1. Data Cleaning & Preprocess

Load data and initial scan

In [81]:

```
import pandas as pd
#import data
survey_data = pd.read_csv('survey_data.csv')
usage_data = pd.read_csv('survey_users_app_usage.csv')
df_survey = pd.DataFrame(survey_data)
df_usage = pd.DataFrame(usage_data)
```

In [82]:

```
#initial scan
df_survey.head()
```

Out[82]:

	user_id	age	annual_income	country	duolingo_platform	duolingo_subscriber	duolingo_usag
0	35c9fc6e72c911e99681dca9049399ef	18-34	26,000−75,000	JP	Android phone or tablet	No, I have never paid for Duolingo Plus	Dail
1	35c9fdde72c911e98630dca9049399ef	18-34	26,000−75,000	JP	iPhone or iPad	No, I have never paid for Duolingo Plus	Weekl
2	35c9feb072c911e9ab4cdca9049399ef	18-34	76,000−150,000	JP	iPhone or iPad	Yes, I currently pay for Duolingo Plus	Dail
3	35c9ff7072c911e9900ddca9049399ef	18-34	76,000−150,000	JP	iPhone or iPad	No, but I have previously paid for Duolingo Plus	Dail
4	35ca002672c911e99effdca9049399ef	35 - 54	76,000−150,000	JP	Android phone or tablet	Yes, I currently pay for Duolingo Plus	Dail

In [83]:

```
df_usage.head()
```

Out[83]:

	user_id	duolingo_start_date	daily_goal	highest_course_progress	took_placement_test	purcha
0	35cb7e8f72c911e9888edca9049399ef	1/10/22 21:14	NaN	46.0	True	
1	35ca34fd72c911e99ed6dca9049399ef	2/28/21 5:01	NaN	50.0	True	
2	35d1a54a72c911e98e25dca9049399ef	5/7/18 17:55	1.0	71.0	False	
3	35d4beb072c911e9aa92dca9049399ef	4/27/22 9:28	NaN	2.0	False	
4	35ccf4bd72c911e9be2edca9049399ef	4/9/19 3:16	NaN	34.0	False	

In [84]:

```
#check percentage of null
df_survey.isnull().mean() * 100
```

Out[84]:

```
user_id                                  0.000000
age                                      5.640860
annual_income                           16.243737
country                                  0.000000
duolingo_platform                        4.460967
duolingo_subscriber                      4.622596
duolingo_usage                           4.460967
employment_status                        7.386455
future_contact                          11.976725
gender                                   5.640860
other_resources                         27.687086
primary_language_commitment              2.666882
primary_language_review                  2.796186
primary_language_motivation              3.862938
primary_language_motivation_followup    39.954744
primary_language_proficiency             2.586068
student                                 10.732180
survey_complete                          0.000000
time_spent_seconds                       0.000000
dtype: float64
```

In [85]:

```
df_usage.drop('Unnamed: 12',inplace=True, axis=1)
df_usage.isnull().mean() * 100
```

Out[85]:

```
user_id                     0.000000
duolingo_start_date         0.000000
daily_goal                 56.301838
highest_course_progress     0.227679
took_placement_test         0.227679
purchased_subscription      0.000000
highest_crown_count         4.748740
n_active_days               0.000000
n_lessons_started           2.536998
n_lessons_completed         2.536998
longest_streak              0.000000
n_days_on_platform          0.000000
dtype: float64
```

Handling missing values

Note on missing values:

- **over 50% of daily_goal is missing, might consider drop the feature or transform into categorical feature (e.g., [nan, 1-3, ...])**
- **null values handling in survey dataset is a bit tricky here since I do not know the reason for missing. For feautre other_sources null could mean no other resource is used, for age and annual income null is usually because respondent refuse to disclose. Will just drop rows if <10% is missing and for all others consider null as a category.**

In [86]:

```
df_survey = df_survey.dropna(subset=['age','duolingo_platform',
                             'duolingo_subscriber',
'duolingo_usage',
'employment_status',
'gender',
'primary_language_commitment',
'primary_language_review',
'primary_language_motivation',
```

```
'primary_language_proficiency',
'student'])
df_survey.shape
```

Out[86]:

```
(5378, 19)
```

In [87]:

```
df_usage = df_usage.dropna(subset=['duolingo_start_date',
'highest_course_progress',
'took_placement_test',
'purchased_subscription',
'highest_crown_count',
'n_active_days',
'n_lessons_started',
'n_lessons_completed',
'longest_streak',
'n_days_on_platform'])
df_usage.shape
```

Out[87]:

```
(5743, 12)
```

In [112]:

```
#join the two datasets based on user id
df = pd.merge(df_survey, df_usage, on='user_id')
df.shape
```

Out[112]:

```
(5010, 30)
```

Feature engineering

In [113]:

```
#transform daily_goal to categorical
df.daily_goal.value_counts()
```

Out[113]:

```
20.0    873
30.0    511
50.0    471
10.0    403
1.0      49
16.0      2
32.0      1
Name: daily_goal, dtype: int64
```

from above distribution, will allocate 16 to 10 and 32 to 30. So we end up with 6 categories, namely [nan,1,10,20,30,50] Then we can do ordinal encoding on them.

In [114]:

```
#transform categorical features that have ordinal nature to ordinal encoding
ordinal = ['age','annual_income','duolingo_usage','primary_language_proficiency','primary
_language_commitment','daily_goal']
age_mapper = {'Under 18':1,'18-34':2,'35 - 54':3,'55 - 74':4,'75 or older':5}
income_mapper = {'$0 - $10,000':1,'$11,000 - $25,000':2,'$26,000 - $75,000':3,'$76,000 -
$150,000':4,'$151,000 or more':5}
usage_mapper = {"I don't use Duolingo":1,'Less than once a month':2,'Monthly':3,'Weekly'
:4,'Daily':5}
prof_mapper = {'Beginner':1,'Intermediate':2,'Advanced':3}
com_mapper = {"I'm not at all committed to learning this language.":1,"I'm slightly commi
tted to learning this language.":2,"I'm moderately committed to learning this language.":
3,"I'm very committed to learning this language.":4,"I'm extremely committed to learning
```

```
this language.":5}
goal_mapper = {1:1,10:1,16:2,20:2,30:3,32:3,50:4}
df['age_n'] = df['age'].replace(age_mapper)
df['annual_income_n'] = df['annual_income'].replace(income_mapper)
df['duolingo_usage_n'] = df['duolingo_usage'].replace(usage_mapper)
df['primary_language_proficiency_n'] = df['primary_language_proficiency'].replace(prof_m
apper)
df['primary_language_commitment_n'] = df['primary_language_commitment'].replace(com_mapp
er)
df['daily_goal_n']=df['daily_goal'].replace(goal_mapper)
```

In [115]:

```
df['annual_income_n'] = df['annual_income_n'].fillna(0)
df['daily_goal_n'] = df['daily_goal_n'].fillna(0)
```

primary_language_motivation_followup and other_resources are tricky since it's set data (respondents can choose multiple options). My method is to create dummy variables for each option and then run PCA to extract some principal components to feed into the clustering model. The reason is that binary variabls are not very suitable for clustering model and including all options would introduce too many binary variables.

