Functions as arguments

SI 507

Functions as arguments

Exercises:

- Filter() function
  - Section 1: Do it your own way
  - Section 2: Use filter() function
  - Section 3: Use filter() function with lambda
- Sort() function
- Map() function

Section 1: implement using what you know

- You are analyzing Tweets from Twitter users
- Q: Which Tweets mention Donald Trump's Twitter account (`"@realDonaldTrump"`) ? Output a list that contains **only** these Tweets
- Format: `[ (tweet_ID, tweet_source_user, text) … ]`

```
[
  (1, "this_is_alex", "I wonder what @realDonaldTrump thinks about this"),
  (151, "motivational_quotes", "inspirational quote for the day: love trumps hate"),
  (41, "i_heart_president_trump", "Thank you @realDonaldTrump for your amazing service
to this country!!!") ,
   …
]
```

Functions as arguments

- Some functions take **functions as arguments**
  - `sort()`
  - **`filter()`**
  - `map()`
  - `reduce()`

Section 2: implement, using filter() function

- You are analyzing Tweets from Twitter users
- Q: Which Tweets mention Donald Trump's Twitter account ("`@realDonaldTrump`") ?
- Format: `[ (tweet_ID, tweet_source_user, text) … ]`
- **Use `filter()`**

```
[
  (1, "this_is_alex", "I wonder what @realDonaldTrump thinks about this"),
  (151, "motivational_quotes", "inspirational quote for the day: love trumps hate"),
  (41, "i_heart_president_trump", "Thank you @realDonaldTrump for your amazing service
to this country!!!") ,
   …
]
```

Lambda Functions (syntax)

```
def add(x,y):
    return x + y
```

Can be translated to:

```
lambda x, y: x + y
```

Lambdas differ from normal Python methods because they can have only one expression, can't contain any statements and their return type is a `function` object. So the line of code above doesn't exactly return the value `x + y` but the function that calculates `x + y`.

Section 3: implement, using filter() and lambda function

- You are analyzing Tweets from Twitter users
- Q: Which Tweets mention Donald Trump's Twitter account ("`@realDonaldTrump`") ?
- Format: `[ (tweet_ID, tweet_source_user, text) … ]`
- Use `filter()` **with `lambda` function**

```
[
  (1, "this_is_alex", "I wonder what @realDonaldTrump thinks about this"),
  (151, "motivational_quotes", "inspirational quote for the day: love trumps hate"),
  (41, "i_heart_president_trump", "Thank you @realDonaldTrump for your amazing service to this country!!!") ,
   …
]
```

Functions as arguments (recap)

- Recap
  - Underlying concept: **Functions as arguments**
  - Make your code a bit "cleaner" and faster to write: **Lambda functions**
- Tips
  - When in doubt, write a regular function
  - If it will be re-used, write a regular function
  - Be aware of the function's inputs, outputs

Exercise 1 (sort)

- A student's grades on different subjects are given. The student would like help prioritizing which subject's homework to work on. Ideally, their worst subject should be worked on first, followed by their second subject, … . Their best subject should be worked on last
- Q: Reorder the list to match this ordering
- Format: [(subject_name, grade), …]
- Use sort()

[(‘English’, 88), (‘Science’, 90), (‘Maths’, 97), (‘Social Sciences’, 82), …]

Exercise 2 (map)

- U of M departments within LSA are given. We'd like to get their acronyms (e.g. "School of Information" → "SI")
- Q: Convert this list to a list of acronyms
- Format: [department_name, …]
- Use map()

["School of Information", "Museum of Anthropological Archaeology", "Center for the Study of Complex Systems", …]

Example from Work

```
In [836]: # Pandas DataFrame (basically, a table)
present
```

Out[836]:

topic	alt-right	antitheist	politics-left	politics-right	random
strategy					
dislike-recommendation	0.530726	0.570000	0.51	0.365000	1
no-channel	0.577957	0.510000	0.23	0.450000	1
not-interested	0.487805	0.612245	0.46	0.357143	1

```
In [837]: present = present.apply(lambda x: x.apply(lambda y: '{0:.0f}%'.format(y*100)))
```

```
In [838]: present
```

Out[838]:

topic	alt-right	antitheist	politics-left	politics-right	random
strategy					
dislike-recommendation	53%	57%	51%	36%	100%
no-channel	58%	51%	23%	45%	100%
not-interested	49%	61%	46%	36%	100%

"In the table, **apply** the following function to every row x:
in the row x, **apply** the following to every element y:
given the decimal y, format it to the percentage string version"

