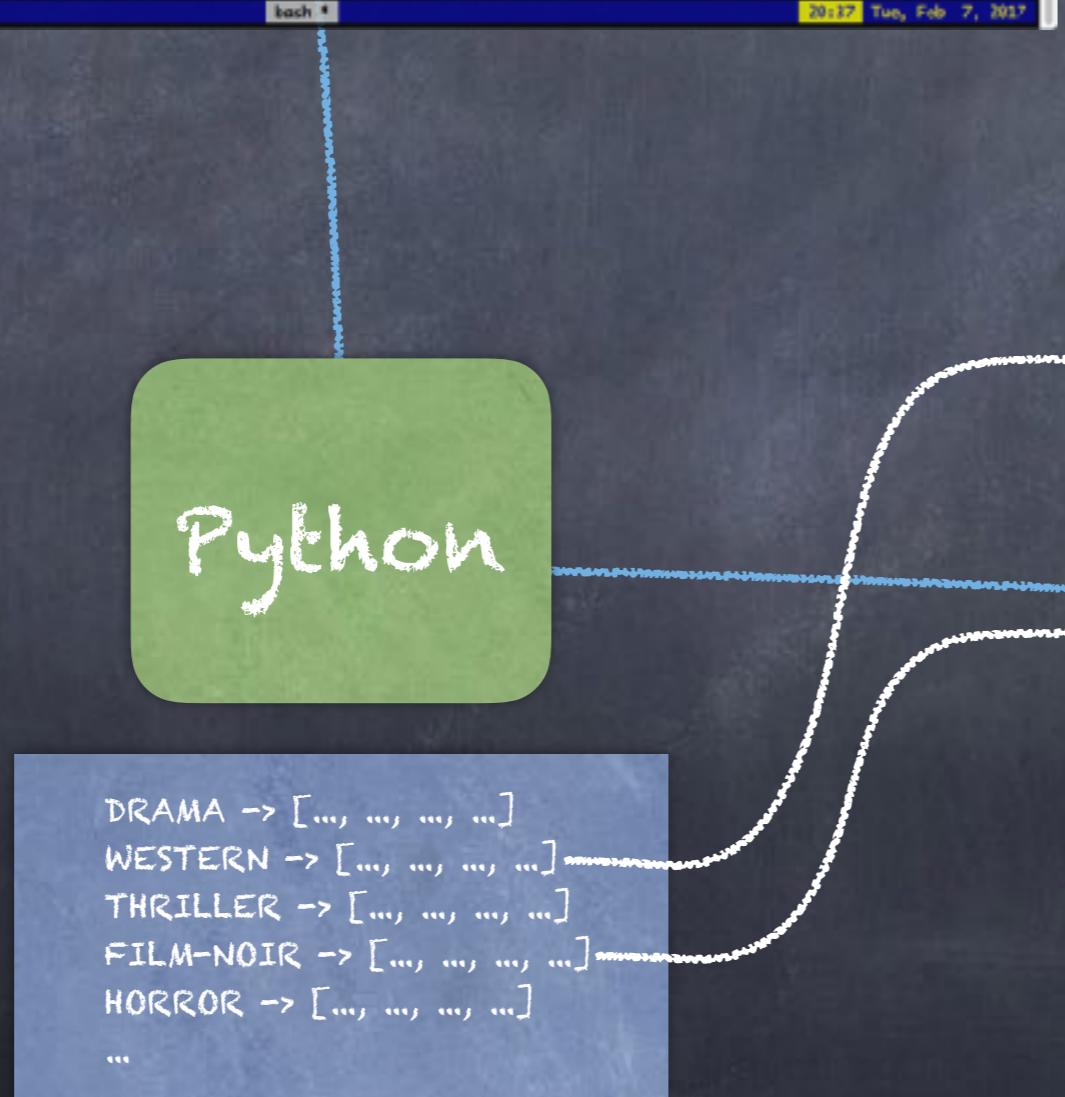
```
with open('a', 'r') as f_input, open('b', 'w') as f_output:  
  
    f_output.write('# FORMAT:\n')  
    f_output.write('# > CATEGORY\n')  
    f_output.write('# Movie: Rating \t Name (Year)\n')  
  
    for line in f_input:  
        if line.startswith('#'):  
            continue  
  
        # Do something with that line
```



