JWillis_D208ExpDA_PA2

January 8, 2023

1 Import Libraries

[55]:
```
# pip install bioinfokit
```

[56]:
```
import pandas as pd
import numpy as np
import scipy as sc
import matplotlib.pyplot as plt
import seaborn as sb
import statsmodels as stats
from pandas import DataFrame
from scipy.stats import kurtosis, skew
from matplotlib.ticker import StrMethodFormatter
from bioinfokit.analys import stat
```

1.0.1 Load Data From medical_clean.csv

[57]:
```
# load data file
df = pd.read_csv('medical_clean.csv')
# quick test the data is present and see the shape
df.head(-1)
```

[57]:

	CaseOrder	Customer_id	Interaction	
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	
…	…	…	…	
9994	9995	M583491	15c2b4bb-2c36-41b2-b1e2-206144fae1dc	
9995	9996	B863060	a25b594d-0328-486f-a9b9-0567eb0f9723	
9996	9997	P712040	70711574-f7b1-4a17-b15f-48c54564b70f	
9997	9998	R778890	1d79569d-8e0f-4180-a207-d67ee4527d26	
9998	9999	E344109	f5a68e69-2a60-409b-a92f-ac0847b27db0	

	UID	City	State	County	\
0	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	Morgan	

```
1     176354c5eef714957d486009feabf195      Marianna    FL       Jackson
2     e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD     Minnehaha
3     cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca
4     d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William
…                                  …           …   …           …
9994  b9dd180aa8894ecea6af33a46b22e015     Fellsmere    FL  Indian River
9995  39184dc28cc038871912ccc4500049e5       Norlina    NC        Warren
9996  3cd124ccd43147404292e883bf9ec55c        Milmay    NJ      Atlantic
9997  41b770aeee97a5b9e7f69c906a8119d7     Southside    TN    Montgomery
9998  2bb491ef5b1beb1fed758cc6885c167a         Quinn    SD    Pennington

        Zip       Lat        Lng  …  TotalCharge Additional_charges Item1  \
0     35621  34.34960  -86.72508  …  3726.702860       17939.403420     3
1     32446  30.84513  -85.22907  …  4193.190458       17612.998120     3
2     57110  43.54321  -96.63772  …  2434.234222       17505.192460     2
3     56072  43.89744  -93.51479  …  2127.830423       12993.437350     3
4     23181  37.59894  -76.88958  …  2113.073274        3716.525786     2
…     …       …         …  …           …                  …   …
9994  32948  27.88942  -80.73347  …  5607.716000       12045.860000     2
9995  27563  36.42886  -78.23716  …  6850.942000        8927.642000     3
9996   8340  39.43609  -74.87302  …  7741.690000       28507.150000     3
9997  37171  36.36655  -87.29988  …  8276.481000       15281.210000     3
9998  57775  44.10354 -102.01590  …  7644.483000        7781.678000     5

     Item2  Item3  Item4  Item5 Item6 Item7 Item8
0        3      2      2      4     3     3     4
1        4      3      4      4     4     3     3
2        4      4      4      3     4     3     3
3        5      5      3      4     5     5     5
4        1      3      3      5     3     4     3
…    …     …     …    …   …   …   …
9994     3      2      1      5     3     4     2
9995     2      2      3      4     3     4     2
9996     3      4      2      5     3     4     4
9997     3      3      4      4     2     3     2
9998     5      3      4      4     3     4     3

[9999 rows x 50 columns]
```

1.0.2 Any Rows With All Nulls?

[58]:
```
df.isnull().all(axis=1).any()
```

[58]:
```
False
```

1.0.3 Any Missing Values?

[59]:
```
df.loc[:, df.isnull().any()]
```

[59]:
```
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99,
…]

[10000 rows x 0 columns]
```

1.0.4 [A] Question: "Did a patient's initial inpatient stay (Initial_days) show an effect on their potential readmissions within our hospital chain?"

[60]:
```
#https://stackoverflow.com/questions/24524104/
 ↪pandas-describe-is-not-returning-summary-of-all-columns
# Describe Numeric Fields
df.describe(include = [np.number])
```

[60]:

	CaseOrder	Zip	Lat	Lng	Population
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	50159.323900	38.751099	-91.243080	9965.253800
std	2886.89568	27469.588208	5.403085	15.205998	14824.758614
min	1.00000	610.000000	17.967190	-174.209700	0.000000
25%	2500.75000	27592.000000	35.255120	-97.352982	694.750000
50%	5000.50000	50207.000000	39.419355	-88.397230	2769.000000
75%	7500.25000	72411.750000	42.044175	-80.438050	13945.000000
max	10000.00000	99929.000000	70.560990	-65.290170	122814.000000

	Children	Age	Income	VitD_levels	Doc_visits
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	2.097200	53.511700	40490.495160	17.964262	5.012200
std	2.163659	20.638538	28521.153293	2.017231	1.045734
min	0.000000	18.000000	154.080000	9.806483	1.000000
25%	0.000000	36.000000	19598.775000	16.626439	4.000000
50%	1.000000	53.000000	33768.420000	17.951122	5.000000
75%	3.000000	71.000000	54296.402500	19.347963	6.000000
max	10.000000	89.000000	207249.100000	26.394449	9.000000

