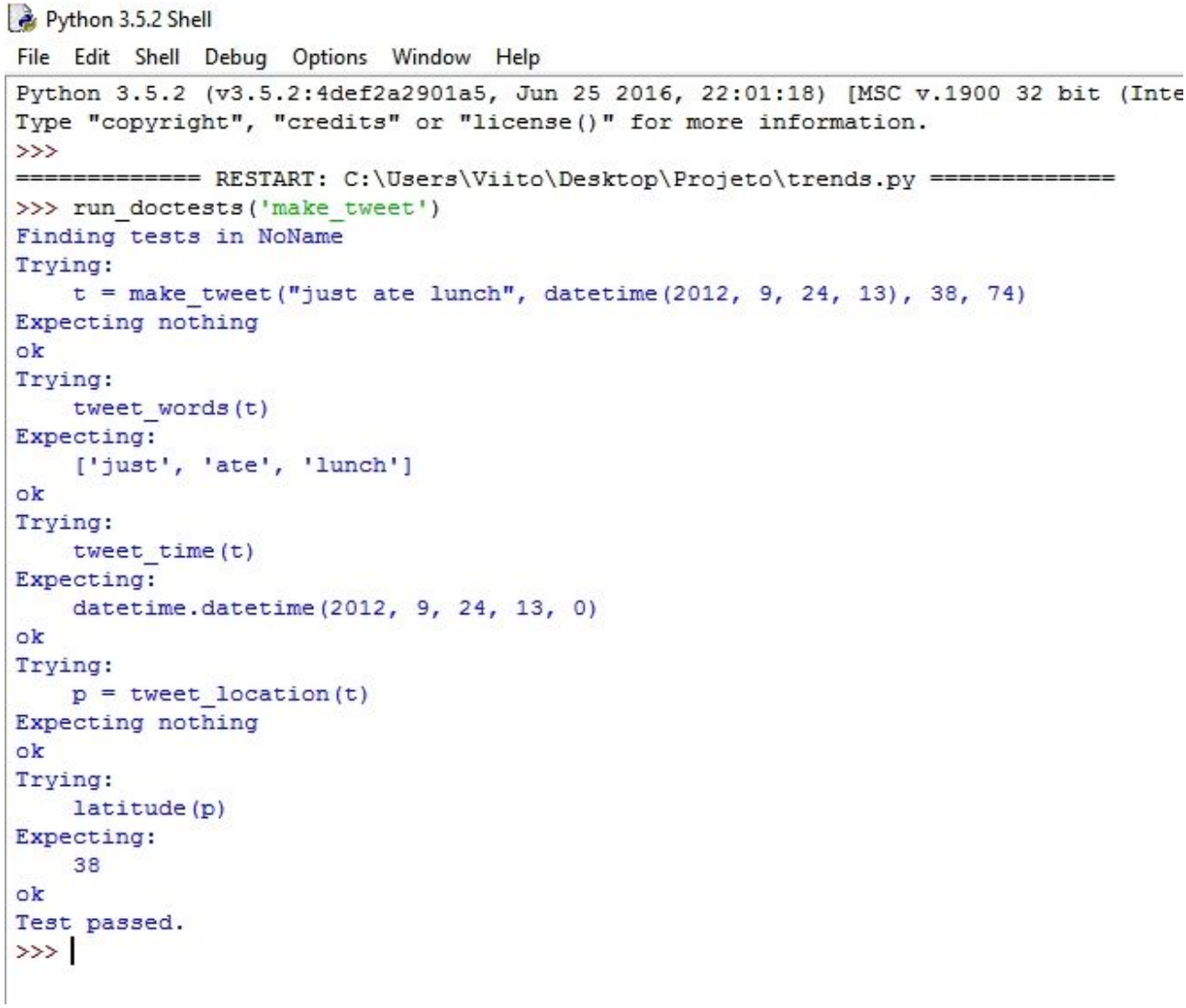
EXPLICAÇÃO DO PROJETO DE P1 - VITOR

Começando pelo doctest porque cada vez que você o rodar uma fada vai morrer e você vai chorar HAHUAHUAHUAHUUA

**************************PARA FAZER UM DOCTEST, COLOQUE SEU PROGRAMA PARA RODAR E DIGITE "run_doctest('nomedafunção')" no shell. Apenas algumas funções possuem um doctest. Elas aparecem, neste documento, no fim de cada problema, quando for possível testá-las de fato.********************************** Você deve esperar a seguinte resposta:**



Caso alguma o resultado dos testes não estejam de acordo com o esperado, o programa mostrará a resposta esperada e

a resposta obtida. Ainda assim ele retornará "Teste passed.", mas isso é um erro da função doctests. O teste só passará se todas as respostas esperadas retornarem "ok".