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Reformat 250.imdb into other format
# =====
```

	Votes	Rating	Year	Runtime	URL	Genres	Title
126887	8.51	957152981	https://images-eu.ssl-images-amazon.com/images/M/M5B0T15Nbc0JUNEYzBm688MjoxLThzLbN14200gx02NSN-N00Ey0fFcc0d0XVvA03Mh05kz				
713791	8.21	1925143281	https://images-eu.ssl-images-amazon.com/images/M/M5BNEY4M0QNeAMNSBM158cn8nX0FzZT1w4DUS005...V1..jpg	Adventure, Comedy, Drama, F			
704991	8.31	2089157591	https://images-eu.ssl-images-amazon.com/images/M/M5BNT3NE2z34WF5BM158cn8nX0FzZTgnZEJNEMT0...V1..jpg	Action, Adventure, ,			
276121	8.31	1928168461	https://images-eu.ssl-images-amazon.com/images/M/M5BNjx40jk500Lw15M158cn8nX0FzZTg00T2MjLMwF...V1..jpg	Biography, Drama, Hist			
224481	8.41	1995184561	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZTgnju40DE4Mz0B...V1..jpg	Lion, Adventure, Cri			
1853421	8.21	1954164381	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZTow1E20T3M1EB...V1..jpg	Drama, Thriller			
1155231	8.11	1995179491	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZtgMD4Nj3dHT0...V1..jpg	Biography, Drama, In			
224991	8.31	1952177491	https://images-eu.ssl-images-amazon.com/images/M/M5BNjx40jk500Lw15M158cn8nX0FzZt0WtCjU0CmWE...V1..jpg	Crime, Drama, Thriller			
158821	8.21	1986174481	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZt0WtCjU0CmWE...V1..jpg	Crime, Drama, Thriller			
6544101	8.31	1995140601	https://images-eu.ssl-images-amazon.com/images/M/M5BNDj22001Mj1h3HTRHjy002TASLWechD5jVhNe2TdxTV1ZBjLokdy0fqc0d0XVvA0020144kz				
1365391	8.11	1995168681	https://images-eu.ssl-images-amazon.com/images/M/M5B2D12TIANYzAH0D3N180CTRjLTkd2LHGE3DM4Zh1ANTUXXEjy0fFcc0d0XVvA0007000T				
1368381	8.11	19951588801	https://images-eu.ssl-images-amazon.com/images/M/M5BNDH12A5Vjy209M00Cq2LW6SM1HGe0N8LYj244R1xdy0fqc0d0XVvA004j05ta				
5876221	8.31	1957169981	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZtgMD4Nj3dHT0...V1..jpg	Drama, Thriller			
1817601	8.11	1984157901	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZtgMD4Nj3dHT0...V1..jpg	Drama, Thriller			
5875281	8.41	1986124811	https://images-eu.ssl-images-amazon.com/images/M/M5BNGyXTA0MEELYjy0EYj002L5LtgRNjE1ZD4Hntcy0fM0RTVjL21LYM0L2LLYRt0KxEyXKf9G				
7501701	8.61	1994166201	https://images-eu.ssl-images-amazon.com/images/M/M5BNjx40jk500Lw15M158cn8nX0FzZt0WtCjU0CmWE...V1..jpg				
2772001	8.21	1957170081	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZtgMD4Nj3dHT0...V1..jpg				
3294411	8.41	1941171401	https://images-eu.ssl-images-amazon.com/images/M/M5BNTQ2M1CD00M158cn8nX0FzZt0WtCjU0CmWE...V1..jpg	Drama, Mystery, Cr			
1556461	8.31	1955179881	https://images-eu.ssl-images-amazon.com/images/M/M5BNTQ2M1j0TEzMF5BM158cn8nX0FzZt0WtCjU0CmWE...V1..jpg	Western, For a Few			
8262591	8.51	1995111340	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZt0WtCjU0CmWE...V1..jpg	C, Drama, Fantasy, Mys			
1221441	8.61	2006176881	https://images-eu.ssl-images-amazon.com/images/M/M5BNTw15M158cn8nX0FzZt0WtCjU0CmWE...V1..jpg	Drama, Musical			
2890641	8.31	1920178501	https://images-eu.ssl-images-amazon.com/images/M/M5B003BYY1MhC42Y3N180CTy0LIMhZGzL2NE3D1YH1T1DNUHx0fFcc0d0XVvA0007000T				
5815441	8.31	1976167981	https://images-eu.ssl-images-amazon.com/images/M/M5BNG0XNgLz0WzLTMjN1802NLExZNE1NE1jEy0fEhM2ASXeKy0fqc0d0XVvA0007000T				
15322651	8.81	20061183801	https://images-eu.ssl-images-amazon.com/images/M/M5BNjx40jk500Lw15M158cn8nX0FzZt0WtCjU0CmWE...V1..jpg	Action, Adventure, Sci			
18723451	8.61	1995176281	https://images-eu.ssl-images-amazon.com/images/M/M5BCTjw00M6MTLzJu0h60CTx4LtgSNM1UmWHTAdzTNjrcy0fqc0d0XVvA0007000T				
311141	8.21	1948178801	https://images-eu.ssl-images-amazon.com/images/M/M5NDRj2TRjMhC7DVMjA002jARlWf8Mh4jL0ZTAr0g240T121+Y88T1.71Y88T1.0FyXkFeyG				
3465171	8.31	200616142301	https://images-eu.ssl-images-amazon.com/images/M/M5BNTx4fj9Dg4f150M158cn8nX0FzZtg00cyNjA50T0B...V1..jpg	Drama, Nonfiction, by			
1398381	8.21	1995177981	https://images-eu.ssl-images-amazon.com/images/M/M5BNjw1L8012875BM158cn8nX0M1z10M1H3M1UyN0E...V1..jpg	Drama, Mystery, Thrill			
4810241	8.21	19821663801	https://images-eu.ssl-images-amazon.com/images/M/M5B2W12YhEYT1hGRjyA002m6LwEdM04202NjZLNTMh4jQ0xdy0fqc0d0XVvA0007000T				
352181	8.31	2007199861	https://images-eu.ssl-images-amazon.com/images/M/M5BNTv1yTkbNjAyjzr3M0801j...j1TlkYekTzgjYhMy000ix0tly0kq0d0XVvA0007000T				
3364421	8.11	1995194801	https://images-eu.ssl-images-amazon.com/images/M/M5BNTx40004DL5W5BM158cn8nX0FzZt0WtCjU0CmWE...V1..jpg	Drama, Historical, The			
325421	8.31	1946175501	https://images-eu.ssl-images-amazon.com/images/M/M5BNTQ4Mj0U0Tw5V5BM158cn8nX0FzZtgMD4HngyHj0B...V1..jpg	Adventure, Drama, West			