	…	TotalCharge	Additional_charges	Item1	Item2
count	…	10000.000000	10000.000000	10000.000000	10000.000000

```
mean   …   5312.172769        12934.528587      3.518800      3.506700
std    …   2180.393838         6542.601544      1.031966      1.034825
min    …   1938.312067         3125.703000      1.000000      1.000000
25%    …   3179.374015         7986.487755      3.000000      3.000000
50%    …   5213.952000        11573.977735      4.000000      3.000000
75%    …   7459.699750        15626.490000      4.000000      4.000000
max    …   9180.728000        30566.070000      8.000000      7.000000

              Item3         Item4         Item5         Item6         Item7  \
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.000000
mean       3.511100      3.515100      3.496900      3.522500      3.494000
std        1.032755      1.036282      1.030192      1.032376      1.021405
min        1.000000      1.000000      1.000000      1.000000      1.000000
25%        3.000000      3.000000      3.000000      3.000000      3.000000
50%        4.000000      4.000000      3.000000      4.000000      3.000000
75%        4.000000      4.000000      4.000000      4.000000      4.000000
max        8.000000      7.000000      7.000000      7.000000      7.000000

              Item8
count  10000.000000
mean       3.509700
std        1.042312
min        1.000000
25%        3.000000
50%        3.000000
75%        4.000000
max        7.000000

[8 rows x 23 columns]
```

[61]:
```
# Describe Categorical Fields
df.describe(include = ['O'])
```

[61]:
```
       Customer_id                            Interaction  \
count        10000                                  10000
unique       10000                                  10000
top         C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
freq            1                                      1

                                    UID     City  State     County   Area  \
count                             10000    10000  10000      10000  10000
unique                            10000     6072     52       1607      3
top     3a83ddb66e2ae73798bdf1d705dc0932  Houston     TX  Jefferson  Rural
freq                                  1       36    553        118   3369

                TimeZone                                  Job  Marital  …  \
count              10000                               10000    10000  …
```

```
unique                26                                    639        5  …
top     America/New_York  Outdoor activities/education manager  Widowed  …
freq                3889                                     29     2045  …

       Overweight Arthritis Diabetes Hyperlipidemia BackPain Anxiety  \
count       10000     10000    10000          10000    10000   10000
unique          2         2        2              2        2       2
top           Yes        No       No             No       No      No
freq         7094      6426     7262           6628     5886    6785

       Allergic_rhinitis Reflux_esophagitis Asthma    Services
count              10000              10000  10000       10000
unique                 2                  2      2           4
top                   No                 No     No  Blood Work
freq                6059               5865   7107        5265

[4 rows x 27 columns]
```

[B cont.] Create Subset Data Group to Focus On and Describe

[62]:
```
focus_df = df[['ReAdmis', 'Overweight', 'TotalCharge', 'Initial_days']]
focus_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   ReAdmis       10000 non-null  object
 1   Overweight    10000 non-null  object
 2   TotalCharge   10000 non-null  float64
 3   Initial_days  10000 non-null  float64
dtypes: float64(2), object(2)
memory usage: 312.6+ KB
```

[B cont.] Describe the data analysis by doing the following: *Using one of the following techniques, write code (in either Python or R) to run the analysis of the data set:* ***ANOVA***

1.1 One-Way ANOVA

1.1.1 Does a Patients Initial Stay Affect Readmissions?

[63]:
```
# https://www.marsja.se/four-ways-to-conduct-one-way-anovas-using-python/
# get ANOVA table
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols
```

[64]:
```
# Set up ANOVA Model
mod = ols('Initial_days ~ ReAdmis',
                data=focus_df).fit()

# Carry out the ANOVA
aov_table = sm.stats.anova_lm(mod, typ=2)
print(aov_table)
```

```
                sum_sq      df             F  PR(>F)
ReAdmis    5.010653e+06     1.0  26222.105595     0.0
Residual   1.910469e+06  9998.0           NaN     NaN
```

[65]:
```
model = smf.ols('Initial_days ~ ReAdmis', data=focus_df).fit()

aov_table = sm.stats.anova_lm(model)
aov_table
```

[65]:

	df	sum_sq	mean_sq	F	PR(>F)
ReAdmis	1.0	5.010653e+06	5.010653e+06	26222.105595	0.0
Residual	9998.0	1.910469e+06	1.910851e+02	NaN	NaN

[66]:
```
# ANOVA table using bioinfokit v1.0.3 or later (it uses wrapper script for␣
 ↪anova_lm)
from bioinfokit.analys import stat
res_t = stat()
res_t.anova_stat(df=df, res_var='value', anova_model='Initial_days ~␣
 ↪C(ReAdmis)')
res_t.anova_summary
```

[66]:

	df	sum_sq	mean_sq	F	PR(>F)
C(ReAdmis)	1.0	5.010653e+06	5.010653e+06	26222.105595	0.0
Residual	9998.0	1.910469e+06	1.910851e+02	NaN	NaN

[C] *Identify the distribution of two continuous variables and two categorical variables* ***using univariate*** - Represent your findings in Part C, visually as part of your submission.

