Introduction to

– with Application to Bioinformatics

NBIS

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 MMM) 19:23 1.63

step2.py

```python
#!/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 MNM) 19:23 1.63
Use +,-,0 for further adjustment

step3.py

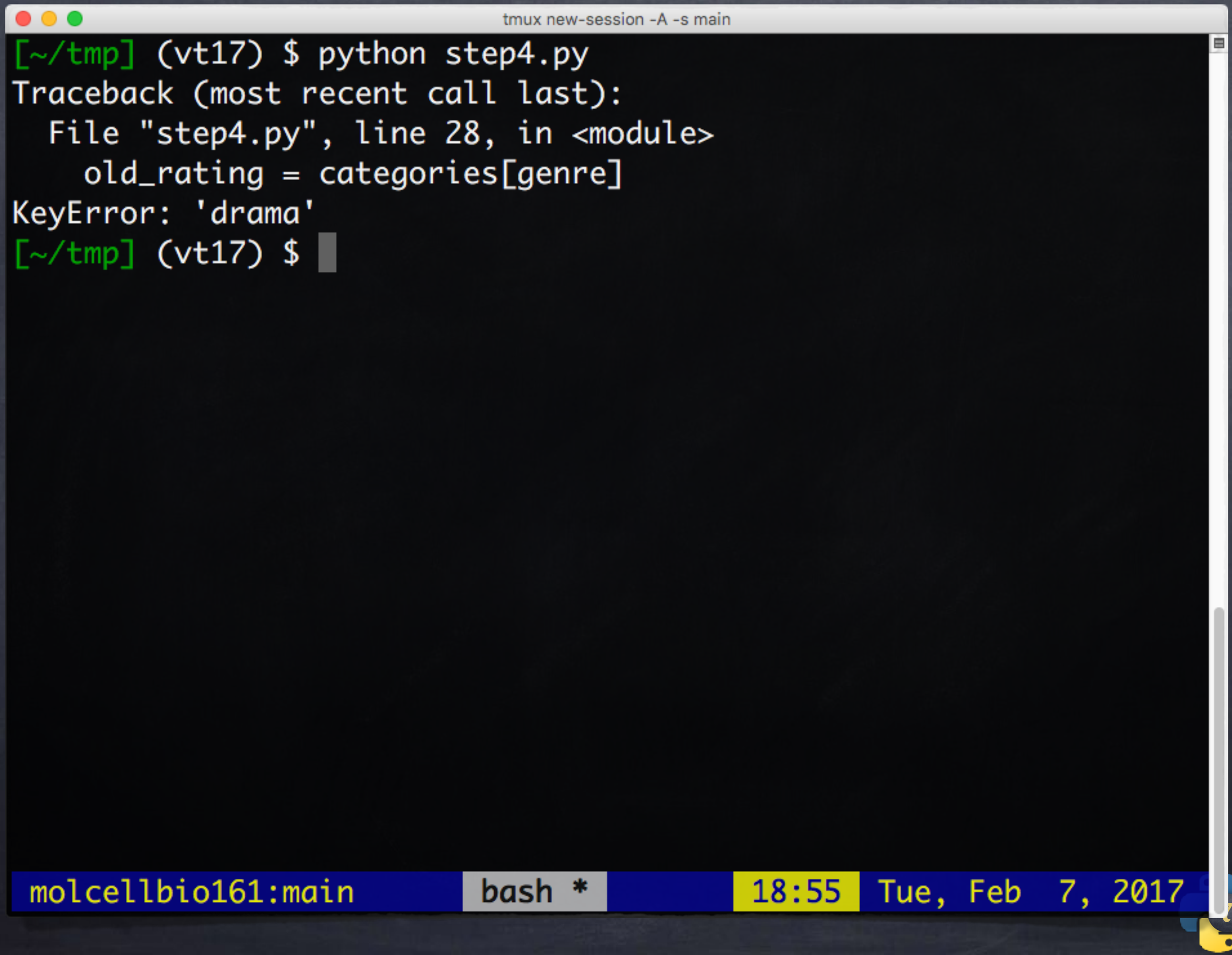
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ========================================
# 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 L1     (Python +2 MMM) 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[genre]

            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)
```

