1. The brief

Imagine working for a digital marketing agency, and the agency is approached by a massive online retailer of furniture. They want to test our skills at creating large campaigns for all of their website. We are tasked with creating a prototype set of keywords for search campaigns for their sofas section. The client says that they want us to generate keywords for the following products:

- sofas
- convertible sofas
- love seats
- recliners
- sofa beds

The brief: The client is generally a low-cost retailer, offering many promotions and discounts. We will need to focus on such keywords. We will also need to move away from luxury keywords and topics, as we are targeting price-sensitive customers. Because we are going to be tight on budget, it would be good to focus on a tightly targeted set of keywords and make sure they are all set to exact and phrase match.

Based on the brief above we will first need to generate a list of words, that together with the products given above would make for good keywords. Here are some examples:

- Products: sofas, recliners
- Words: buy, prices

The resulting keywords: 'buy sofas', 'sofas buy', 'buy recliners', 'recliners buy', 'prices sofas', 'sofas prices', 'prices recliners', 'recliners prices'.

As a final result, we want to have a DataFrame that looks like this:

Campaign	Ad Group	Keyword	Criterion Type
Campaign1	AdGroup_1	keyword 1a	Exact
Campaign1	AdGroup_1	keyword 1a	Phrase
Campaign1	AdGroup_1	keyword 1b	Exact
Campaign1	AdGroup_1	keyword 1b	Phrase
Campaign1	AdGroup_2	keyword 2a	Exact
Campaign1	AdGroup_2	keyword 2a	Phrase

The first step is to come up with a list of words that users might use to express their desire in buying low-cost sofas.

In [366]:
```
# List of words to pair with products
words = ['buy', 'price', 'discount', 'promotion', 'promo', 'shop']

# Print list of words
print(words)
```

```
['buy', 'price', 'discount', 'promotion', 'promo', 'shop']
```

In [367]:
```
%%nose

def test_words_in_long_word_list_task_1():
    correct_words = ['buy', 'price', 'discount', 'promotion', 'promo',
 'shop',
                     'buying', 'prices', 'pricing', 'shopping', 'discoun
ts',
                     'promos', 'ecommerce', 'e commerce', 'buy online',
                     'shop online', 'cheap', 'best price', 'lowest pric
e',
                     'cheapest', 'best value', 'offer', 'offers', 'promo
tions',
                     'purchase', 'sale', 'bargain', 'affordable',
                     'cheap', 'low cost', 'low price', 'budget', 'inexpe
nsive', 'economical',]
    assert all([word in correct_words for word in words]), \
    'The variable `words` should contain relevant words to the products
 in the brief.'

def test_len_words_task_1():
    assert 6 <= len(set(words)) <= 10, \
    "There should be six to ten brief-related words in the list `words
`."
```

Out[367]: 2/2 tests passed

2. Combine the words with the product names

Imagining all the possible combinations of keywords can be stressful! But not for us, because we are keyword ninjas! We know how to translate campaign briefs into Python data structures and can imagine the resulting DataFrames that we need to create.

Now that we have brainstormed the words that work well with the brief that we received, it is now time to combine them with the product names to generate meaningful search keywords. We want to combine every word with every product once before, and once after, as seen in the example above.

As a quick reminder, for the product 'recliners' and the words 'buy' and 'price' for example, we would want to generate the following combinations:

buy recliners
recliners buy
price recliners
recliners price
…

and so on for all the words and products that we have.