Note on Univariate Statistics:

- General Information: (data type, count of total values, number of unique values)
- Range and Middle: (min, max, mean, median, mode, quartiles)
- Normality and Spread: (std dev, skweness, kurtosis)

[67]:
```
#Describe Data:
print('Continuous: \n\n' + str(focus_df.describe(include = [np.number])))
print('\n' + '------'*5 +'\n')
print('Categorical: \n\n' + str(focus_df.describe(include = ['O'])))
```

```
Continuous:

        TotalCharge  Initial_days
count  10000.000000  10000.000000
mean    5312.172769     34.455299
std     2180.393838     26.309341
min     1938.312067      1.001981
25%     3179.374015      7.896215
50%     5213.952000     35.836244
75%     7459.699750     61.161020
max     9180.728000     71.981490

------------------------------

Categorical:

       ReAdmis Overweight
count    10000      10000
unique       2          2
top         No        Yes
freq      6331       7094
```

[68]:
```
print('Describe Data (cont.)')
print('Median: \n\n' + str(focus_df.median()))
print('\n' + '------'*5 +'\n')
#print('Mode: \n\n' + str(focus_df.mode(numeric_only=True)))
print('Mode: \n\n' + str(focus_df['TotalCharge'].value_counts(ascending=True).
 ↪loc[lambda x : x>1].to_frame()) +
      '\n\n' + str(focus_df['Initial_days'].value_counts(ascending=True).
 ↪loc[lambda x : x>1].to_frame()))
```

```
Describe Data (cont.)
Median:

TotalCharge     5213.952000
Initial_days      35.836244
dtype: float64

------------------------------

Mode:

          TotalCharge
7555.452            2
7964.681            2
8081.346            2

          Initial_days
```

```
63.54432             2
70.32542             2
67.42139             2
```

```
/var/folders/45/_087y05165x0c7wb_dw4k6nh0000gn/T/ipykernel_60101/2124304241.py:2
: FutureWarning: The default value of numeric_only in DataFrame.median is
deprecated. In a future version, it will default to False. In addition,
specifying 'numeric_only=None' is deprecated. Select only valid columns or
specify the value of numeric_only to silence this warning.
  print('Median: \n\n' + str(focus_df.median()))
```

[69]:
```
# Look for Unique Values
print('Look for Unique Values:')
print('-----'*5)
focus_df.nunique()
```

```
Look for Unique Values:
-------------------------
```

[69]:
```
ReAdmis            2
Overweight         2
TotalCharge     9997
Initial_days    9997
dtype: int64
```

[70]:
```
# Look for Nulls
print('Look for Nulls:')
print('-----'*5)
focus_df.isnull().sum()
```

```
Look for Nulls:
-------------------------
```

[70]:
```
ReAdmis         0
Overweight      0
TotalCharge     0
Initial_days    0
dtype: int64
```

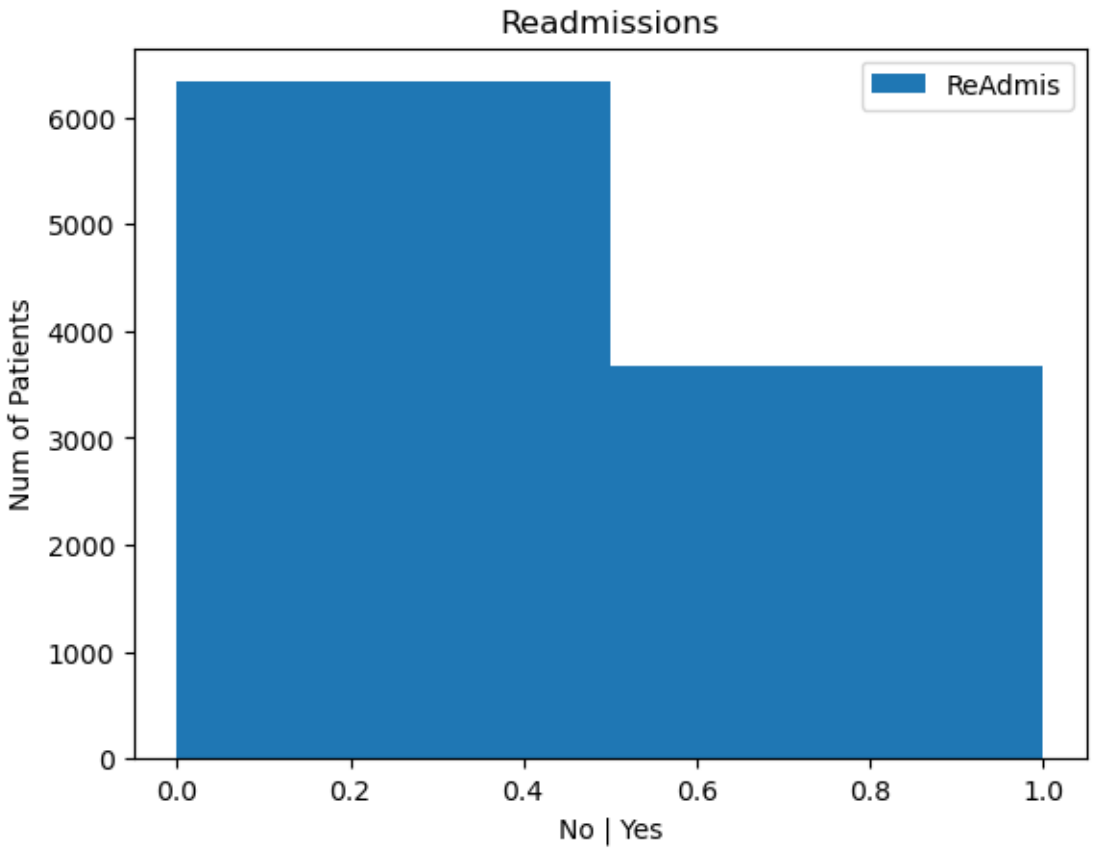
[71]:
```
print('Look at Skewness (rule = +/- 1):')
print('-----'*5)
print('Total Charge: \t' + str(focus_df.TotalCharge.skew()))
print('Initial Days: \t' + str(focus_df.Initial_days.skew()))
print('')
print('Look at Kurtosis (rule = +/- 1):')
print('-----'*5)
print('Total Charge: \t' + str(focus_df.TotalCharge.kurt()))
print('Initial Days: \t' + str(focus_df.Initial_days.kurt()))
```

```
Look at Skewness (rule = +/- 1):
-------------------------
Total Charge:   0.06966094634824574
Initial Days:   0.07028608266045332

Look at Kurtosis (rule = +/- 1):
-------------------------
Total Charge:   -1.6682665427232286
Initial Days:   -1.7545246170896873
```