```

8.4  Shaw (1975)
8.2  The General (1926)
8.5  The Dark Knight Rises (2012)
8.7  Star Wars: Episode IV - A New Hope (1977)
8.1  Mad Max: Fury Road (2015)
> WESTERN
8.3  For a Few Dollars More (1965)
8.3  Unforgiven (1992)
8.3  The Treasure of the Sierra Madre (1948)
8.6  Once Upon a Time in the West (1968)
8.9  The Good, the Bad and the Ugly (1966)
8.1  Butch Cassidy and the Sundance Kid (1969)
8.4  Django Unchained (2012)
8.2  The General (1926)
> FILM-NOIR
8.1  Touch of Evil (1958)
8.1  Strangers on a Train (1951)
8.2  Dial M for Murder (1954)
8.3  The Third Man (1949)
8.1  The Maltese Falcon (1941)
8.4  Double Indemnity (1944)
8.5  Sunset Blvd. (1950)
> ROMANCE
8.1  Before Sunrise (1995)
8.6  La La Land (2016)
8.2  Rebecca (1940)
8.1  The Princess Bride (1987)
8.3  Rango De Rosanti (2006)
8.6  City Lights (1931)
8.3  Sunrise (1927)
8.3  Eternal Sunshine of the Spotless Mind (2004)
8.4  American Beauty (1999)
8.2  Gone with the Wind (1939)
:
```

molcellbio161:main

less \*





```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Reformat 250.imdb into other format
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:

    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    for line in f_input:

        if line.startswith('#'): # Not interested
            continue

        # Reformate that line
        # and put it somewhere to remember it with its category
```



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Reformat 250.imdb into other format
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:

    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:

        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip() # clean it
        year = fields[2].strip() # it too
        rating = fields[1].strip() # who knows...
```



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Reformat 250.imdb into other format
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:

    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:

        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip() # clean it
        year = fields[2].strip() # it too
        rating = fields[1].strip() # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories[genre]
```

```
step5.py
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Reformat 250.imdb into other format
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:

    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:

        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip() # clean it
        year = fields[2].strip() # it too
        rating = fields[1].strip() # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories.get(genre, [])
            movies.append(genre)
            categories[genre] = movies
```

|

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# =====
# Reformat 250.imdb into other format
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:

    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:

        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip()           # clean it
        year = fields[2].strip()            # it too
        rating = fields[1].strip()          # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories.get(genre, [])
            movies.append(new_line)
            categories[genre] = movies

    # Done constructing the intermediate data structure
    # Can dump it into the output file now
    for cat,movies in categories.items():

        # Print category first, with '> '
        f_output.write('> ')
        f_output.write(cat)
        f_output.write('\n')

        # Print all movies for that category after
        for m in movies:
            f_output.write(m)
            f_output.write('\n')
```



```
final.py
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:

    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:

        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip()           # clean it
        year = fields[2].strip()            # it too
        rating = fields[1].strip()          # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories.get(genre)
            if movies is None:
                categories[genre] = [new_line] # one item
            else:
                movies.append(new_line)

    # Done constructing the intermediate data structure
    # Can dump it into the output file now
    for cat,movies in categories.items():

        # Print category first, with '> '
        f_output.write('> ')
        f_output.write(cat)
        f_output.write('\n')

        # Print all movies for that category after
        for m in movies:
            f_output.write(m)
            f_output.write('\n')
```

```

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
open('output.txt', 'w', encoding='utf-8') as f_output:
    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:
        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip() # clean it
        year = fields[2].strip() # it too
        rating = fields[1].strip() # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories.get(genre, [])
            movies.append(new_line)
            categories[genre] = movies

    # Done constructing the intermediate data structure
    # Can dump it into the output file now
    for cat,movies in categories.items():

        # Print category first, with '> '
        f_output.write('> ')
        f_output.write(cat)
        f_output.write('\n')

        # Print all movies for that category after
        for m in movies:
            f_output.write(m)
            f_output.write('\n')

```

```

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
open('output.txt', 'w', encoding='utf-8') as f_output:
    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:
        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip() # clean it
        year = fields[2].strip() # it too
        rating = fields[1].strip() # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories.get(genre)
            if movies is None:
                categories[genre] = [new_line] # one item
            else:
                movies.append(new_line)

    # Done constructing the intermediate data structure
    # Can dump it into the output file now
    for cat,movies in categories.items():

        # Print category first, with '> '
        f_output.write('> ')
        f_output.write(cat)
        f_output.write('\n')

        # Print all movies for that category after
        for m in movies:
            f_output.write(m)
            f_output.write('\n')

```

Another format?