In [92]:

```
df['test'] = df['primary_language_motivation_followup'].map(lambda x: x.split(','),na_ac
tion='ignore')
options = []
for x in df['test'].tolist():
    if type(x) is not float:
        for t in x:
            if t not in options:
                options.append(t)
df['test'] = df['test'].apply(lambda d: d if isinstance(d, list) else ['no_answer'])
options.append('no_answer')
```

In [93]:

```
options_matrix_t = df['test'].apply(lambda x: [option in x for option in options]).tolis
t()
options_matrix=[]
for x in options_matrix_t:
    options_matrix.append([int(elem) for elem in x])
```

In [94]:

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(options_matrix)
options_matrix_sc = scaler.transform(options_matrix)
X = np.array(options_matrix_sc)
pca = PCA(n_components=6)
pca.fit(X)
print(pca.explained_variance_ratio_)
```

```
[0.13539717 0.09582105 0.0631722  0.06089517 0.05815888 0.05703661]
```

ok...pca does not seem to capture good components for primary_language_motivation_followup. For now will just change this to one binary variable (if any followup).

In [95]:

```
#now for other_resources
df['test'] = df['other_resources'].map(lambda x: x.split(','),na_action='ignore')
options = []
for x in df['test'].tolist():
    if type(x) is not float:
        for t in x:
            if t not in options:
```

```
                options.append(t)
df['test'] = df['test'].apply(lambda d: d if isinstance(d, list) else ['no_answer'])
options.append('no_answer')
```

In [96]:

```
options_matrix_t = df['test'].apply(lambda x: [option in x for option in options]).tolis
t()
options_matrix=[]
for x in options_matrix_t:
    options_matrix.append([int(elem) for elem in x])
```

In [97]:

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(options_matrix)
options_matrix_sc = scaler.transform(options_matrix)
X = np.array(options_matrix_sc)
pca = PCA(n_components=6)
pca.fit(X)
print(pca.explained_variance_ratio_)
```

```
[0.32462938 0.13283847 0.08531014 0.08075619 0.07675934 0.07358671]
```

pca works much better for other_resources but still does not cover more than half of variance. Will change this feature to just one binary variable (if any other_resources)

In [119]:

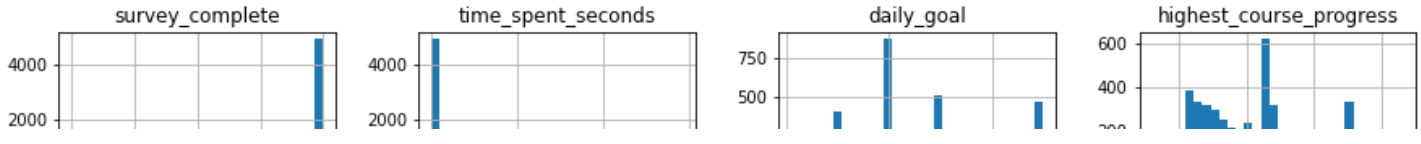
```
#binary feature engineering
binary = ['other_resources','primary_language_motivation_followup']
df['followup_bi'] = df['primary_language_motivation_followup'].fillna(0)
df['followup_bi'] = df['followup_bi'].apply(lambda d: d if d == 0 else 1)
df['resources_bi'] = df['other_resources'].fillna(0)
df['resources_bi'] = df['resources_bi'].apply(lambda d: d if d == 0 else 1)
```

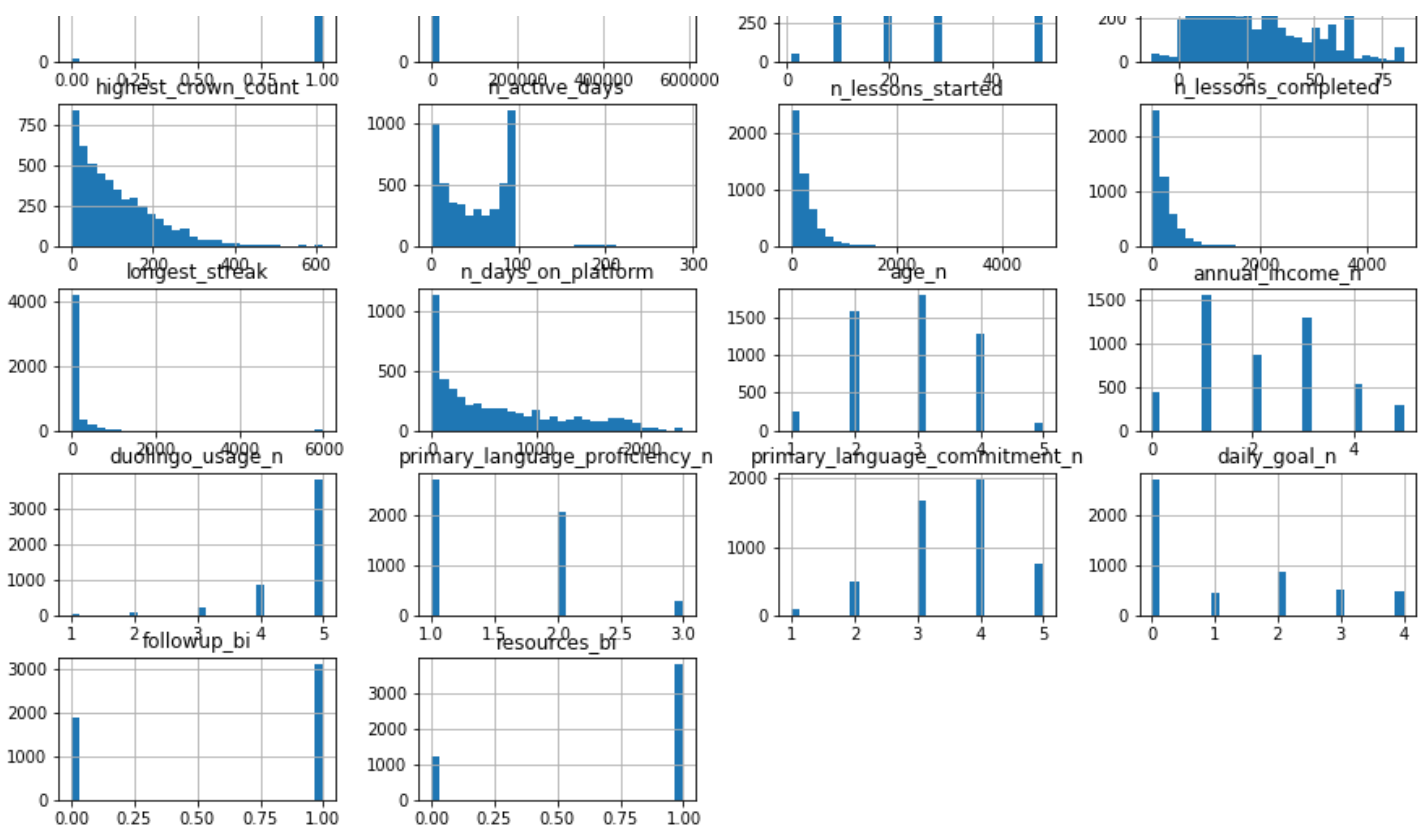
In [99]:

```
#some histograms just to see how the numerical values are distributed, so as to choose th
e appropriate scaler
df.hist(bins=30, figsize=(15, 10))
```