Univariate Visualizations (cont.)

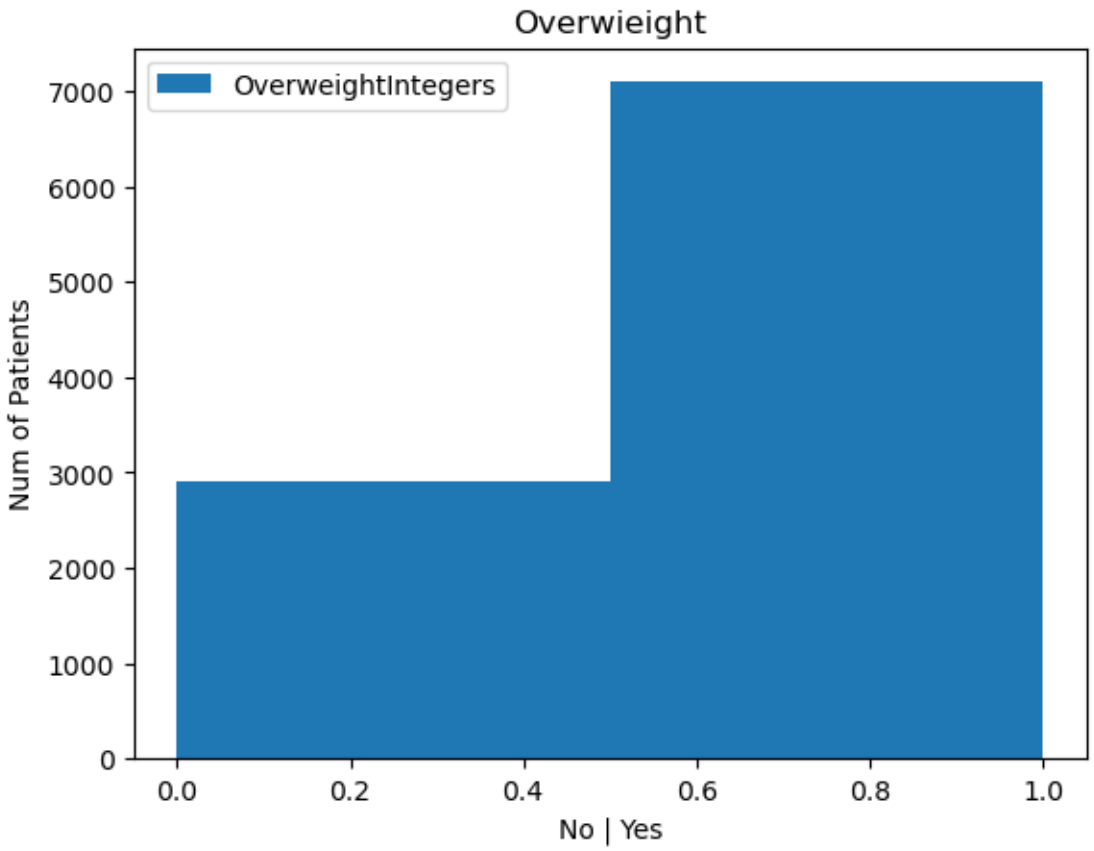
[72]:
```
# Univariate: Readmissions
readmis = []
for value in df['ReAdmis']:

    if value == 'Yes':
        readmis.append(1)
    elif value == 'No':
        readmis.append(0)

df['ReAdmis'] = readmis
df[['ReAdmis']].plot.hist(bins=2);

plt.xlabel('No | Yes')
plt.ylabel('Num of Patients')
plt.title('Readmissions')
```