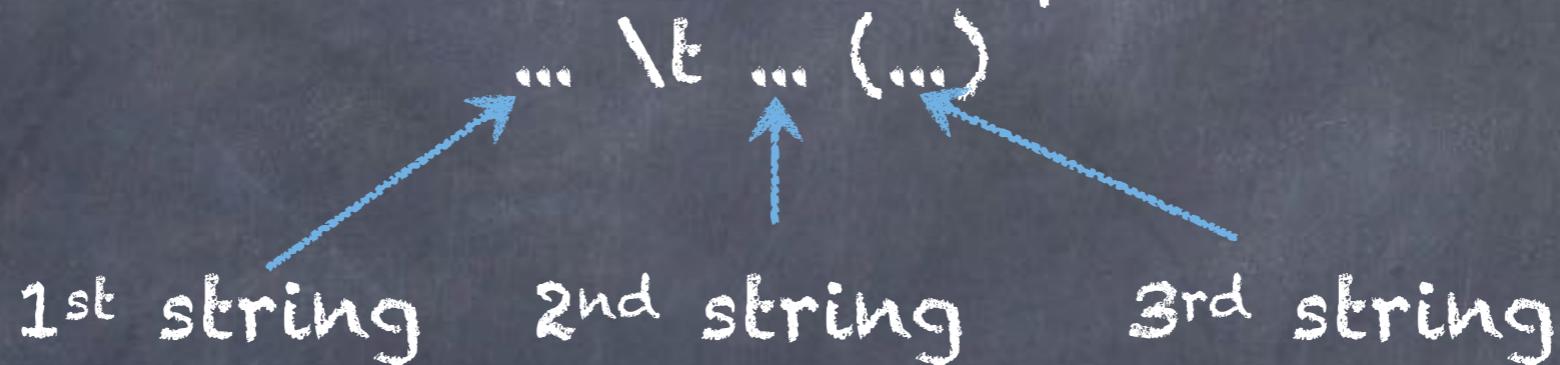
#-\*- CATEGORY -\*-#

movie

movie

movie ←———— Name ; Year ; Rating ; Votes

Given 3 strings,  
define a function that outputs them like



```
def format_nicely(str1,str2,str3):  
  
    ans = str1 + '\t' + str2 + ' (' + str3 + ')'  
  
    return ans
```

```
triplets = [
    ('10', 'Great Movie', '2003'),
    ('9.4', 'Some Movie', '2017'),
    ('8.6', 'Not so good', '2000'),
]

for (a,b,c) in triplets:

    print( format_nicely( a,b,c ) )
```

```
tmp.py
```

```
def format_nicely(str1,str2,str3):
    ans = str1 + '\t' + str2 + ' (' + str3 + ')'
    return ans

triplets = [
    ('10', 'Great Movie', '2003'),
    ('9.4', 'Some Movie', '2017'),
    ('8.6', 'Not so good', '2000'),
]
for (a,b,c) in triplets:
    print( format_nicely( a,b,c ) )
```

```
[[molcellbio161: ~]] $ python tmp.py
10      Great Movie (2003)
9.4     Some Movie (2017)
8.6     Not so good (2000)
[[molcellbio161: ~]] $ |
```

```
-:--- tmp.py      All L1      (Python +2 NMM) 22:45 1.06
U:-- *shell*      All L5      (Shell:run +2) 22:45 1.06
```

```
triplets = [
    ('10', 'Great Movie', '2003' ),
    ('9.4', 'Some Movie', '2017' ),
    ('8.6', 'Not so good', '2000' ),
]
for (a,b,c) in triplets:
    print( format_nicely( a,b,c ) )
```

```
def format_nicely(str1,str2,str3):
    ans = str1 + '\t' + str2 + ' (' + str3 + ')'
    return ans
```

```
*shell*
[[molcellbio161: ~]] $ python tmp.py
Traceback (most recent call last):
  File "tmp.py", line 9, in <module>
    print( format_nicely( a,b,c ) )
NameError: name 'format_nicely' is not defined
[[molcellbio161: ~]] $
```

-!— tmp.py All L23 (Python +2 MMM) 22:46 1.51  
Use +,-,0 for further adjustment

U!\*\*- \*shell\* All L6 (Shell:run +2) 22:46 1.51



```
def functionName(arg1, arg2, arg3):  
    someValue = ... # Make it something  
  
    # Some code with someValue  
    # using arg1, arg2, arg3 (or not!)  
    #  
    # Some more code  
  
    return someValue
```

```
def functionName(arg1, arg2, arg3):  
  
    someValue = ... # Make it something  
  
    # Some code with someValue  
    # using arg1, arg2, arg3 (or not!)  
    #  
    # Some more code  
  
    return someValue
```

→ Notebook 5

```
# Some code for x1, x2 and x3  
v = functionName(x1,x2,x3)  
# Some code using v
```

```
def functionName( parameters ):
```

```
    someValue = ... # Make it something
```

```
    # Some code with someValue  
    # using parameters (or not!)  
    #  
    # Some more code
```

```
return someValue
```