In [368]:
```
products = ['sofas', 'convertible sofas', 'love seats', 'recliners', 'so
fa beds']

# Create an empty list
keywords_list = []

# Loop through products
for product in products:
    # Loop through words
    for word in words:
        # Append combinations
        keywords_list.append([product, product + ' ' + word])
        keywords_list.append([product, word + ' ' + product])

# Inspect keyword list
from pprint import pprint
pprint(keywords_list)
```

```
[['sofas', 'sofas buy'],
 ['sofas', 'buy sofas'],
 ['sofas', 'sofas price'],
 ['sofas', 'price sofas'],
 ['sofas', 'sofas discount'],
 ['sofas', 'discount sofas'],
 ['sofas', 'sofas promotion'],
 ['sofas', 'promotion sofas'],
 ['sofas', 'sofas promo'],
 ['sofas', 'promo sofas'],
 ['sofas', 'sofas shop'],
 ['sofas', 'shop sofas'],
 ['convertible sofas', 'convertible sofas buy'],
 ['convertible sofas', 'buy convertible sofas'],
 ['convertible sofas', 'convertible sofas price'],
 ['convertible sofas', 'price convertible sofas'],
 ['convertible sofas', 'convertible sofas discount'],
 ['convertible sofas', 'discount convertible sofas'],
 ['convertible sofas', 'convertible sofas promotion'],
 ['convertible sofas', 'promotion convertible sofas'],
 ['convertible sofas', 'convertible sofas promo'],
 ['convertible sofas', 'promo convertible sofas'],
 ['convertible sofas', 'convertible sofas shop'],
 ['convertible sofas', 'shop convertible sofas'],
 ['love seats', 'love seats buy'],
 ['love seats', 'buy love seats'],
 ['love seats', 'love seats price'],
 ['love seats', 'price love seats'],
 ['love seats', 'love seats discount'],
 ['love seats', 'discount love seats'],
 ['love seats', 'love seats promotion'],
 ['love seats', 'promotion love seats'],
 ['love seats', 'love seats promo'],
 ['love seats', 'promo love seats'],
 ['love seats', 'love seats shop'],
 ['love seats', 'shop love seats'],
 ['recliners', 'recliners buy'],
 ['recliners', 'buy recliners'],
 ['recliners', 'recliners price'],
 ['recliners', 'price recliners'],
 ['recliners', 'recliners discount'],
 ['recliners', 'discount recliners'],
 ['recliners', 'recliners promotion'],
 ['recliners', 'promotion recliners'],
 ['recliners', 'recliners promo'],
 ['recliners', 'promo recliners'],
 ['recliners', 'recliners shop'],
 ['recliners', 'shop recliners'],
 ['sofa beds', 'sofa beds buy'],
 ['sofa beds', 'buy sofa beds'],
 ['sofa beds', 'sofa beds price'],
 ['sofa beds', 'price sofa beds'],
 ['sofa beds', 'sofa beds discount'],
 ['sofa beds', 'discount sofa beds'],
 ['sofa beds', 'sofa beds promotion'],
 ['sofa beds', 'promotion sofa beds'],
 ['sofa beds', 'sofa beds promo'],
```

```
 ['sofa beds', 'promo sofa beds'],
 ['sofa beds', 'sofa beds shop'],
 ['sofa beds', 'shop sofa beds']]
```

In [369]:
```
%%nose

def test_list_task_2():
    assert isinstance(keywords_list, list), 'The variable `keywords_list
` is not a Python list.'

def test_keywords_list_created_correctly_task_2():
    test_keywords_list = []

    for product in products:
        for word in words:
            test_keywords_list.append([product, word + ' ' + product])
            test_keywords_list.append([product, product + ' ' + word])

    assert all([kl in test_keywords_list for kl in keywords_list]), \
     'Make sure you have \'product word\' or \'word product\' as the sec
ond element in each sublist.'
```

Out[369]:
```
2/2 tests passed
```

3. Convert the list of lists into a DataFrame

Now we want to convert this list of lists into a DataFrame so we can easily manipulate it and manage the final output.

In [370]:
```
# Load library
import pandas as pd

# Create a DataFrame from list
keywords_df = pd.DataFrame.from_records(keywords_list)

# Print the keywords DataFrame to explore it
print(keywords_df)
```