[72]:
```
Text(0.5, 1.0, 'Readmissions')
```



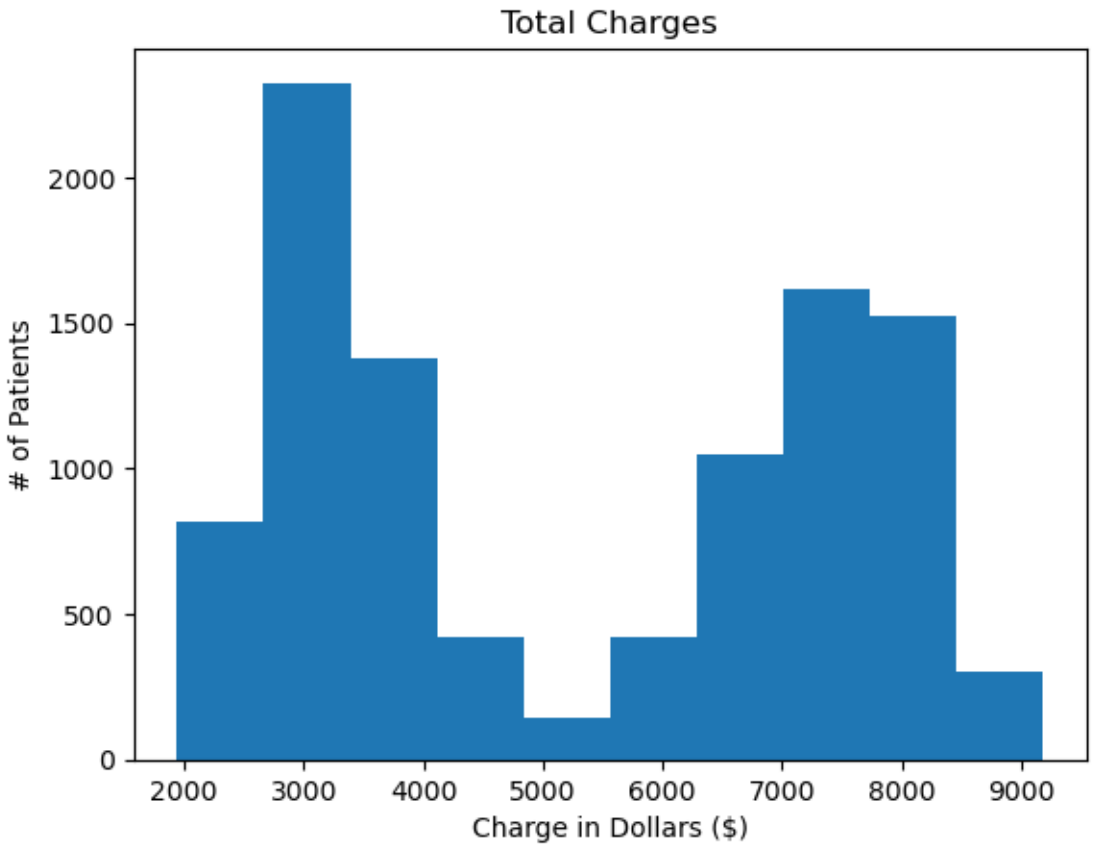
[73]:
```
# Univariate: Overweight
overweight = []
for value in df['Overweight']:

    if value == 'Yes':
        overweight.append(1)
    elif value == 'No':
        overweight.append(0)

df['OverweightIntegers'] = overweight
df[['OverweightIntegers']].plot.hist(bins=2);

plt.xlabel('No | Yes')
plt.ylabel('Num of Patients')
plt.title('Overwieight')
```