Might have a default value

\* Positional arguments  
\* Keyword arguments

```
# Some code for x1, x2 and x3  
v = functionName( parameters )  
# Some code using v
```

```
def functionName( parameters ):
    "Documentation" # Usually a triple quote

    someValue = ... # Make it something

    # Some code with someValue
    # using parameters (or not!)
    #
    # Some more code

return someValue
```

```
# ---  
  
with open('250.imdb',  
          open('output.txt'  
              f_output.write('#  
f_output.write('#  
f_output.write('#  
  
# Main data struct  
categories = {}  
# Mapping: category  
  
for line in f_input:  
    if line.startswith('#'):  
        continue  
  
    # Get some info about that line  
    fields = line.split('|')  
  
    genres = fields[-2].upper().split(',') # List of strings (uppercase)  
    title = fields[-1].strip() # clean it  
    year = fields[2].strip()  
    rating = fields[1].strip()  
  
    new_line = rating + '\t' + title + '\n' + year + '\n'  
  
    for genre in genres: # upper case  
        # Get the list of movies  
        movies = categories.get(genre)  
        if movies is None:  
            categories[genre] = []  
        else:  
            movies.append(new_line)  
  
# Done constructing the intermediate data structure  
# Can dump it into the output file now  
for cat,movies in categories.items():  
    fc = format_category(cat,movies)  
    f_output.write(fc)  
    f_output.write('\n')
```

```
# Done constructing the intermediate data structure  
# Can dump it into the output file now  
for cat,movies in categories.items():  
  
    # Print category first, with '> '  
    f_output.write('> ')  
    f_output.write(cat)  
    f_output.write('\n')  
  
    # Print all movies for that category after  
    for m in movies:  
        f_output.write(m)  
        f_output.write('\n')
```

```
final.py
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:

    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:

        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip() # clean it
        year = fields[2].strip() # it too
        rating = fields[1].strip() # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories.get(genre)
            if movies is None:
                categories[genre] = [new_line] # one item
            else:
                movies.append(new_line)

    # Done constructing the intermediate data structure
    # Can dump it into the output file now
    for cat,movies in categories.items():

        # Print category first, with '> '
        f_output.write('> ')
        f_output.write(cat)
        f_output.write('\n')

        # Print all movies for that category after
        for m in movies:
            f_output.write(m)
            f_output.write('\n')

U:--- final.py      Bot L51   (Python -2 NMM) 21:55 1.24
```

```
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:
    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:
        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip() # clean it
        year = fields[2].strip() # it too
        rating = fields[1].strip() # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

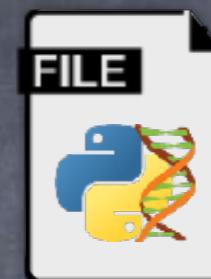
            # Get the list of movies for that genre
            movies = categories.get(genre)
            if movies is None:
                categories[genre] = [new_line] # one item
            else:
                movies.append(new_line)

    # Done constructing the intermediate data structure
    # Can dump it into the output file now
    for cat,movies in categories.items():

        # Print category first, with '> '
        f_output.write('> ')
        f_output.write(cat)
        f_output.write('\n')

        # Print all movies for that category after
        for m in movies:
            f_output.write(m)
            f_output.write('\n')

U:---- final.py      Bot L51   (Python +2 MM) 21:55 1.24
```



# extra.py

```
def func1():
```

10

10

```
def func2(arg1,arg2):
```

10

```
final.py
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:
    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:
        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip()           # clean it
        year = fields[2].strip()            # it too
        rating = fields[1].strip()          # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories.get(genre)
            if movies is None:
                categories[genre] = [new_line] # one item
            else:
                movies.append(new_line)

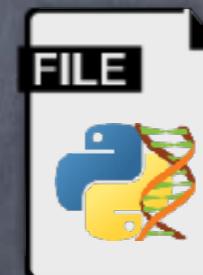
    # Done constructing the intermediate data structure
    # Can dump it into the output file now
    for cat,movies in categories.items():

        # Print category first, with '> '
        f_output.write('> ')
        f_output.write(cat)
        f_output.write('\n')

        # Print all movies for that category after
        for m in movies:
            f_output.write(m)
            f_output.write('\n')

U:--- final.py      Bot L51      (Python -2 MMN) 21:55 1.24
```

extra.py



def func1():

...

...

def func2(arg1,arg2):

...

from extra import func1

```
final.py
# =====

with open('250.imdb', 'r', encoding='utf-8') as f_input, \
    open('output.txt', 'w', encoding='utf-8') as f_output:
    f_output.write('# FORMAT:\n')
    f_output.write('# > CATEGORY\n')
    f_output.write('# Movie: Rating \t Name (Year)\n')

    # Main data structure
    categories = {}
    # Mapping: category => list of movies (already formatted)

    for line in f_input:
        if line.startswith('#'): # Not interested
            continue

        # Get some info about that line
        fields = line.split('|')

        genres = fields[-2].upper().split(',') # List of strings (uppercase)
        title = fields[-1].strip()           # clean it
        year = fields[2].strip()            # it too
        rating = fields[1].strip()          # and it too, who knows...

        new_line = rating + '\t' + title + ' (' + year + ')'

        for genre in genres: # uppercase already

            # Get the list of movies for that genre
            movies = categories.get(genre)
            if movies is None:
                categories[genre] = [new_line] # one item
            else:
                movies.append(new_line)

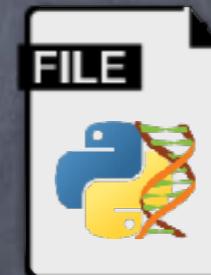
    # Done constructing the intermediate data structure
    # Can dump it into the output file now
    for cat,movies in categories.items():

        # Print category first, with '> '
        f_output.write('> ')
        f_output.write(cat)
        f_output.write('\n')

        # Print all movies for that category after
        for m in movies:
            f_output.write(m)
            f_output.write('\n')

U:--- final.py      Bot L51      (Python -2 MMN) 21:55 1.24
```

extra.py



def func1():

...