	0	1
0	sofas	sofas buy
1	sofas	buy sofas
2	sofas	sofas price
3	sofas	price sofas
4	sofas	sofas discount
5	sofas	discount sofas
6	sofas	sofas promotion
7	sofas	promotion sofas
8	sofas	sofas promo
9	sofas	promo sofas
10	sofas	sofas shop
11	sofas	shop sofas
12	convertible sofas	convertible sofas buy
13	convertible sofas	buy convertible sofas
14	convertible sofas	convertible sofas price
15	convertible sofas	price convertible sofas
16	convertible sofas	convertible sofas discount
17	convertible sofas	discount convertible sofas
18	convertible sofas	convertible sofas promotion
19	convertible sofas	promotion convertible sofas
20	convertible sofas	convertible sofas promo
21	convertible sofas	promo convertible sofas
22	convertible sofas	convertible sofas shop
23	convertible sofas	shop convertible sofas
24	love seats	love seats buy
25	love seats	buy love seats
26	love seats	love seats price
27	love seats	price love seats
28	love seats	love seats discount
29	love seats	discount love seats
30	love seats	love seats promotion
31	love seats	promotion love seats
32	love seats	love seats promo
33	love seats	promo love seats
34	love seats	love seats shop
35	love seats	shop love seats
36	recliners	recliners buy
37	recliners	buy recliners
38	recliners	recliners price
39	recliners	price recliners
40	recliners	recliners discount
41	recliners	discount recliners
42	recliners	recliners promotion
43	recliners	promotion recliners
44	recliners	recliners promo
45	recliners	promo recliners
46	recliners	recliners shop
47	recliners	shop recliners
48	sofa beds	sofa beds buy
49	sofa beds	buy sofa beds
50	sofa beds	sofa beds price
51	sofa beds	price sofa beds
52	sofa beds	sofa beds discount
53	sofa beds	discount sofa beds
54	sofa beds	sofa beds promotion
55	sofa beds	promotion sofa beds

```
56          sofa beds              sofa beds promo
57          sofa beds             promo sofa beds
58          sofa beds              sofa beds shop
59          sofa beds              shop sofa beds
```

In [371]:
```
%%nose

def test_pandas_loaded_task_3():
    assert 'pd' in globals(), \
    'Did you forget to import pandas aliased as pd?'

def test_keywords_df_created_correctly_task_3():
    import pandas as pd
    correct_keywords_df = pd.DataFrame.from_records(keywords_list)
    assert correct_keywords_df.equals(keywords_df), "The contents of `ke
ywords_df` doesn't appear to be correct. Don't specify column names ye
t!"

#     correct_keywords_df = pd.DataFrame.from_records(keywords_list)

#     assert (correct_keywords_df
#             .sort_values(list(sorted(correct_keywords_df.columns.value
s)))
#             .reset_index(drop=True)
#             .equals(keywords_df
#                     .sort_values(list(sorted(keywords_df.columns.value
s)))
#                     .reset_index(drop=True))) , \
#     'The DataFrame you created doesn\'t seem to be the right one.'
```

Out[371]: 2/2 tests passed

4. Rename the columns of the DataFrame

Before we can upload this table of keywords, we will need to give the columns meaningful names. If we inspect the DataFrame we just created above, we can see that the columns are currently named `0` and `1`. `Ad Group` (example: "sofas") and `Keyword` (example: "sofas buy") are much more appropriate names.

In [372]:
```
# Rename the columns of the DataFrame
keywords_df = keywords_df.rename(columns={0: 'Ad Group', 1: 'Keyword'})
```

In [373]:
```
%%nose

def test_df_columns_renamed_correctly_task_4():
    assert 'Ad Group' in keywords_df.columns.values and 'Keyword' in key
words_df.columns.values , \
    'Make sure you are using the proper names for the columns. Capitaliz
ation matters!'
```

Out[373]: 1/1 tests passed

5. Add a campaign column

Now we need to add some additional information to our DataFrame. We need a new column called `Campaign` for the campaign name. We want campaign names to be descriptive of our group of keywords and products, so let's call this campaign 'SEM_Sofas'.

In [374]:
```
# Add a campaign column
keywords_df['Campaign']='SEM_Sofas'
```

In [375]:
```
%%nose

def test_campaign_column_created_task_5():
    assert 'Campaign' in keywords_df.columns.values, \
    "`Campaign` needs to be the name of the new column. Capitalization m
atters!"

def test_campaign_name_task_5():
    assert len(set(keywords_df['Campaign'])) == 1 and set(keywords_df['C
ampaign']).pop() == 'SEM_Sofas', \
    "Is 'SEM_Sofas' the campaign name in every row of the `Campaign` col
umn?"
```

Out[375]:
```
2/2 tests passed
```

6. Create the match type column

There are different keyword match types. One is exact match, which is for matching the exact term or are close variations of that exact term. Another match type is broad match, which means ads may show on searches that include misspellings, synonyms, related searches, and other relevant variations.