Out[99]:

```
array([[<AxesSubplot:title={'center':'survey_complete'}>,
        <AxesSubplot:title={'center':'time_spent_seconds'}>,
        <AxesSubplot:title={'center':'daily_goal'}>,
        <AxesSubplot:title={'center':'highest_course_progress'}>],
       [<AxesSubplot:title={'center':'highest_crown_count'}>,
        <AxesSubplot:title={'center':'n_active_days'}>,
        <AxesSubplot:title={'center':'n_lessons_started'}>,
        <AxesSubplot:title={'center':'n_lessons_completed'}>],
       [<AxesSubplot:title={'center':'longest_streak'}>,
        <AxesSubplot:title={'center':'n_days_on_platform'}>,
        <AxesSubplot:title={'center':'age_n'}>,
        <AxesSubplot:title={'center':'annual_income_n'}>],
       [<AxesSubplot:title={'center':'duolingo_usage_n'}>,
        <AxesSubplot:title={'center':'primary_language_proficiency_n'}>,
        <AxesSubplot:title={'center':'primary_language_commitment_n'}>,
        <AxesSubplot:title={'center':'daily_goal_n'}>],
       [<AxesSubplot:title={'center':'followup_bi'}>,
        <AxesSubplot:title={'center':'resources_bi'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```





Noticed some extreme outliers in longest_streak, n_active_days, n_lessons_started, n_lessons_completed. Better to use robust scaler.

It is also impossible to have 6000 longest_streak. Following this lead, I discovered some questionable observations dating back to the 2000s. Considering the fact that Duolingo first came out to the general public in June 2012, I suspect these are internal test accounts thus not suitable for the task.

In [120]:

```
df['duolingo_start_date'] = pd.to_datetime(df['duolingo_start_date'])
df = df[(df['duolingo_start_date'] > '2012-06-19') & (df['longest_streak']<3650)]
```

In [122]:

```
#numerical/ordinal features need to be scaled and relevant to the task
ordinal = ['age_n','annual_income_n','duolingo_usage_n','primary_language_proficiency_n'
,'primary_language_commitment_n']
numerical = ['highest_course_progress','highest_crown_count','n_active_days','n_lessons_s
tarted','n_lessons_completed','longest_streak','n_days_on_platform']
from sklearn.preprocessing import RobustScaler
rs = RobustScaler()
scaled_df = pd.DataFrame()
for i in ordinal+numerical:
    scaled = rs.fit_transform(df[i].values.reshape(-1,1))[:,0]
    scaled_d = pd.DataFrame(scaled,columns=[i])
    scaled_df = pd.concat([scaled_df,scaled_d],axis=1)
```

Train matrix preparation

In [127]:

```
#categorical features that are relevant to the task (including the binary ones)
cate = ['duolingo_platform','duolingo_subscriber','employment_status',
        'gender','primary_language_review','primary_language_motivation','student','resou
rces_bi','followup_bi']
cate_df = df[cate]
train_df = pd.concat([scaled_df,cate_df],axis=1)
```

In [129]:

```
scaled_df.reset_index(drop=True, inplace=True)
cate_df.reset_index(drop=True, inplace=True)
train_df = pd.concat([scaled_df,cate_df],axis=1)
```

In [105]:

```
train_df.head()
```

Out[105]:

	age_n	annual_income_n	duolingo_usage_n	primary_language_proficiency_n	primary_language_commitment_n	highest_cour
0	-0.5	0.5	0.0	2.0	0.0	
1	-0.5	0.5	-1.0	1.0	-2.0	
2	-0.5	1.0	0.0	0.0	-1.0	
3	-0.5	1.0	0.0	1.0	0.0	
4	0.0	1.0	0.0	1.0	0.0	

5 rows × 22 columns

In [139]:

```
cate_loc = [12,13,14,15,16,17,18,19,20]
```

In [140]:

```
dfMatrix = train_df.to_numpy()
dfMatrix
```

Out[140]:

```
array([[-0.5, 0.5, 0.0, ..., 'Not currently a student', 1, 1],
       [-0.5, 0.5, -1.0, ..., 'Not currently a student', 0, 1],
       [-0.5, 1.0, 0.0, ..., 'Not currently a student', 0, 0],
       ...,
       [0.5, 1.5, -4.0, ..., 'Full-time student', 1, 1],
       [0.5, 0.5, 0.0, ..., 'Not currently a student', 0, 0],
       [0.5, -1.0, 0.0, ..., 'Not currently a student', 1, 0]],
      dtype=object)
```

2. Clustering Analysis

Since our data is a mix of numerical and categorical value, I choose to use kprototype, which is based on kmeans but improved for categorical valus.

In [176]:

```
#choose number of clusters using elbow method
from kmodes.kprototypes import KPrototypes
cost = []
for cluster in range(1, 10):
```

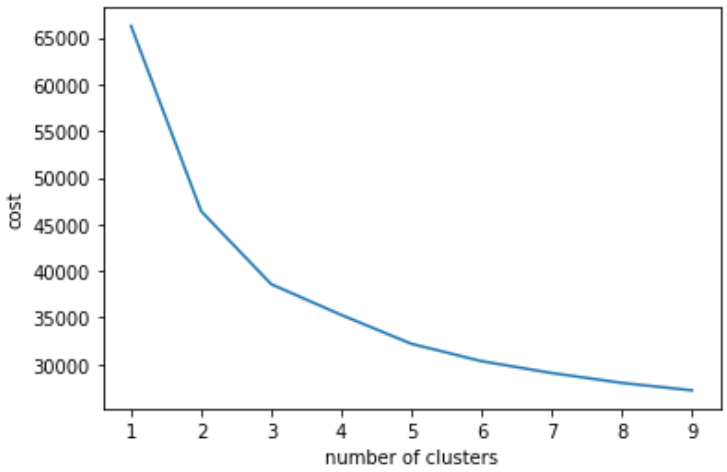
```
    kprototype = KPrototypes(n_jobs = -1, n_clusters = cluster, init = 'Cao', random_sta
te = 0)
    kprototype.fit_predict(dfMatrix, categorical = cate_loc)
    cost.append(kprototype.cost_)
```

In [177]:

```
plt.plot(range(1,10),cost)
plt.ylabel('cost')
plt.xlabel('number of clusters')
```

Out[177]:

```
Text(0.5, 0, 'number of clusters')
```



elbow method shows three

In [178]:

```
kprototype = KPrototypes(n_jobs = -1, n_clusters = 3, init = 'Cao', random_state = 0)
kprototype.fit_predict(dfMatrix, categorical = cate_loc)
```

Out[178]:

```
array([2, 0, 0, ..., 0, 0, 0], dtype=uint16)
```

3. Results Interpretation

In [179]:

```
kprototype.cluster_centroids_
```

Out[179]:

```
array([['-0.14893283113622097', '-0.0160075329566855',
        '-0.5131826741996234', '0.4322033898305085',
        '-0.6537978656622725', '-0.17870208022165865',
        '-0.17264232942199065', '-0.24295553610425927',
        '-0.18906569659526617', '-0.18792172981126967',
        '-0.019218697184799533', '0.13632491174864178', 'iPhone or iPad',
        'No, I have never paid for Duolingo Plus', 'Employed full-time',
        'Female',
        'I am using Duolingo to learn this language for the first time.',
        'I need to be able to speak the local language where I live',
        'Not currently a student', '1', '1'],
       ['0.09520500347463516', '0.24600416956219598',
        '-0.021542738012508687', '0.655316191799861',
        '-0.009034051424600417', '0.3977139296925544',
        '0.7729619928369043', '0.4806056631659301', '1.2459891711370306',
        '1.2604866603075122', '1.378009759524704', '0.27525728846090053',
        'Android phone or tablet',
        'Yes, I currently pay for Duolingo Plus', 'Employed full-time',
        'Male',
        "I am using Duolingo to review a language I've studied before.",
```

```
        'I want to connect with my heritage or identity',
        'Not currently a student', '1', '1'],
       ['0.09402985074626866', '0.29850746268656714',
        '-0.0029850746268656717', '0.8417910447761194',
        '-0.1791044776119403', '0.670200720535263', '1.1501722158438703',
        '0.5957536262350179', '0.7610759153871014', '0.7561886385562937',
        '7.970510086927963', '0.783301027330878', 'iPhone or iPad',
        'Yes, I currently pay for Duolingo Plus', 'Employed full-time',
        'Male',
        "I am using Duolingo to review a language I've studied before.",
        'I want to use my time more productively',
        'Not currently a student', '1', '1']], dtype='<U62')
```

In [180]:

```
# Add the cluster to the dataframe
df['cluster_label'] = kprototype.labels_
```

some visualization

Below are some visualization drafts. Quite messy. Not all plots are included in the report.

In [181]:

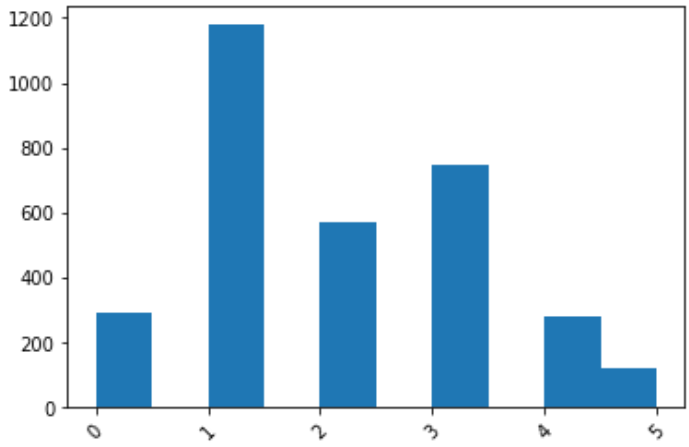
```
df_one = df[df['cluster_label']==0]
df_two = df[df['cluster_label']==1]
df_three = df[df['cluster_label']==2]
```

In [306]:

```
plt.hist(df_one["annual_income_n"])
plt.xticks(rotation=45)
```

Out[306]:

```
(array([-1.,  0.,  1.,  2.,  3.,  4.,  5.,  6.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



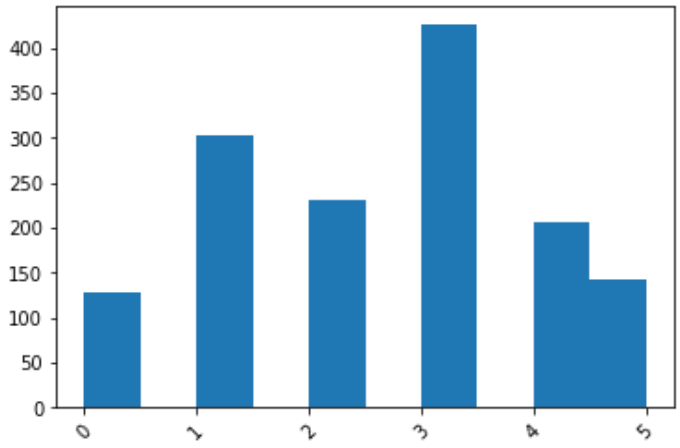
In [307]:

```
plt.hist(df_two["annual_income_n"])
plt.xticks(rotation=45)
```

Out[307]:

```
(array([-1.,  0.,  1.,  2.,  3.,  4.,  5.,  6.]),
 [Text(0, 0, ''),
```

```
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```

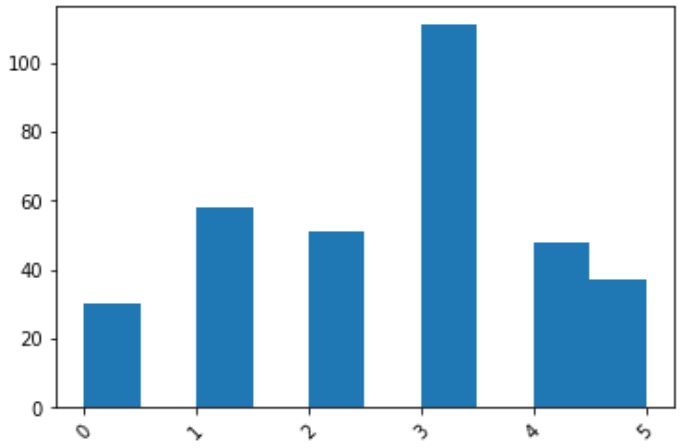


In [308]:

```
plt.hist(df_three["annual_income_n"])
plt.xticks(rotation=45)
```

Out[308]:

```
(array([-1.,  0.,  1.,  2.,  3.,  4.,  5.,  6.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```

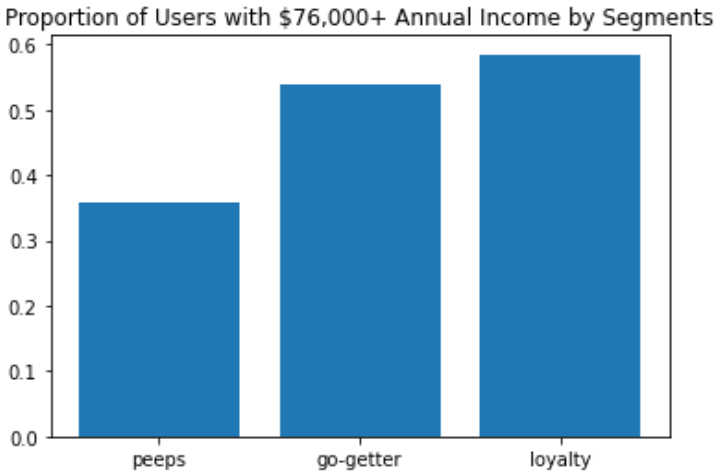


In [316]:

```
groups = [df_one,df_two,df_three]
labels = ['peeps','go-getter','loyalty']
income_id = []
for i in groups:
    num = i[(i['annual_income_n'] == 3) | (i['annual_income_n'] == 4) | (i['annual_incom
e_n'] == 5)].shape[0]
    de = i.shape[0]
    income_id.append(num/de)
plt.bar(labels,income_id)
plt.title('Proportion of Users with $76,000+ Annual Income by Segments')
```

Out[316]:

```
Text(0.5, 1.0, 'Proportion of Users with $76,000+ Annual Income by Segments')
```

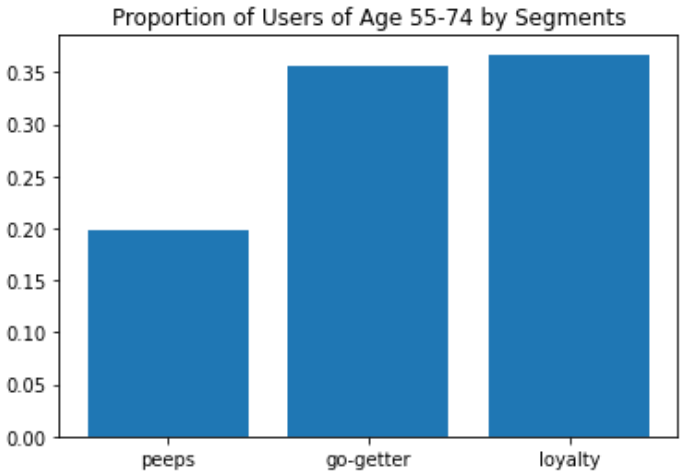


In [291]:

```
groups = [df_one,df_two,df_three]
labels = ['peeps','go-getter','loyalty']
age_id = []
for i in groups:
    num = i[i['age']=='55 - 74'].shape[0]
    de = i.shape[0]
    age_id.append(num/de)
plt.bar(labels,age_id)
plt.title('Proportion of Users of Age 55-74 by Segments')
```

Out[291]:

```
Text(0.5, 1.0, 'Proportion of Users of Age 55-74 by Segments')
```



In [325]:

```
df_one['primary_language_commitment'].value_counts()
```

Out[325]:

```
I'm moderately committed to learning this language.    1264
I'm very committed to learning this language.          1103
I'm slightly committed to learning this language.       434
I'm extremely committed to learning this language.      301
I'm not at all committed to learning this language.      84
Name: primary_language_commitment, dtype: int64
```

In [329]:

```
def get_pie_df(df,options):
    pers = []
    for i in options:
        de = df.shape[0]
        num = df[df['primary_language_commitment']==i].shape[0]
        pers.append(round((num/de)*100,0))
    return pers
```

In [332]:

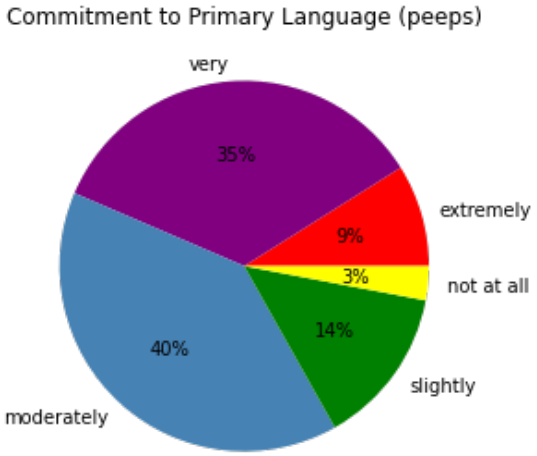
```
pers
```

Out[332]:

```
[9.0, 35.0, 0.0, 14.0, 3.0]
```

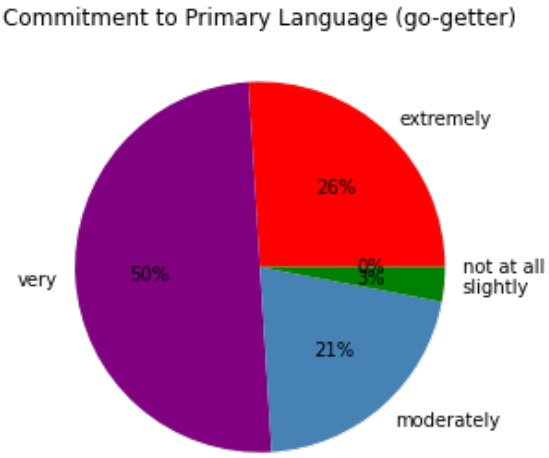
In [336]:

```
options = ["I'm extremely committed to learning this language.",
"I'm very committed to learning this language.",
"I'm moderately committed to learning this language.",
"I'm slightly committed to learning this language.",
"I'm not at all committed to learning this language."]
labels = ['extremely','very','moderately','slightly','not at all']
pers = get_pie_df(df_one,options)
fig, ax = plt.subplots()
ax.pie(pers, labels=labels, colors = ['red', 'purple', 'steelblue','green','yellow'],aut
opct='%.0f%%')
ax.set_title('Commitment to Primary Language (peeps)')
plt.tight_layout()
```