Começando o projeto - Fase 1

Existe a função "make_tweet(text, time, lat, lon)", ela recebe um tweet nesse formato:

("just ate...lunch", datetime(2012, 9, 24, 13), 38, 74)

devolve um dicionário nesse formato: {'text': "just ate...lunch", 'time': datetime.datetime(2012, 9, 24, 13, 0), 'latitude': 38, 'longitude': 74}

você NÃO deve mexer na função make_tweet nem se preocupar nas entradas que ela vai receber. Existem funções, dentro do trends.py, que fazem isso quando é necessário.

PROBLEMAS 1 E 2

Função "extract_words" - Recebe um texto e devolve um dicionário com apenas as palavras, descartando os caracteres especiais.

Exemplo:

Entrada:

"They're my...friends"

Saída:

['they', 're', 'my', 'friends']

PS: lembre-se que são quase 2 milhões de tweets, alguns até com caracteres do alfabeto chinês.

_

Função "tweet_words" - Recebe o dicionário da função make_tweet e retorna apenas o texto, sem nenhum caractere especial.

Exemplo:

Entrada:

{'text': “just ate...lunch”, 'time': datetime.datetime(2012, 9, 24, 13, 0), 'latitude': 38, 'longitude': 74}

Saída:

“[‘just’, ’ate’, ‘lunch’]”

PS: Para tirar a pontuação do texto na função tweet_words, você vai precisar usar a função “extract_words”.

_

Função “tweet_time” - Recebe um dicionário da função make_tweet e retorna o datetime do tweet.

(para saber mais sobre datetime acesse o site: <https://docs.python.org/3.5/library/datetime.html>).

Exemplo:

Entrada:

{'text': “just ate...lunch”, 'time': datetime.datetime(2012, 9, 24, 13, 0), 'latitude': 38, 'longitude': 74}

Saída:

datetime.datetime(2012, 9, 24, 13, 0)

_

Função “tweet_location” - Recebe um dicionário da função make_tweet e retorna uma tupla com as coordenadas(latitude e longitude) do tweet.

Exemplo:

Entrada:

{'text': “just ate...lunch”, 'time': datetime.datetime(2012, 9, 24, 13, 0), 'latitude': 38, 'longitude': 74}

Saída:

(38,74)

_

Função “tweet_string” - Recebe um dicionário da função make_tweet e retorna uma string no formato: “‘texto do tweet’ @ (lat,lon)”

Exemplo:

Entrada:

{'text': “just ate...lunch”, 'time': datetime.datetime(2012, 9, 24, 13, 0), 'latitude': 38, 'longitude': 74}

Saída:

“‘just ate...lunch’ @ (38,74)”

PS: note que você vai precisar usar a função ‘tweet_location’ dentro da ‘tweet_string’.

Quando você completar os problemas 1 e 2, o doctest para make_tweet deve passar.

_

Problemas 3 e 4

*****Você pode usar qualquer representação que quiser para sentimentos (tupla, dicionário, etc.). Escolhi dicionário então vou continuar com dicionário.*****

Função “make_sentiment” - Recebe um valor e retorna um dicionário com onde a key é uma string qualquer (a minha foi “valor”) e o valor da key é o valor do sentimento (se esse valor estiver dentro dos limites imposto pela linha *“assert value is None or (value >= -1 and value <= 1), 'Illegal value'”* que, basicamente, checa se o valor é válido, não faz nada se for e retorna um erro se não for.

Exemplo:
Entrada:
0.2
Saída:
{'sua-string': 0.2}

_

Função “has_sentiment” - Recebe um dicionário criado pela função make_sentiment e retorna se determinado sentimento tem um valor ou não (expressao booleana(True ou False)).

Exemplo1:
Entrada:
{'sua-string': 0.2}
Saída:
True

Exemplo2:
Entrada:
{'sua-string': None}
Saída:
False

_

Função "sentiment_value" - Recebe um dicionário criado pela função make_sentiment e retorna o valor contido key SE existir um valor. Quem vai ficar responsável por saber se tem um valor ou não para retornar esse valor é a linha *"assert has_sentiment(s), 'No sentiment value'"* que já está na função.