Straight from Google's AdWords documentation (https://support.google.com/google-ads/answer/2497836?hl=en):

> In general, the broader the match type, the more traffic potential that keyword will have, since your ads may be triggered more often. Conversely, a narrower match type means that your ads may show less often—but when they do, they're likely to be more related to someone's search.

Since the client is tight on budget, we want to make sure all the keywords are in exact match at the beginning.

In [376]:
```
# Add a criterion type column
keywords_df['Criterion Type']='Exact'
```

In [377]:
```
%%nose

def test_criterion_type_column_created_task_6():
    assert 'Criterion Type' in keywords_df.columns.values, \
    "`Criterion Type` needs to be the name of the new column. Capitalization matters!"

def test_criterion_type_task_6():
    assert len(set(keywords_df['Criterion Type'])) == 1 and set(keywords_df['Criterion Type']).pop() == 'Exact', \
    "Is 'Exact' the campaign name in every row of the `Criterion Type` column?"
```

Out[377]:
```
2/2 tests passed
```

7. Duplicate all the keywords into 'phrase' match

The great thing about exact match is that it is very specific, and we can control the process very well. The tradeoff, however, is that:

1. The search volume for exact match is lower than other match types
2. We can't possibly think of all the ways in which people search, and so, we are probably missing out on some high-quality keywords.

So it's good to use another match called *phrase match* as a discovery mechanism to allow our ads to be triggered by keywords that include our exact match keywords, together with anything before (or after) them.

Later on, when we launch the campaign, we can explore with modified broad match, broad match, and negative match types, for better visibility and control of our campaigns.

In [378]:
```
# Make a copy of the keywords DataFrame
keywords_phrase = keywords_df.copy()

# Change criterion type match to phrase
keywords_phrase['Criterion Type'] = 'Phrase'

# Append the DataFrames
keywords_df_final = keywords_df.append(keywords_phrase)
```

In [379]:
```
%%nose

def test_criterion_type_task_7():
    assert len(set(keywords_phrase['Criterion Type'])) == 1 and set(keyw
ords_phrase['Criterion Type']).pop() == 'Phrase', \
    "Is 'Phrase' the campaign name in every row of the `Criterion Type`
 column in `keyword_phrase`?"

def test_phrase_df_created_task_7():
    test_keywords_df_final = keywords_df.append(keywords_phrase)
    assert test_keywords_df_final.equals(keywords_df_final), \
    'The final DataFrame does not seem to be created correctly.'
```

Out[379]:
```
2/2 tests passed
```

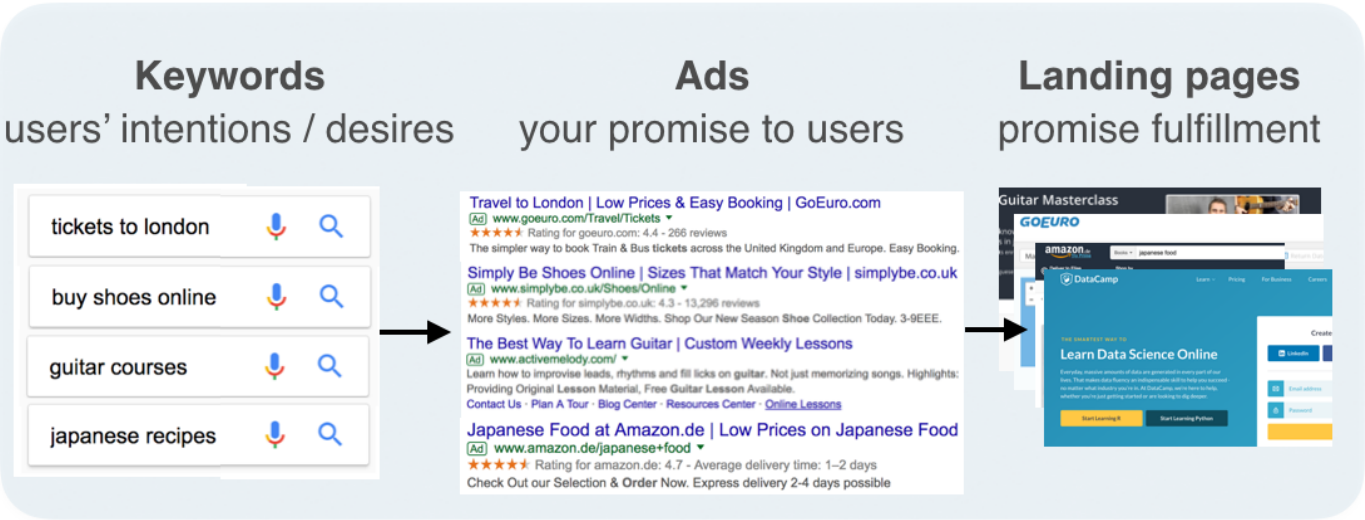
8. Save and summarize!