`ain_dict`: maps channel ID's to categories

Example from Work (2)

```
table['ain_category'] = table['channel_id'].apply(
    lambda x: ain_dict[x]
)
```

	bot_name	video_id	channel_id	timestep
20	alt-right_delete_0	9Xd8xq06FCw	UCCND6a0H56zHL4YuY226ZOQ	0
21	alt-right_delete_0	SzuZnh91U8E	UCmyjVwYZbp5YPYTUyeopO2g	0
22	alt-right_delete_0	jfKfPfyJRdk	UCSJ4gkVC6NrvII8umztf0Ow	0
23	alt-right_delete_0	2isYuQZMbdU	UCX6OQ3DkcsbYNE6H8uQQuVA	0
24	alt-right_delete_0	M4QUy02cQqY	UCpsN2TfWGmun4peN2IPgcKg	0
...	...	...	...	...
95903	alt-right_watch_4	SzuZnh91U8E	UCmyjVwYZbp5YPYTUyeopO2g	79
95904	alt-right_watch_4	Z8HJNypu_1Y	UCCB1oLQY3XM86ACD05Lq4HQ	79
95905	alt-right_watch_4	HxCcKzRAGWk	UCGr-rTYtP1m-r_-ncspdVQQ	79
95906	alt-right_watch_4	IlgtAzQhfOs	UC51KgmlyNO6uqxb68sjglXw	79
95907	alt-right_watch_4	2isYuQZMbdU	UCX6OQ3DkcsbYNE6H8uQQuVA	79

59761 rows × 4 columns

	bot_name	video_id	channel_id	timestep	ain_category
20	alt-right_delete_0	9Xd8xq06FCw	UCCND6a0H56zHL4YuY226ZOQ	0	other
21	alt-right_delete_0	SzuZnh91U8E	UCmyjVwYZbp5YPYTUyeopO2g	0	other
22	alt-right_delete_0	jfKfPfyJRdk	UCSJ4gkVC6NrvII8umztf0Ow	0	other
23	alt-right_delete_0	2isYuQZMbdU	UCX6OQ3DkcsbYNE6H8uQQuVA	0	other
24	alt-right_delete_0	M4QUy02cQqY	UCpsN2TfWGmun4peN2IPgcKg	0	other
...	...	...	...	...	...
95903	alt-right_watch_4	SzuZnh91U8E	UCmyjVwYZbp5YPYTUyeopO2g	79	other
95904	alt-right_watch_4	Z8HJNypu_1Y	UCCB1oLQY3XM86ACD05Lq4HQ	79	other
95905	alt-right_watch_4	HxCcKzRAGWk	UCGr-rTYtP1m-r_-ncspdVQQ	79	other
95906	alt-right_watch_4	IlgtAzQhfOs	UC51KgmlyNO6uqxb68sjglXw	79	other
95907	alt-right_watch_4	2isYuQZMbdU	UCX6OQ3DkcsbYNE6H8uQQuVA	79	other

59761 rows × 5 columns

Example from Work (2)

```
table.groupby('timestep')['ain_category'].apply(
    lambda x: percentages of each category in x
)
```

	bot_name	video_id	channel_id	timestep	ain_category
20	alt-right_delete_0	9Xd8xq06FCw	UCCND6a0H56zHL4YuY226ZOQ	0	other
21	alt-right_delete_0	SzuZnh91U8E	UCmyjVwYZbp5YPYTUyeopO2g	0	other
22	alt-right_delete_0	jfKfPfyJRdk	UCSJ4gkVC6NrvII8umztf0Ow	0	other
23	alt-right_delete_0	2isYuQZMbdU	UCX6OQ3DkcsbYNE6H8uQQuVA	0	other
24	alt-right_delete_0	M4QUy02cQqY	UCpsN2TfWGmun4peN2IPgcKg	0	other
...	...	...	...	...	...
95903	alt-right_watch_4	SzuZnh91U8E	UCmyjVwYZbp5YPYTUyeopO2g	79	other
95904	alt-right_watch_4	Z8HJNypu_1Y	UCCB1oLQY3XM86ACD05Lq4HQ	79	other
95905	alt-right_watch_4	HxCcKzRAGWk	UCGr-rTYtP1m-r_-ncspdVQQ	79	other
95906	alt-right_watch_4	IlgtAzQhfOs	UC51KgmlyNO6uqxb68sjglXw	79	other
95907	alt-right_watch_4	2isYuQZMbdU	UCX6OQ3DkcsbYNE6H8uQQuVA	79	other

59761 rows × 5 columns