Exemplos:

Entrada:

{'sua-string': 0.2}

Saída:

0.2

Entrada:

{'sua-string': None}

Saída:

Traceback (most recent call last):
File "<pyshell#6>", line 1, in <module> sentiment_value(make_sentiment(None))
File "C:\Users\Viito\Desktop\Projeto\trends.py", line 101, in sentiment_value assert has_sentiment(s), 'No sentiment value'
AssertionError: No sentiment value

_

Função "get_word_sentiment" - Essa função cria um sentimento pra determinada palavra, se essa palavra ainda não existir no banco de dados e tiver algum valor. Você não deve mexer nessa função.

Quando você acabar os problemas 3 e 4, devem passar os doctests para make_sentiment e get_word_sentiment.

_

Problema 5

Função "analyze_tweet_sentiment" - Recebe um dicionário criado pela função make_tweet e retorna o "make_sentiment"(ou seja, um dicionario contendo um valor de sentimento) da média de sentimento de todas as palavras do tweet('text' que está no dicionário recebido) **que possuem um sentimento.**

Exemplo:

Entrada:

{'latitude': 0, 'longitude': 0, 'time': None, 'text': 'i love my job. #winning'}

Saída:

{'valor': 0.29167}

PS: Você pode usar funções implementadas anteriormente para conseguir o resultado esperado.

FASE 2 - Geometria dos Mapas

Função "find_centroid" - Recebe uma lista de tuplas no formato (x,y), que representa as coordenadas de um polígono e retorna uma tupla com a latitude, longitude e a área da figura. Aqui você precisa implementar a fórmula da área, da coordenada X e da coordenada Y de um polígono. Todas as fórmulas são somatórios.

Site com as 3 formulas e sua aplicação pratica:

<http://dan-scientia.blogspot.com.br/2009/10/centroide-de-um-poligono.html>

Exemplo:

Entrada:

[(1,2),(3,4),(5,0)(1,2)]

Saída:

(3.0, 2.0, 6.0)

PS: Na saída, os dois primeiros valores da tupla são os centróides do polígono(o primeiro a latitude e o segundo a longitude). E o terceiro é a área.

Quando você completar esse problema, o doctest para find_centroid deve passar.

Função "find_center" - Recebe uma lista com um ou mais polígonos, também no formato (x,y) e retorna a latitude e longitude do centro do polígono. O centro de uma forma é a média ponderada dos centróides de seus polígonos componentes, ponderados por sua área, veja o exemplo abaixo:

No caso de dois polígonos:

P1=[(1,2),(3,4),(5,0),(1,2)]
P2=[(5,6),(7,8),(9,0),(5,6)]

Seus centróides seriam:

CentroideP1=(3.0, 2.0, 6.0)
CentroideP2= (7.0, 4.6, 10.0)

A latitude e a longitude do seu centro seria:

LatCentro= ((3.0*6)+(7.0*10))/(6+10)=5.5
LongiCentro=((2.0*6)+(4.6*10))/(6+10)=3.6

Exemplo:

Entrada:

[[(18.948267, -155.634835), (19.035898, -155.881297), (19.123529, -155.919636), (19.348084, -155.886774), (19.73147, -156.062036), (19.857439, -155.925113), (20.032702, -155.826528), (20.147717, -155.897728), (20.26821, -155.87582), (20.12581, -155.596496), (20.021748, -155.284311), (19.868393, -155.092618), (19.736947, -155.092618), (19.523346, -154.807817), (19.348084, -154.983079), (19.26593, -155.295265), (19.134483, -155.514342), (18.948267, -155.634835)], [(21.029505, -156.587823), (20.892581, -156.472807), (20.952827, -156.324929), (20.793996, -156.00179), (20.651596, -156.051082), (20.580396, -156.379699), (20.60778, -156.445422), (20.783042, -156.461853), (20.821381, -156.631638), (20.919966, -156.697361), (21.029505, -156.587823)]]

Saída:

(19.777279178568115, -155.645005454296)

PS: Nos casos onde você recebe apenas um polígono no find_center, você pode retornar o "find_centroid" desse polígono.

Quando você completar esse problema, o doctest para find_center deve passar.

No fim da fase 2, a função draw_centered_map('SiglaEstado', n=X) desenhará os X estados mais próximos de um dado estado, incluindo ele mesmo. Exemplo: "draw_centeres_map('TX', n=10)" desenhará 10 estados.

FASE 3 - O Humor da Nação

Problema 8

Função "find_closest_state" - Recebe um dicionário criado pela função "make_tweet" e uma lista com as coordenadas de todos os estados dos EUA. Retorna a sigla de duas letras do estado mais próximo ao lugar que o tweet foi postado. Faça um código que veja qual dos estados tem uma coordenada mais próxima da coordenada do tweet e imprima o nome dele.