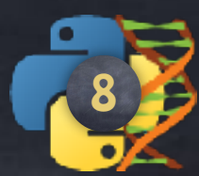
U:--- step4.py All L1 (Python +2 MNM) 19:22 1.20
Use +,-,0 for further adjustment


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) $
molcellbio161:main
bash *
18:55
Tue, Feb 7, 2017

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 MNM) 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) $
```

molcellbio161:main bash * 19:03 Tue, Feb 7, 2017

step6.py

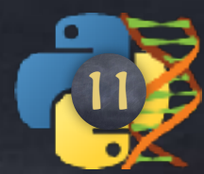
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ========================================
# 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:
            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)
```

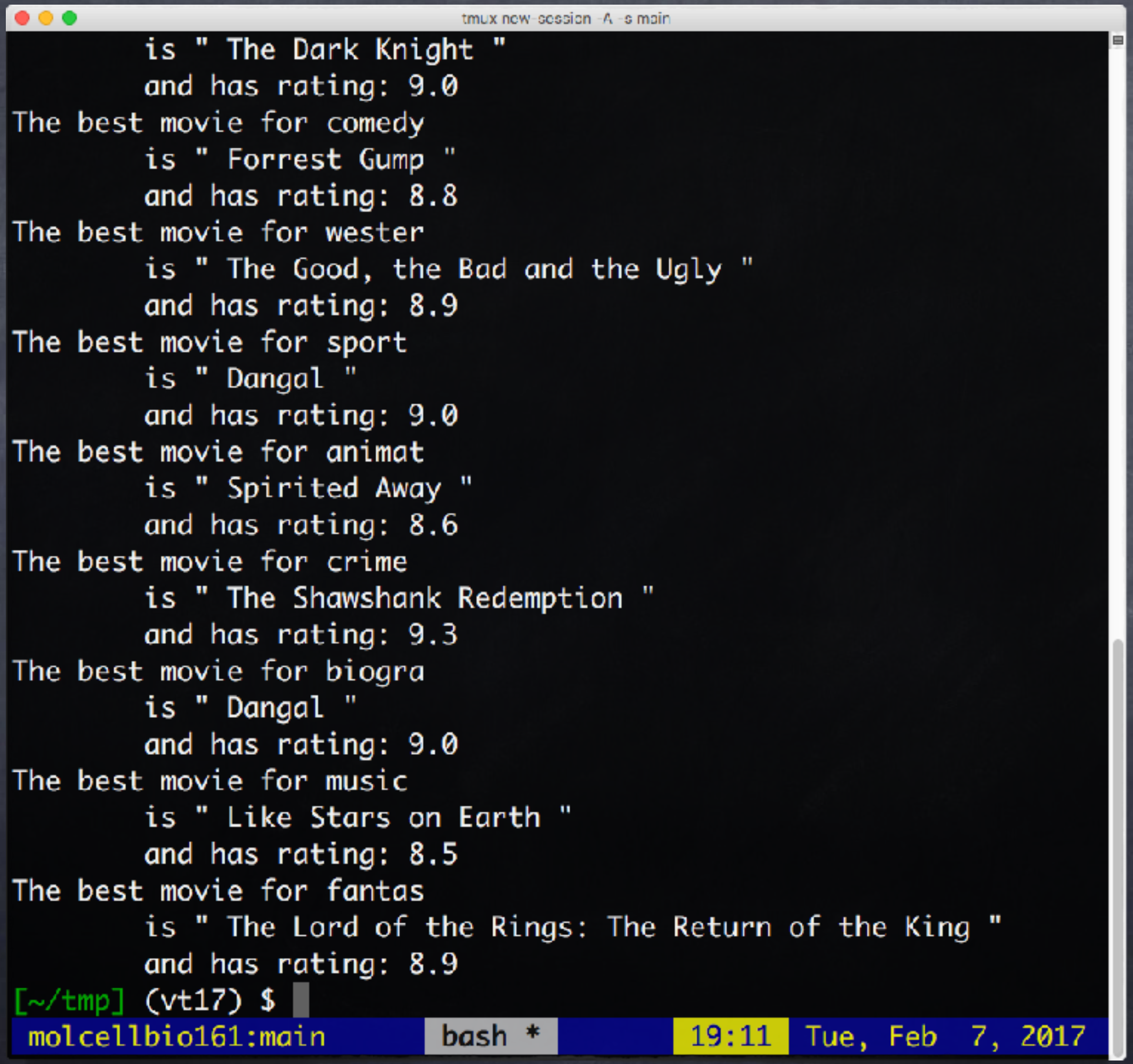
U:--- step6.py All L1 (Python +2 NMN) 19:21 1.08
Use +,-,0 for further adjustment