To upload our campaign, we need to save it as a CSV file. Then we will be able to import it to AdWords editor or BingAds editor. There is also the option of pasting the data into the editor if we want, but having easy access to the saved data is great so let's save to a CSV file!

Looking at a summary of our campaign structure is good now that we've wrapped up our keyword work. We can do that by grouping by ad group and criterion type and counting by keyword. This summary shows us that we assigned specific keywords to specific ad groups, which are each part of a campaign. In essence, we are telling Google (or Bing, etc.) that we want any of the words in each ad group to trigger one of the ads in the same ad group. Separately, we will have to create another table for ads, which is a task for another day and would look something like this:

Campaign	Ad Group	Headline 1	Headline 2	Description	Final URL
SEM_Sofas	Sofas	Looking for Quality Sofas?	Explore Our Massive Collection	30-day Returns With Free Delivery Within the US. Start Shopping Now	DataCampSofas.com/sofas
SEM_Sofas	Sofas	Looking for Affordable Sofas?	Check Out Our Weekly Offers	30-day Returns With Free Delivery Within the US. Start Shopping Now	DataCampSofas.com/sofas
SEM_Sofas	Recliners	Looking for Quality Recliners?	Explore Our Massive Collection	30-day Returns With Free Delivery Within the US. Start Shopping Now	DataCampSofas.com/recliners
SEM_Sofas	Recliners	Need Affordable Recliners?	Check Out Our Weekly Offers	30-day Returns With Free Delivery Within the US. Start Shopping Now	DataCampSofas.com/recliners

Together, these tables get us the sample **keywords -> ads -> landing pages** mapping shown in the diagram below.



In [380]:
```
# Save the final keywords to a CSV file
keywords_df_final.to_csv('keywords.csv', index=False)

# View a summary of our campaign work
summary = keywords_df_final.groupby(['Ad Group', 'Criterion Type'])['Key
word'].count()
print(summary)
```

```
Ad Group           Criterion Type
convertible sofas  Exact             12
                   Phrase            12
love seats         Exact             12
                   Phrase            12
recliners          Exact             12
                   Phrase            12
sofa beds          Exact             12
                   Phrase            12
sofas              Exact             12
                   Phrase            12
Name: Keyword, dtype: int64
```

In [381]:
```
%%nose
import os

# def test_df_saved_to_csv_task_8():
#     test_keywords_df = pd.read_csv('keywords.csv')
#     assert keywords_df_final.equals(test_keywords_df), \
#     'The \'keywords.csv\' file isn\'t saved correctly.'

def test_file_exists_task_8():
    assert os.path.exists("keywords.csv"), \
    'Did you save the `keywords_df_final` DataFrame to \'keywords.csv
\'?'

def test_index_excluded_task_8():
    test_keywords_csv = pd.read_csv('keywords.csv')
    assert len(pd.read_csv('keywords.csv').columns) == 4, \
    'Did you exclude the DataFrame index in \'keywords.csv\' using `inde
x=False`?'
```

Out[381]: 2/2 tests passed