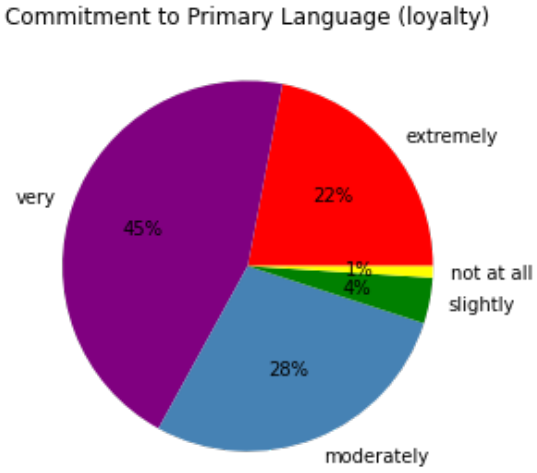
In [337]:

```
options = ["I'm extremely committed to learning this language.",
"I'm very committed to learning this language.",
"I'm moderately committed to learning this language.",
"I'm slightly committed to learning this language.",
"I'm not at all committed to learning this language."]
labels = ['extremely','very','moderately','slightly','not at all']
pers = get_pie_df(df_two,options)
fig, ax = plt.subplots()
ax.pie(pers, labels=labels, colors = ['red', 'purple', 'steelblue','green','yellow'],aut
opct='%.0f%%')
ax.set_title('Commitment to Primary Language (go-getter)')
plt.tight_layout()
```



In [338]:

```
options = ["I'm extremely committed to learning this language.",
"I'm very committed to learning this language.",
"I'm moderately committed to learning this language.",
"I'm slightly committed to learning this language.",
"I'm not at all committed to learning this language."]
labels = ['extremely','very','moderately','slightly','not at all']
pers = get_pie_df(df_three,options)
fig, ax = plt.subplots()
ax.pie(pers, labels=labels, colors = ['red', 'purple', 'steelblue','green','yellow'],aut
opct='%.0f%%')
ax.set_title('Commitment to Primary Language (loyalty)')
plt.tight_layout()
```



In [347]:

```
df_two['duolingo_subscriber'].value_counts()
```

Out[347]:

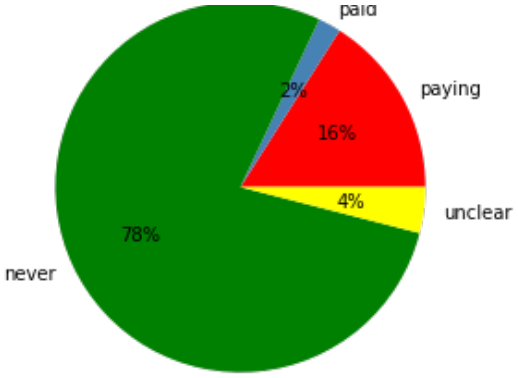
```
Yes, I currently pay for Duolingo Plus              756
No, I have never paid for Duolingo Plus             607
No, but I have previously paid for Duolingo Plus     54
I don't know if I pay for Duolingo Plus              22
Name: duolingo_subscriber, dtype: int64
```

In [341]:

```
options = ['Yes, I currently pay for Duolingo Plus',
           'No, but I have previously paid for Duolingo Plus',
           'No, I have never paid for Duolingo Plus',
           "I don't know if I pay for Duolingo Plus"]
def get_pie_df_plus(df,options):
    pers = []
    for i in options:
        de = df.shape[0]
        num = df[df['duolingo_subscriber']==i].shape[0]
        pers.append(round((num/de)*100,0))
    return pers
```

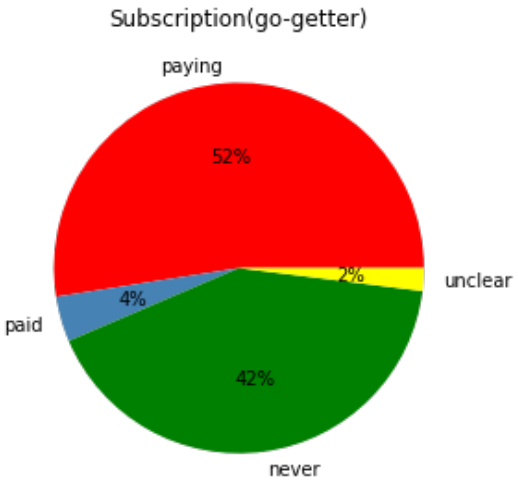
In [343]:

```
labels = ['paying','paid','never','unclear']
pers = get_pie_df_plus(df_one,options)
fig, ax = plt.subplots()
ax.pie(pers, labels=labels, colors = ['red', 'steelblue','green','yellow'],autopct='%.0f
%%')
ax.set_title('Subscription(peeps)')
plt.tight_layout()
```



In [346]:

```
labels = ['paying','paid','never','unclear']
pers = get_pie_df_plus(df_two,options)
fig, ax = plt.subplots()
ax.pie(pers, labels=labels, colors = ['red', 'steelblue','green','yellow'],autopct='%.0f
%%')
ax.set_title('Subscription(go-getter)')
plt.tight_layout()
```



In [349]:

```
df_two['purchased_subscription'].value_counts()
```

Out[349]:

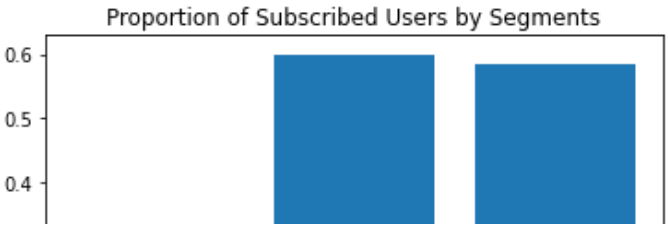
```
True     863
False    576
Name: purchased_subscription, dtype: int64
```