step7.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,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 NMN) 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

step8.py

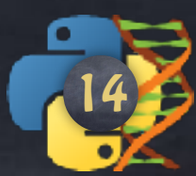
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# ========================================
# 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,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,'\n\tis "',title,'"\n\tand has rating:',rating)
```

U:--- step8.py All L12 (Python +2 NMN) 19:20 1.25

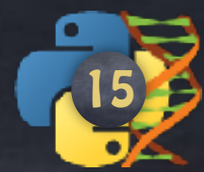
tmux new-session -A -s main

```
        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

> 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')
```

```python
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
```

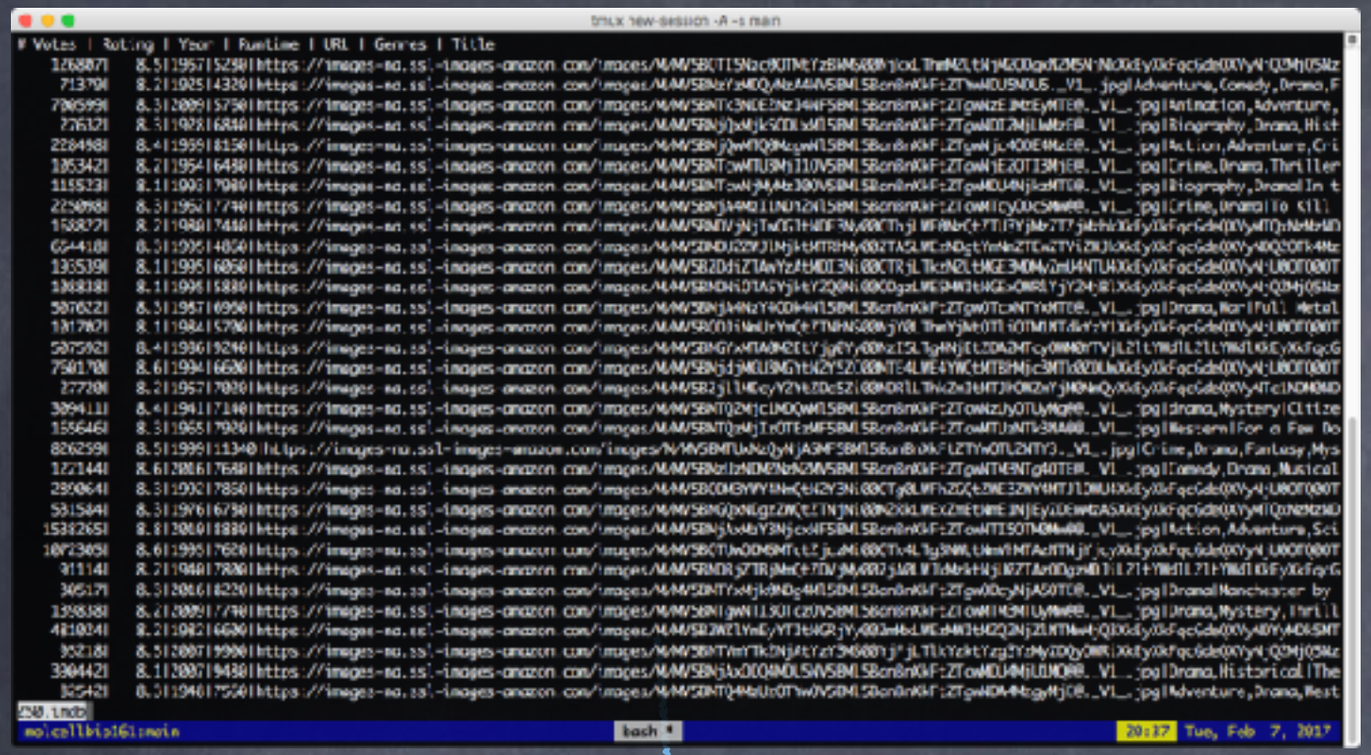
```python
with open('a', 'r') as f_input, open('b', 'w') as f_output:

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

```python
for line in f_input:
    if line.startswith('#'):
        continue

    # Do something with that line
```

step1.py

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

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

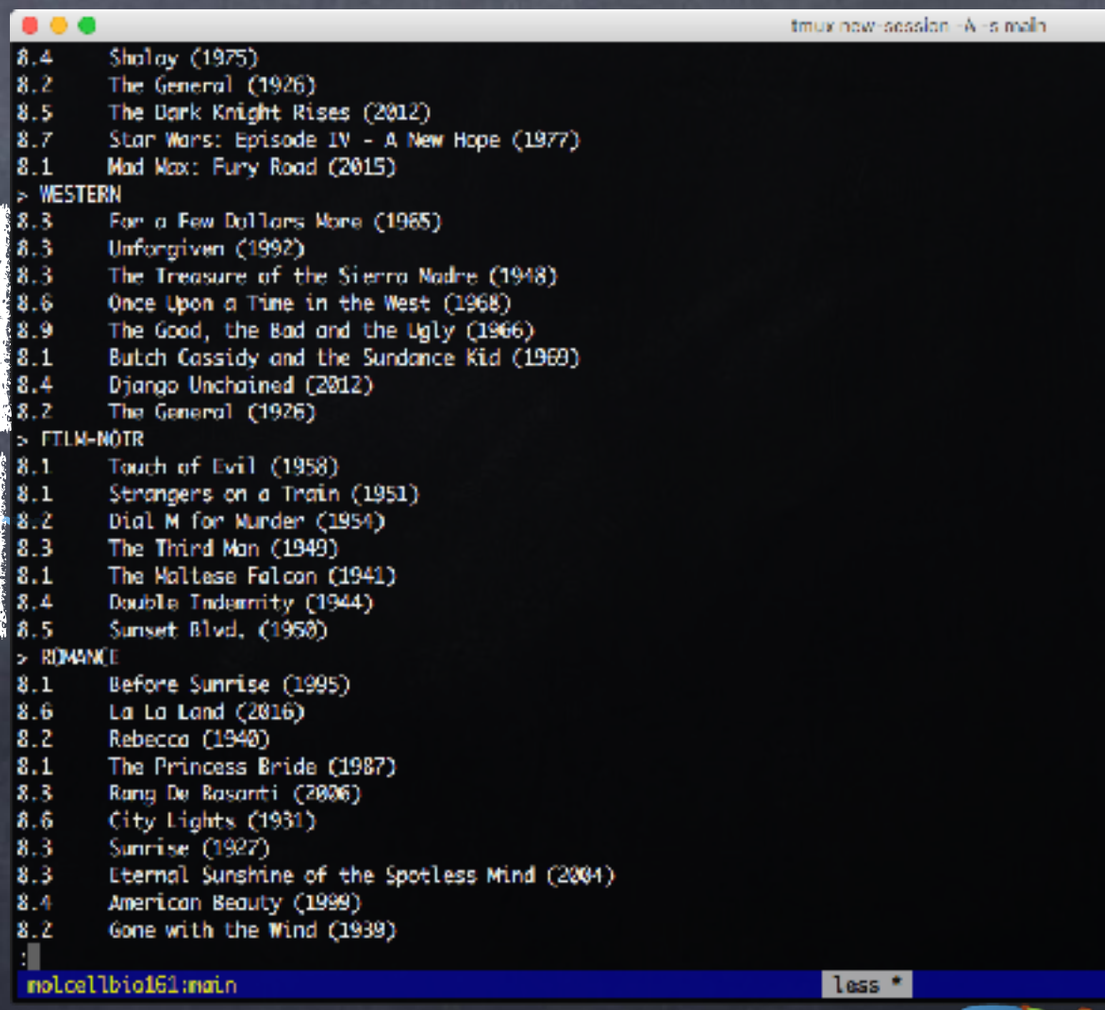
U:--- **step1.py** All L7 (Python +2 MMM) 21:22 1.69