[73]:
```
Text(0.5, 1.0, 'Overwieight')
```



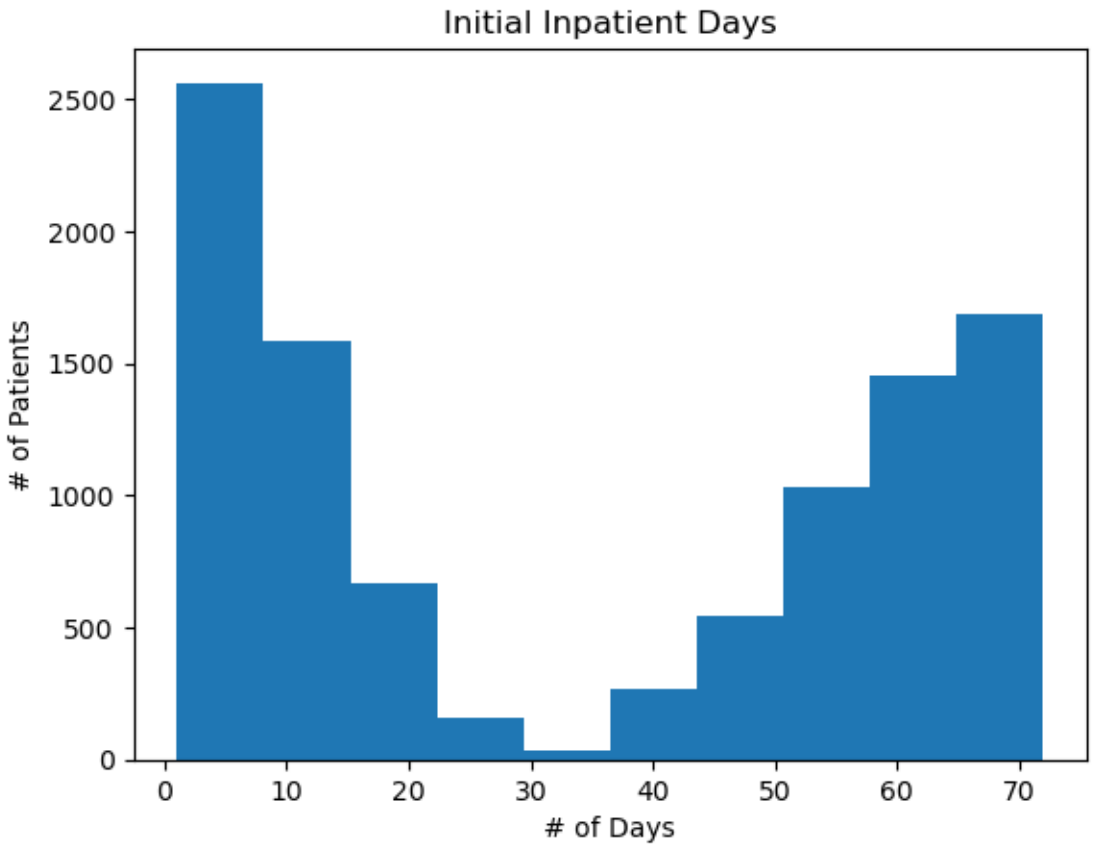
[74]:
```
import matplotlib.pyplot as plt

#Univariate: Total Charges
focus_df['TotalCharge'].plot.hist();

plt.xlabel('Charge in Dollars ($)')
plt.ylabel('# of Patients')
plt.title('Total Charges')
```

[74]: Text(0.5, 1.0, 'Total Charges')



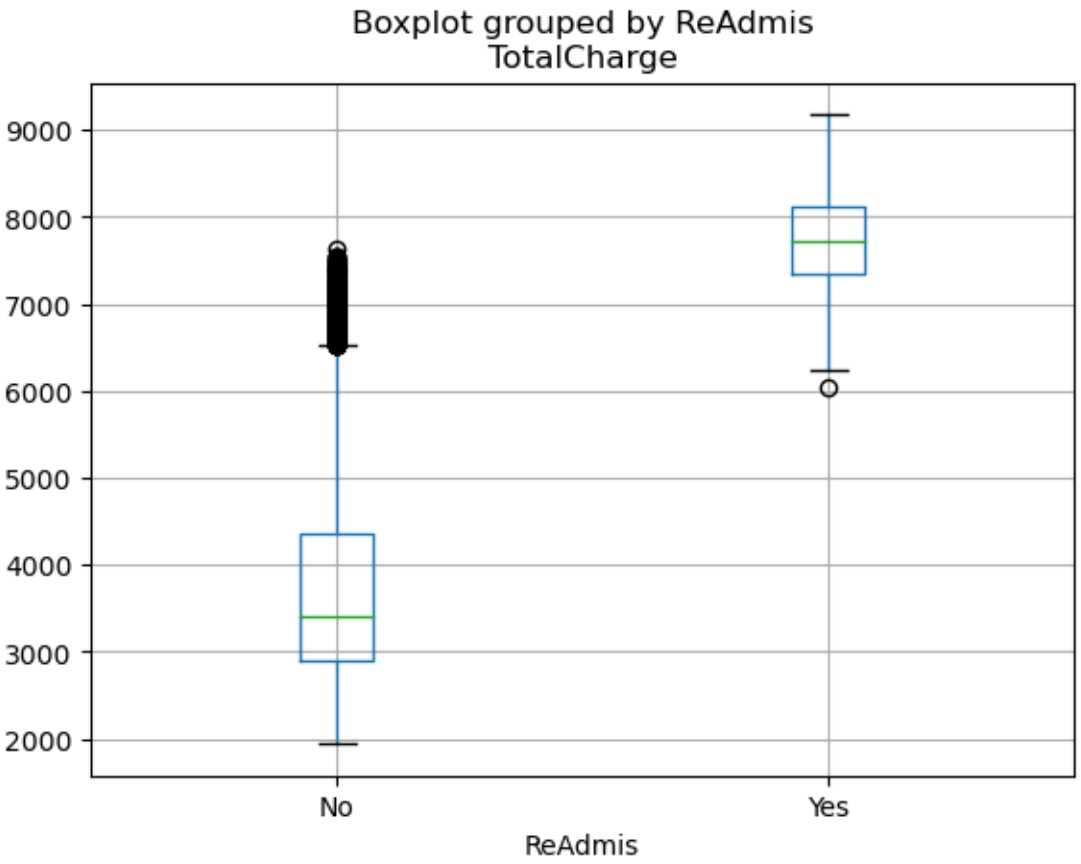
[75]:
```
#Univariate: Initial Days
focus_df['Initial_days'].plot.hist();

plt.xlabel('# of Days')
plt.ylabel('# of Patients')
plt.title('Initial Inpatient Days')
```