Exemplo:

Entrada:

({'time': None, 'latitude': 38, 'longitude': -122, 'text': 'Welcome to San Francisco'}**,** LISTADEESTADOS**)**

Saída:

"'CA'"

(Repare que as coordenadas do tweet são 38 e -122. As da CA, que estão na LISTADEESTADOS, são 37 e -120)

Quando terminar este problema, os doctests para find_closest_state devem passar.

–

Problema 9

Função "group_tweets_by_state"- Recebe uma lista de tweets(em dicionarios, do make_tweet) e procura a localização de todos os tweets na lista

e os coloca em um dicionário que contém as siglas de todos os estados dos EUA como chave.

Exemplo1:

Recebe uma lista com 4 tweets>

[tweet1, tweet2, tweet3, tweet4].

Acha a localização dos 4>

localização T1= CA (Califórnia)

localização T2= HI (Havaí)

localização T3= CA (Califórnia)

localização T4= CA (Califórnia)

Coloca os tweets em seus determinados estados no dicionário CRIADO PELA FUNÇÃO>

DICIO_TWEETS_ESTADO[‘CA’]=[tweet1,tweet3,tweet4]

DICIO_TWEETS_ESTADO[‘HI’]=[tweet2]

Exemplo2:

Entrada:

[{'time': None, 'latitude': 38, 'longitude': -122, 'text': 'Welcome to San Francisco'}**,**{'longitude': -74, 'text': 'Welcome to New York', 'latitude': 41, 'time': None}**]**

Saída:

{'NJ': [{'longitude': -74, 'text': 'Welcome to New York', 'latitude': 41, 'time': None}], 'TX': [], 'RI': [], 'NE': [], 'IA': [], 'NC': [], 'NY': [], 'ID': [], 'KY': [], 'HI': [], 'DE': [], 'MI': [], 'ND': [], 'VT': [], 'WY': [], 'TN': [], 'WI': [], 'MA': [], 'CT': [], 'LA': [], 'MN': [], 'ME': [], 'AZ': [], 'DC': [], 'MD': [], 'OK': [], 'UT': [], 'PA': [], 'VA': [], 'WV': [], 'AK': [], 'IL': [], 'AL': [], 'CA': [{'longitude': -122, 'text': 'Welcome to San Francisco', 'latitude': 38, 'time': None}], 'NM': [], 'MS': [], 'MT': [], 'NV': [], 'NH': [], 'IN': [], 'KS': [], 'SD': [], 'OH': [], 'CO': [], 'MO': [], 'PR': [], 'WA': [], 'AR': [], 'FL': [], 'OR': [], 'GA': [], 'SC': []}

Quando você completar esse problema, o doctests para group_tweets_by_state deve passar.

_

Problema 10

Função “most_talkative_state” - Recebe uma string (um termo) e retorna a sigla do estado que mais falou esse termo. O programa vai se responsabilizar por chamar uma lista contendo todos os tweets que contém a palavra recebida pela função e uma lista contendo todos os estados e suas coordenadas. As listas se chamarão “tweets” e “us_centers”.

Exemplo:
Entrada:
texas
Saída:
'TX'

Quando completar esse problema, os doctests de most_talkative_state devem passar.