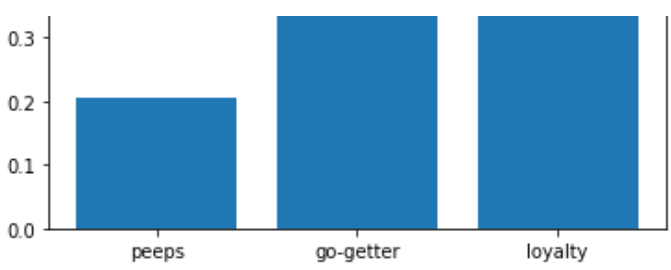
In [353]:

```
perclist = [df_one['purchased_subscription'].sum()/df_one.shape[0],
            df_two['purchased_subscription'].sum()/df_two.shape[0],
            df_three['purchased_subscription'].sum()/df_three.shape[0]]
labels = ['peeps','go-getter','loyalty']
plt.bar(labels,perclist)
plt.title('Proportion of Subscribed Users by Segments')
```

Out[353]:

```
Text(0.5, 1.0, 'Proportion of Subscribed Users by Segments')
```





In [350]:

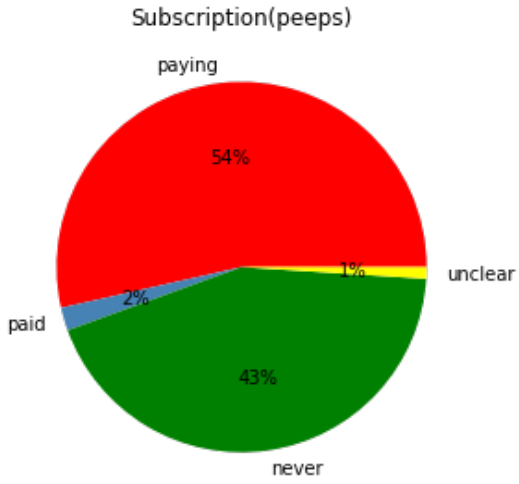
```
df_three['purchased_subscription'].value_counts()
```

Out[350]:

```
True     196
False    139
Name: purchased_subscription, dtype: int64
```

In [345]:

```
labels = ['paying','paid','never','unclear']
pers = get_pie_df_plus(df_three,options)
fig, ax = plt.subplots()
ax.pie(pers, labels=labels, colors = ['red', 'steelblue','green','yellow'],autopct='%.0f
%%')
ax.set_title('Subscription(peeps)')
plt.tight_layout()
```

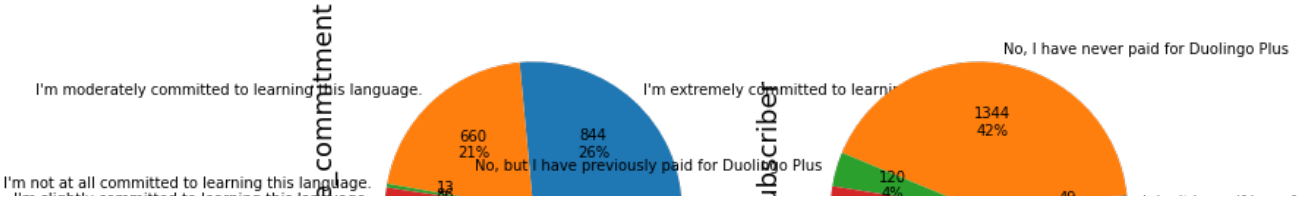


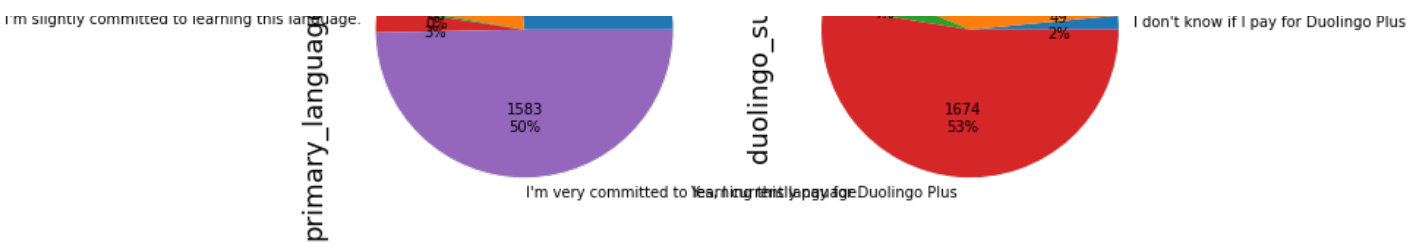
In [287]:

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 5))

df_two.groupby('primary_language_commitment').size().plot(kind='pie', autopct=label_func
tion, textprops={'fontsize': 10},
                                          ax=ax1)
df_two.groupby('duolingo_subscriber').size().plot(kind='pie', autopct=label_function, te
xtprops={'fontsize': 10},
                                        ax=ax2)
ax1.set_ylabel('primary_language_commitment', size=18)
ax2.set_ylabel('duolingo_subscriber', size=18)
plt.tight_layout()
plt.show()
```

```
<ipython-input-287-1a162e6ffdf7>:9: UserWarning: Tight layout not applied. tight_layout c
annot make axes width small enough to accommodate all axes decorations
  plt.tight_layout()
```



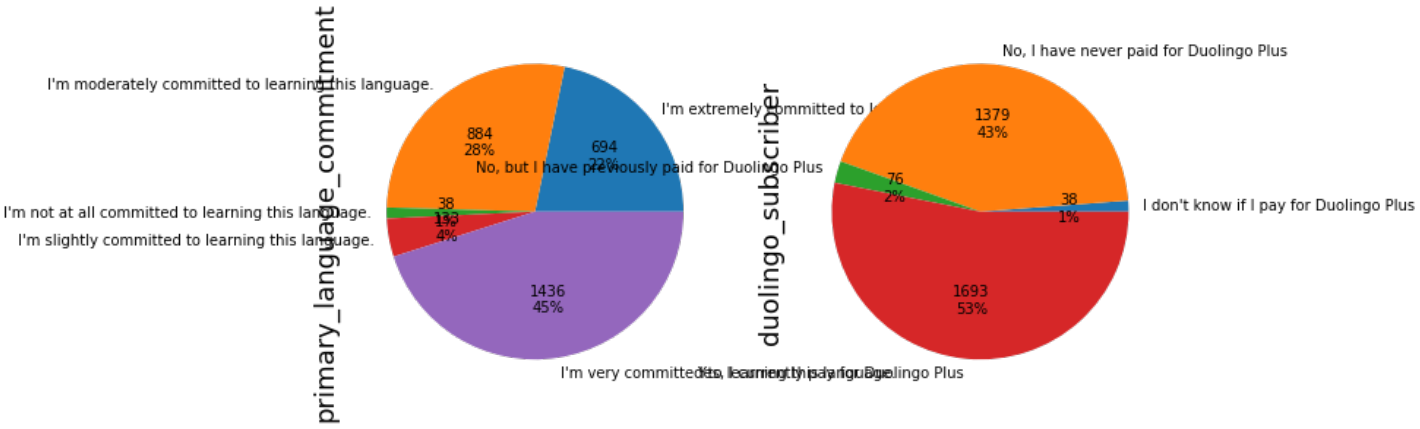


In [288]:

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 5))

df_three.groupby('primary_language_commitment').size().plot(kind='pie', autopct=label_fun
ction, textprops={'fontsize': 10},
                                                            ax=ax1)
df_three.groupby('duolingo_subscriber').size().plot(kind='pie', autopct=label_function,
textprops={'fontsize': 10},
                                                    ax=ax2)
ax1.set_ylabel('primary_language_commitment', size=18)
ax2.set_ylabel('duolingo_subscriber', size=18)
plt.tight_layout()
plt.show()
```

```
<ipython-input-288-ac6524027a86>:9: UserWarning: Tight layout not applied. tight_layout c
annot make axes width small enough to accommodate all axes decorations
  plt.tight_layout()
```