Python

```
DRAMA -> [..., ..., ..., ...]
WESTERN -> [..., ..., ..., ...]
THRILLER -> [..., ..., ..., ...]
FILM-NOIR -> [..., ..., ..., ...]
HORROR -> [..., ..., ..., ...]
...
```

dict



step2.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')


    for line in f_input:

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

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

U:--- step2.py All L24 (Python +2 MMM) 21:46 1.68
Beginning of buffer

step3.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()             # who knows...
```

U:--- step3.py All L1 (Python +2 MMM) 21:47 1.76
Use +,-,0 for further adjustment

step4.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[genre]
```

U:--- step4.py All L40 (Python +1 MMM) 21:48 1.53
Use +,-,0 for further adjustment

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
```

U:--- step5.py All L43 (Python +2 MMM) 21:51 1.44
Beginning of buffer

step6.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(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')
```

U:--- step6.py All L52 (Python +2 MMM] 21:51 1.44
[No changes need to be saved]

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')
```


final.py

```python
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')
```


```python
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
movie  <-------- Name ; Year ; Rating ; Votes
```

Given 3 strings,
define a function that outputs them like
... \t ... (...)

1st string 2nd string 3rd string

Functions

```
def format_nicely(str1,str2,str3):

    ans = str1 + '\t' + str2 + ' (' + str3 +')'

    return ans
```

```python
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

```python
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

shell

```
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
```

```
[[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 NMM) 22:46 1.51

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

Use +,-,0 for further adjustment

```
def functionName(arg1, arg2, arg3):

    someValue = ... # Make it something

    # Some code with someValue
    # using arg1, arg2, arg3 (or not!)
    #
    # Some more code

    return someValue
```

```python
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

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

```python
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

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

* Positional arguments
* Keyword arguments

```
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
```

final.py

```
# ==========================

with open('250.imdb',
     open('output.txt'

    f_output.write('#
    f_output.write('#
    f_output.write('#


    # Main data struct
    categories = {}
    # Mapping: categor

    for line in f_inpu

        if line.starts
            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' + t

        for genre in genres: # uppe

            # Get the list of movie
            movies = categories.get
            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():

        # 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')
```

```
def format_category( category, movies):
    '''Formats the category and movies
       like we were told to do in class'''

    c = '> ' + category + '\n'
    m = '\n'.join(movies)

    return c + m
```

```
# 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')
```

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

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

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):
    """
```

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 MRN) 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 MRN) 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
```

/usr/local/bin/bash --noprofile — python

```
$ 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
Hello, how are you?
Good, and you?
**********
Hello, how are you?
Fine, thanks
**********
Hello, how are you?
great
**********
Hello, how are you?
Can't talk. Doing Python
>>>
```

```
$ 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
Hello, how are you?
Good, and you?
__name__ is set to extra
>>>
$ python extra.py
Hello, how are you?
Good, and you?
__name__ is set to __main__
$
```

```
def print_something(arg = 'great'):

    print('Hello, how are you?')
    print( arg )
    print("__name__ is set to", __name__

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

___name___

___name___

```
from ex
sep = '*' * 20
print(s
print_s
print(s
print_s
```

```
/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
>>>
>>> # no print
...
>>>
$ python extra.py
Hello, how are you?
Good, and you?
$
```

```
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?' )
```

```
from extra import print_something

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

    print(sep)
    print_something("Hello, how are you?", "Fine, thanks")

    print(sep)
    print_something("Hello, how are you?", "great")

    print(sep)
    print_something("Hello, how are you?", "Can't talk. Doing Python")