...

def func2(arg1,arg2):

...

from extra import func1

from extra import func1 as the\_func

```
○ ○ ○ /usr/local/bin/bash --noprofile — -bash --noprofile
$ python code.py

$ python
Python 3.5.0 (default, Sep 25 2015, 16:02:14)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import code
from extra import ****
Hello, how are you?
Good, and you?
sep=*****great
*****
print(sep)
print_something("Can't talk. Doing Python")
Hello, how are you?
print(sep)
print_something("Can't talk. Doing Python")
*****
Hello, how are you?
Can't talk. Doing Python
>>> [REDACTED]

code.py    All L
End of buffer
```

```
code.py
```

```
$ python code.py
Python 3.5.0 (default, Sep 25 2015, 16:02:14)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import extra
Hello, how are you?
sep = Good, and you?
__name__ is set to extra
print(sep)
print>>>
$ python extra.py
printHello, how are you?
printGood, and you?
print(sep)
print__name__ is set to __main__
$
```

```
extra.py
```

```
# Testing my code
print_something( 'Good, and you?' )
```

```
code.py      All L10  (Python -3 NMM) 02:02 1.51
End of buffer
```

```
/usr/local/bin/bash --noprofile -- bash --noprofile
$ python
Python 3.5.0 (default, Sep 25 2015, 16:02:14)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import extra
>>>
from extra import print_something
>>> # no print

sep = '*' * 20
>>>

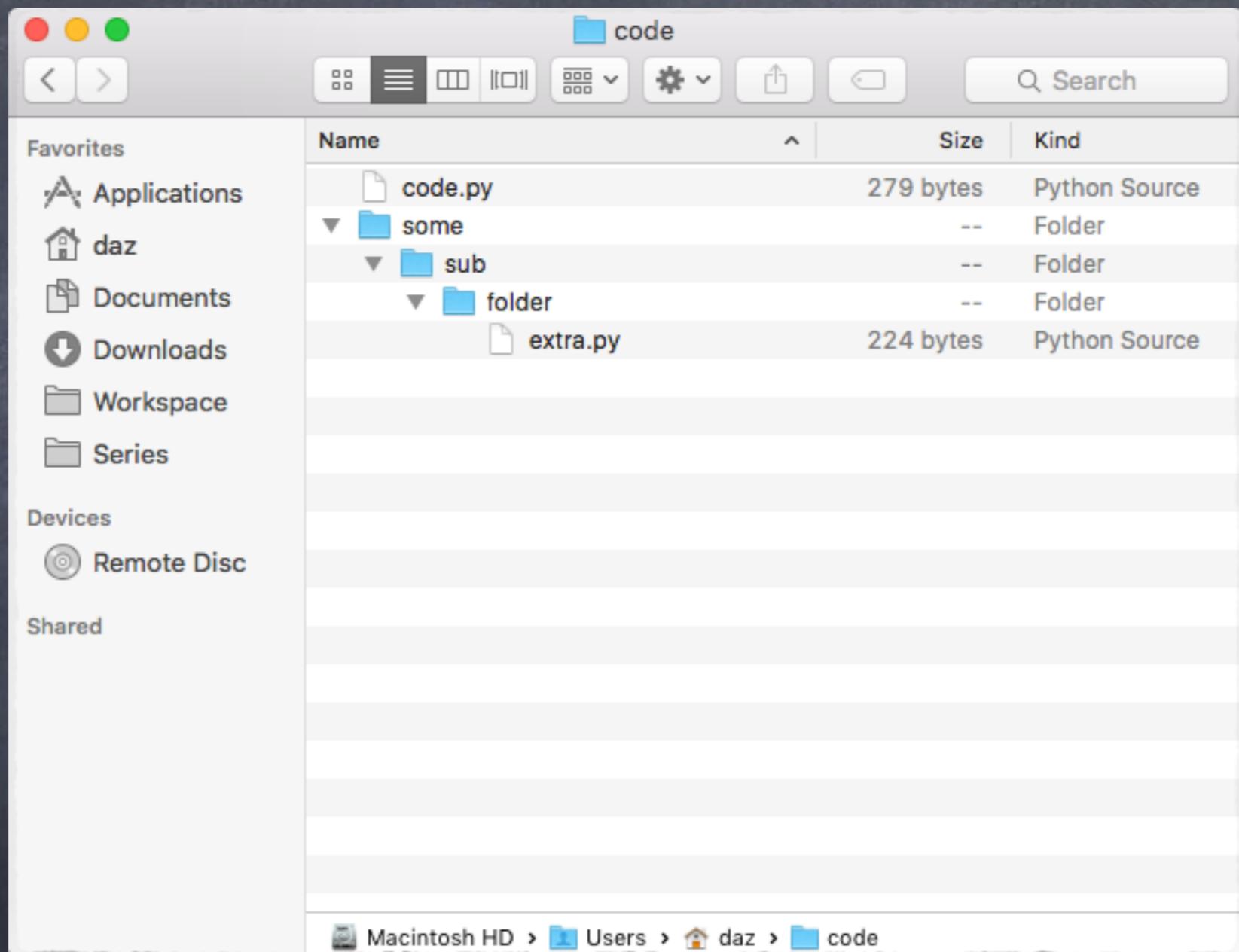
print($ python extra.py
print_something('Fine, thanks')
    Hello, how are you?
print('Good, and you?
print_something()
print($
print(sep)
print_something("Can't talk. Doing Python")
```

```
extra.py      All L14  (Python -3 NMM) 01:54 1.54
----- extra.py      All L14  (Python -3 NMM) 01:54 1.54

def print_something(arg = 'great'):
    print('Hello, how are you?')
    print( arg )
    #print("__name__ is set to", __name__)
if __name__ == '__main__':
    # Testing my code
    print_something( 'Good, and you?'
```

```
$ python code.py
*****
Hello, how are you?
Fine, thanks
*****
from extra import print_something
Hello, how are you?
def work():
    great
    sep = '*' * 20
    *****
    print(sep)
    print_something('Fine, thanks')
    Can't talk. Doing Python
    print(sep)
    print_something("Can't talk. Doing Python")
if __name__ == '__main__':
    work()
```

```
def print_something(arg = 'great'):
    print('Hello, how are you?')
    print(arg)
    #print("__name__ is set to %s, __name__" % __name__)
if __name__ == '__main__':
    # Testing my code
    print_something('Good, aren't you?')
```