[75]: Text(0.5, 1.0, 'Initial Inpatient Days')

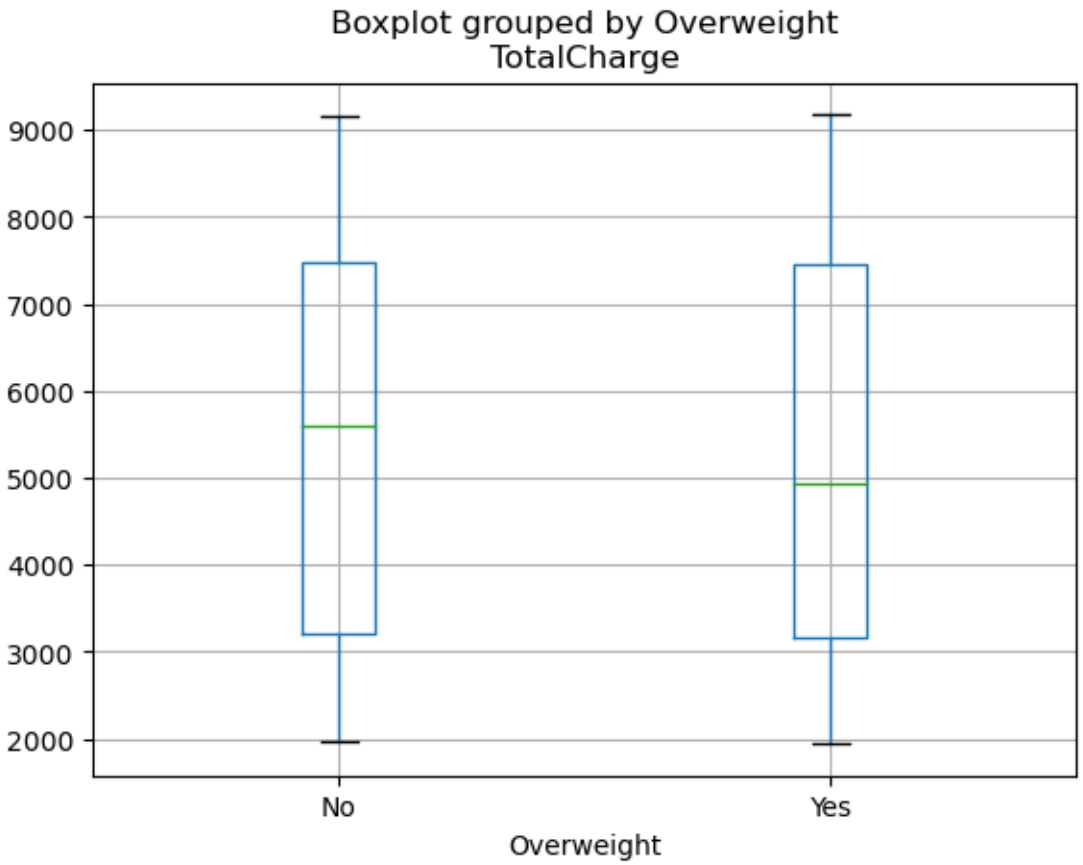


[D] *Identify the distribution of two continuous variables and two categorical variables* ***using bivariate s***
- Represent your findings in Part D, visually as part of your submission.