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

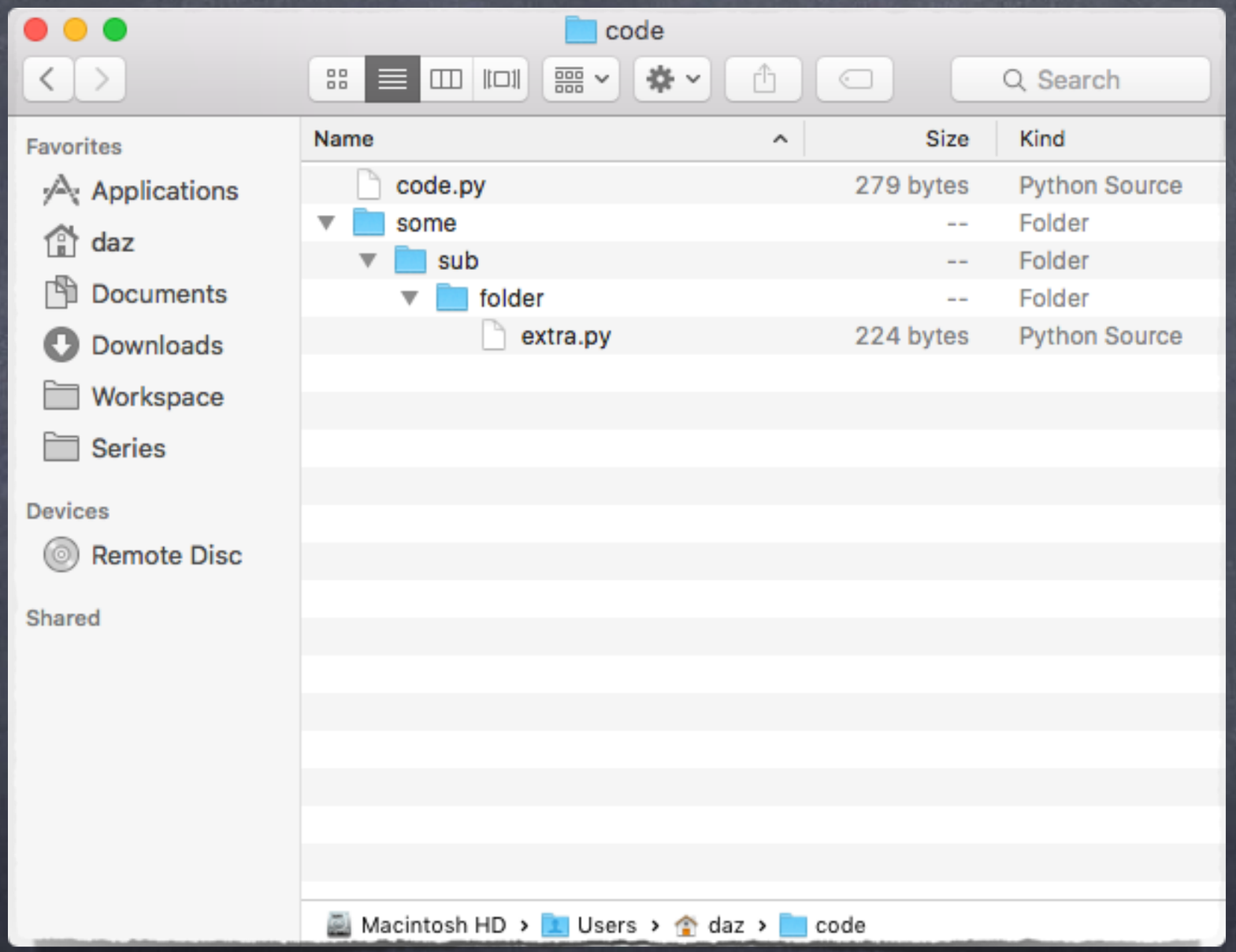
```
code.py        All L21    (Python +3 MMM) 02:35 1.64
Wrote /Users/daz/code.py
```

/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
$
```

```
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?' )
```

```
extra.py       All L14    (Python +3 MMM) 01:54
```


code
Search
Favorites
Applications
daz
Documents
Downloads
Workspace
Series
Devices
Remote Disc
Shared
Name
Size
Kind
code.py
279 bytes
Python Source
some
--
Folder
sub
--
Folder
folder
--
Folder
extra.py
224 bytes
Python Source
Macintosh HD
Users
daz
code

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
```

```python
import sys

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


if __name__ == '__main__':

    if len(sys.argv) > 1: # we received argume

        say_hello( sys.argv[1] ) # the first o

ay_hello()
```

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

/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 !
$
```