code.py<untitled folder>

```
from some.sub.folder.extra import print_something

def work():
    sep = '*' * 20

    print(sep)
    print_something('Fine, thanks')

    print(sep)
    print_something()

    print(sep)
    print_something("Can't talk. Doing Python")

if __name__ == '__main__':
    work()
```

-:--- code.py<untitled folder> All L21 (Python +2 MMM) 03:23 1.31

```
○ ○ ● /usr/local/bin/bash --noprofile — -bash --noprofile
```

```
$ python code.py
*****
Hello, how are you?
Fine, thanks
*****
Hello, how are you?
great
*****
Hello, how are you?
Can't talk. Doing Python
$
```

sys.argv

## Command-line arguments

A List of strings

Position 0: the program name

Position 1: the first argument

...

```
code.py
import sys

def say_hello(name='World'):
    print('Hello', name, '!')

if __name__ == '__main__':
    if len(sys.argv) > 1: # we received arguments
        say_hello( sys.argv[1] ) # the first one
    else:
        say_hello()

U:--- code.py      All L16      (Python +2 MMM) 02:46 1.10
Wrote /Users/daz/code.py
```

```
import sys  
  
def say_hello(name='World'): # prints "Hello [name]!"  
    print('Hello', name, '!')  
  
if __name__ == '__main__':  
    if len(sys.argv) > 1: # we received arguments  
        say_hello( sys.argv[1] ) # the first argument  
    else:  
        say_hello()
```

```
■ /usr/local/bin/bash --noprofile -- -bash --noprofile  
$ python code.py  
Hello World !  
$ python code.py Fred  
Hello Fred !  
$ python code.py Fred and some more  
Hello Fred !  
$
```

U:--- code.py All L16 (Python +2 MMM) 02:46 1.10  
Wrote /User/daz/code.py