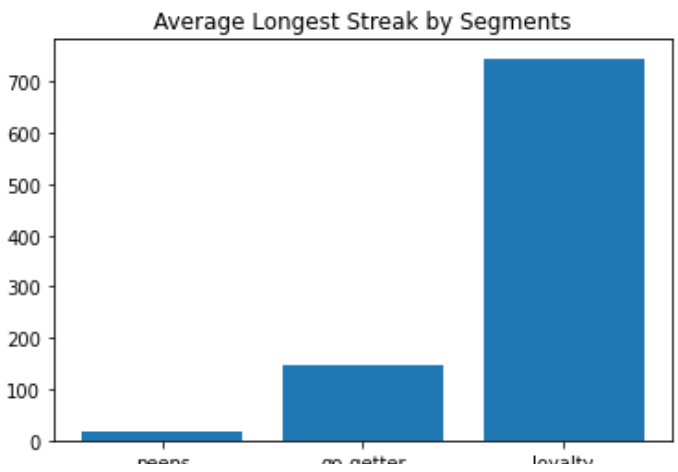


In [354]:

```
groups = [df_one,df_two,df_three]
labels = ['peeps','go-getter','loyalty']
perf_id = []
for i in groups:
    avg = i['longest_streak'].mean()
    perf_id.append(avg)
plt.bar(labels,perf_id)
plt.title('Average Longest Streak by Segments')
```

Out[354]:

```
Text(0.5, 1.0, 'Average Longest Streak by Segments')
```

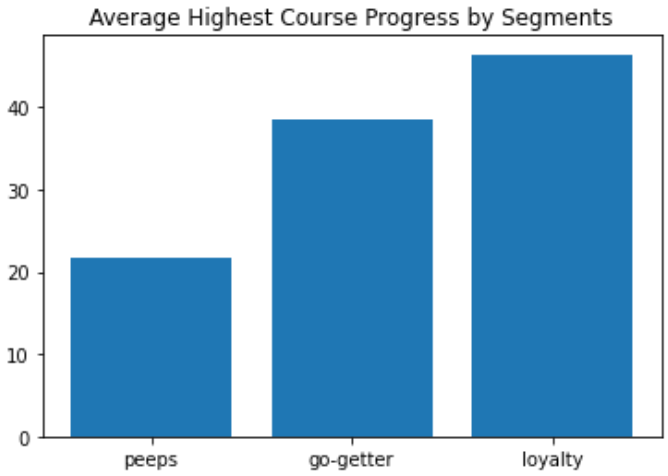


In [355]:

```
groups = [df_one,df_two,df_three]
labels = ['peeps','go-getter','loyalty']
perf_id = []
for i in groups:
    avg = i['highest_course_progress'].mean()
    perf_id.append(avg)
plt.bar(labels,perf_id)
plt.title('Average Highest Course Progress by Segments')
```

Out[355]:

```
Text(0.5, 1.0, 'Average Highest Course Progress by Segments')
```

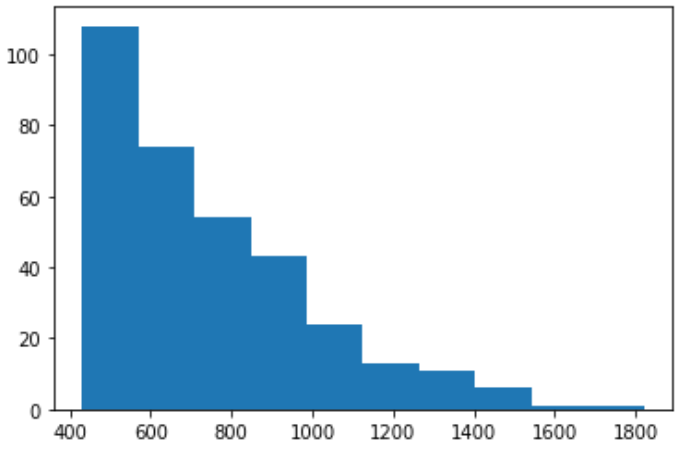


In [356]:

```
plt.hist(df_three['longest_streak'])
```

Out[356]:

```
(array([108.,  74.,  54.,  43.,  24.,  13.,  11.,   6.,   1.,   1.]),
 array([ 431. ,  569.9,  708.8,  847.7,  986.6, 1125.5, 1264.4, 1403.3,
        1542.2, 1681.1, 1820. ]),
 <BarContainer object of 10 artists>)
```

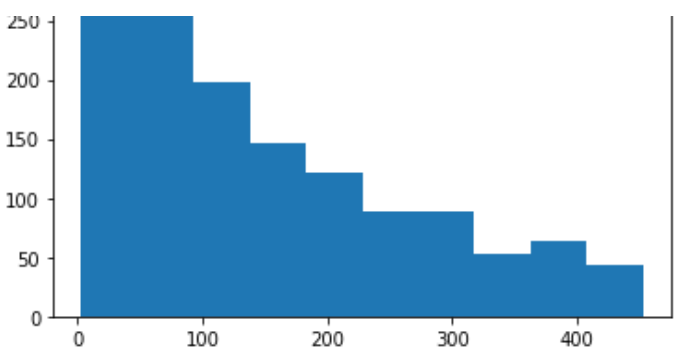


In [357]:

```
plt.hist(df_two['longest_streak'])
```

Out[357]:

```
(array([323., 314., 197., 147., 121.,  89.,  88.,  53.,  64.,  43.]),
 array([  3.,  48.,  93., 138., 183., 228., 273., 318., 363., 408., 453.]),
 <BarContainer object of 10 artists>)
```


250
200
150
100
50
0
0
100
200
300
400