_

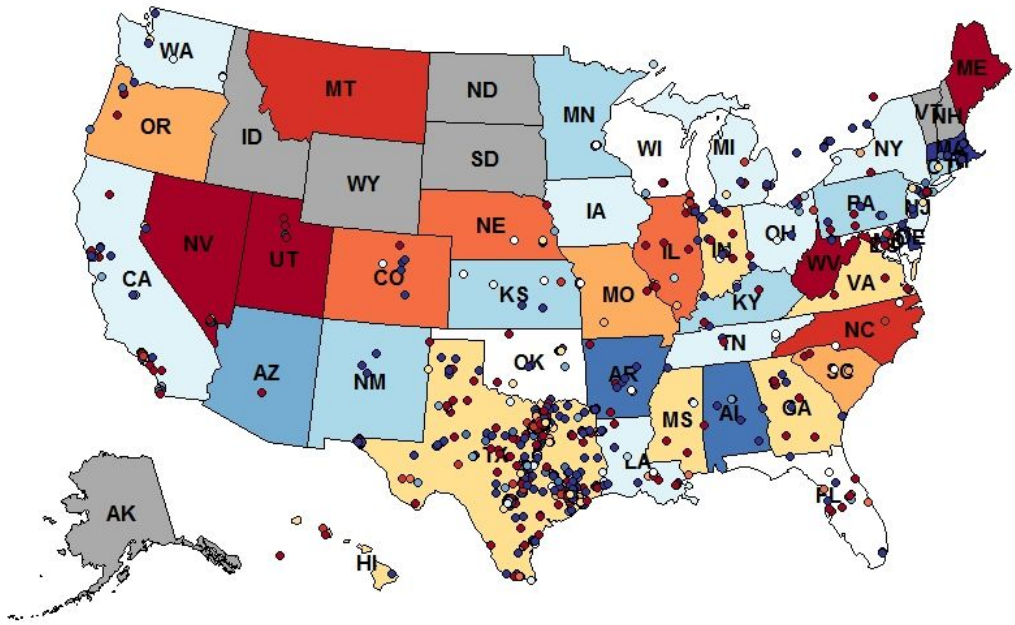
Problema 11

Função "average_sentiments" - Recebe um dicionário com todos os estados dos EUA(a sigla), cada um como uma chave, e o valor das chaves são os tweets feitos no respectivo estado (É o dicionário criado por group_tweets_by_state). Devolve outro dicionário parecido com o que recebe, mas o valor das chaves, em vez de ser tweets, será a média dos valores dos sentimentos de todos os tweets do respectivo estado.

Exemplo:
Entrada:
{'NJ': [{'longitude': -74, 'text': 'I love to New York', 'latitude': 41, 'time': None}], 'TX': [], 'RI': [], 'NE': [], 'IA': [], 'NC': [], 'NY': [], 'ID': [], 'KY': [], 'HI': [], 'DE': [], 'MI': [], 'ND': [], 'VT': [], 'WY': [], 'TN': [], 'WI': [], 'MA': [], 'CT': [], 'LA': [], 'MN': [], 'ME': [], 'AZ': [], 'DC': [], 'MD': [], 'OK': [], 'UT': [], 'PA': [], 'VA': [], 'WV': [], 'AK': [], 'IL': [], 'AL': [], 'CA': [{'longitude': -122, 'text': 'I hate to San Francisco', 'latitude': 38, 'time': None}], 'NM': [], 'MS': [], 'MT': [], 'NV': [], 'NH': [], 'IN': [], 'KS': [], 'SD': [], 'OH': [], 'CO': [], 'MO': [], 'PR': [], 'WA': [], 'AR': [], 'FL': [], 'OR': [], 'GA': [], 'SC': []}
Saída:
{'CA': -0.25, 'NJ': 0.625}

PS: Você precisará usar funções implementadas anteriormente como "sentiment_value" para resolver esse passo.

A partir desse ponto você já pode desenhar mapa coloridos por termo usando a função "draw_map_for_term('TERMOAQUI**')"**
Exemplo: "draw_map_for_term('texas')" desenhará um mapa colorido de acordo com o sentimento das pessoas a respeito do Texas por estado dos EUA.



Fase 4 - Entrando na Quarta Dimensão

Problema 12

Função "group_tweets_by_hour" - Recebe uma lista de dicionários, cada dicionário contém um tweet criado pela função "make_tweet". Devolve um dicionário com 24 keys, cada uma representando uma hora do dia, o valor da key deve ser todos os tweets da lista que foram postados na respectiva hora.

Exemplo:

Entrada:

[{'time': datetime.datetime(2014, 9, 12, 7, 0), 'longitude': 74, 'text': 'Teste para explicação', 'latitude': 38} **,** {'time': datetime.datetime(2013, 7, 25, 13, 0), 'longitude': 74, 'text': 'Explicação para teste', 'latitude': 38}**]**

Saída:

{0: [], 1: [], 2: [], 3: [], 4: [], 5: [], 6: [], 7: [{'time': datetime.datetime(2014, 9, 12, 7, 0), 'longitude': 74, 'text': 'Teste para explicação', 'latitude': 38}], 8: [], 9: [], 10: [], 11: [], 12: [], 13: [{'time': datetime.datetime(2013, 7, 25, 13, 0), 'longitude': 74, 'text': 'Explicação para teste', 'latitude': 38}], 14: [], 15: [], 16: [], 17: [], 18: [], 19: [], 20: [], 21: [], 22: [], 23: []}

A partir desse ponto você já pode desenhar mapa coloridos por termo usando a função "draw_map_by_hour(‘TERMOAQUI’**, pause=**n’**)", a função recebe dois parâmetros, o termo e a pausa, que seria o intervalo de tempo, em segundos e float, que o gráfico com o sentimento das pessoas a respeito do termo naquela hora ficará na tela. Exemplo: draw_map_by_hour(‘texas’, pause=1.0).**

Boa sorte!

hihi!