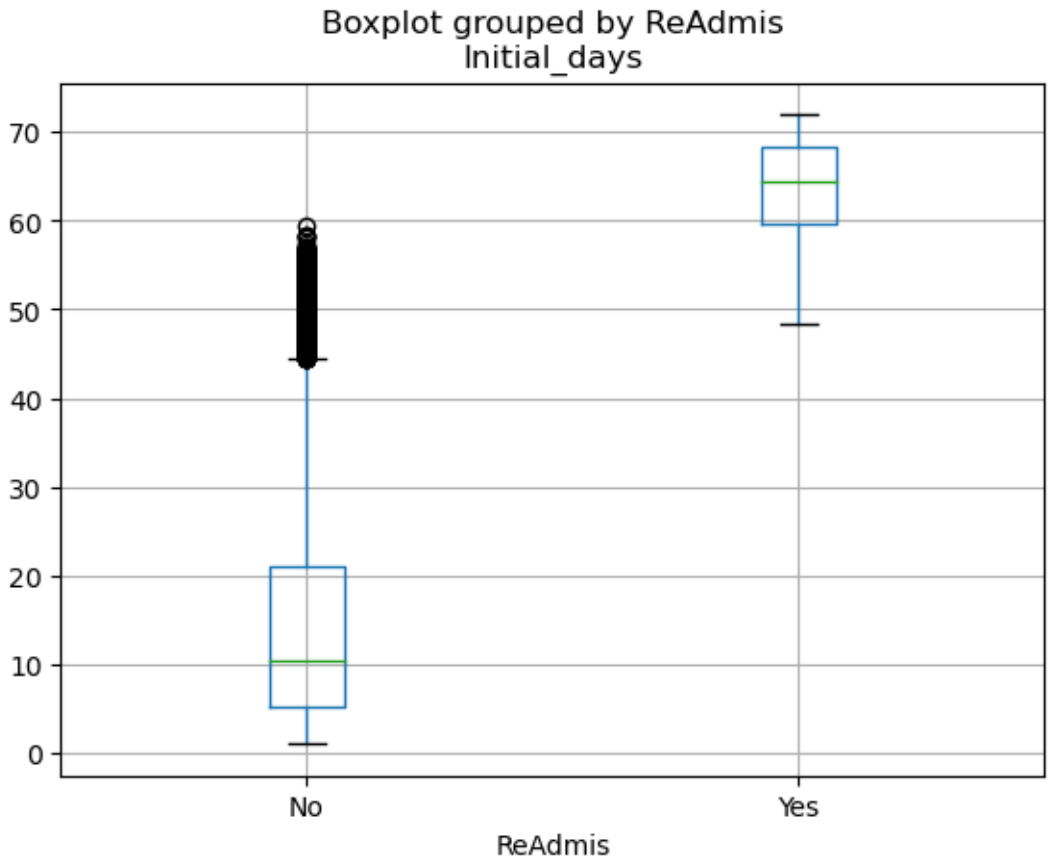
[76]:
```
# BoxPlot of Total Charge and Readmissions
focus_df.boxplot(column='TotalCharge', by='ReAdmis');
```



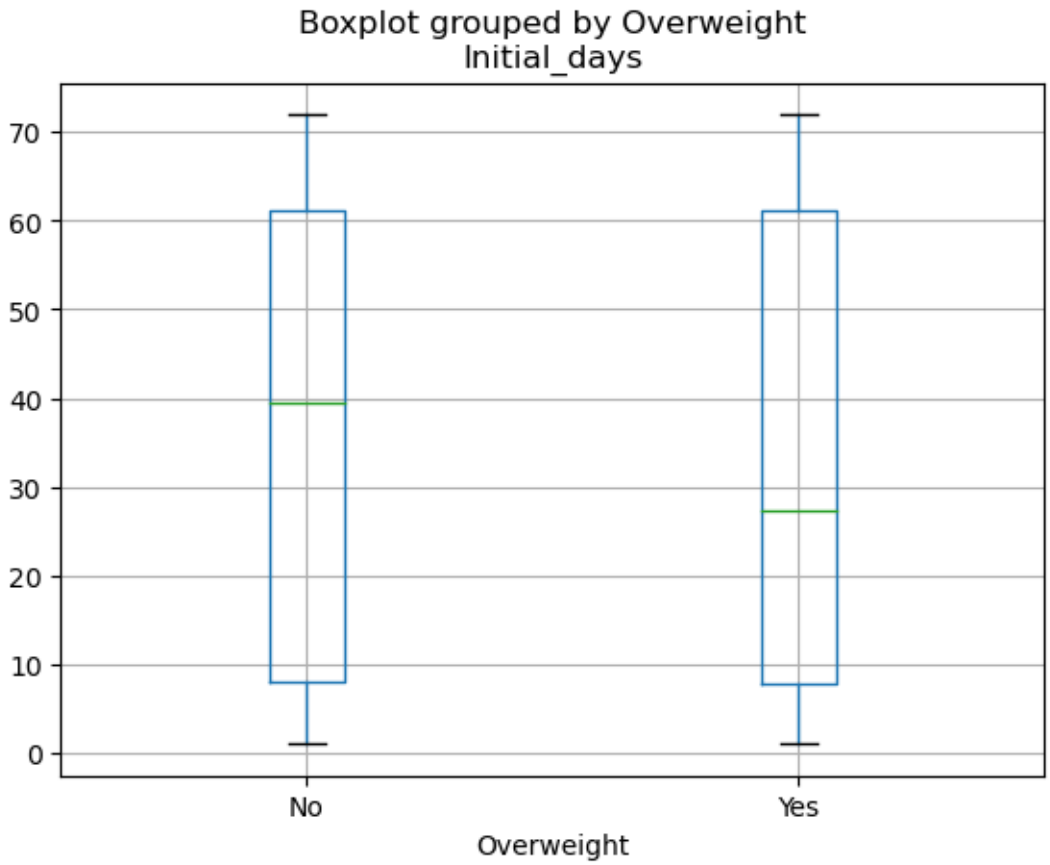
[77]:
```
# BoxPlot of Total Charge and Overweight
focus_df.boxplot(column='TotalCharge', by='Overweight');
```



[78]:
```
# BoxPlot of Initial Days and Readmissions
focus_df.boxplot(column='Initial_days', by='ReAdmis');
```



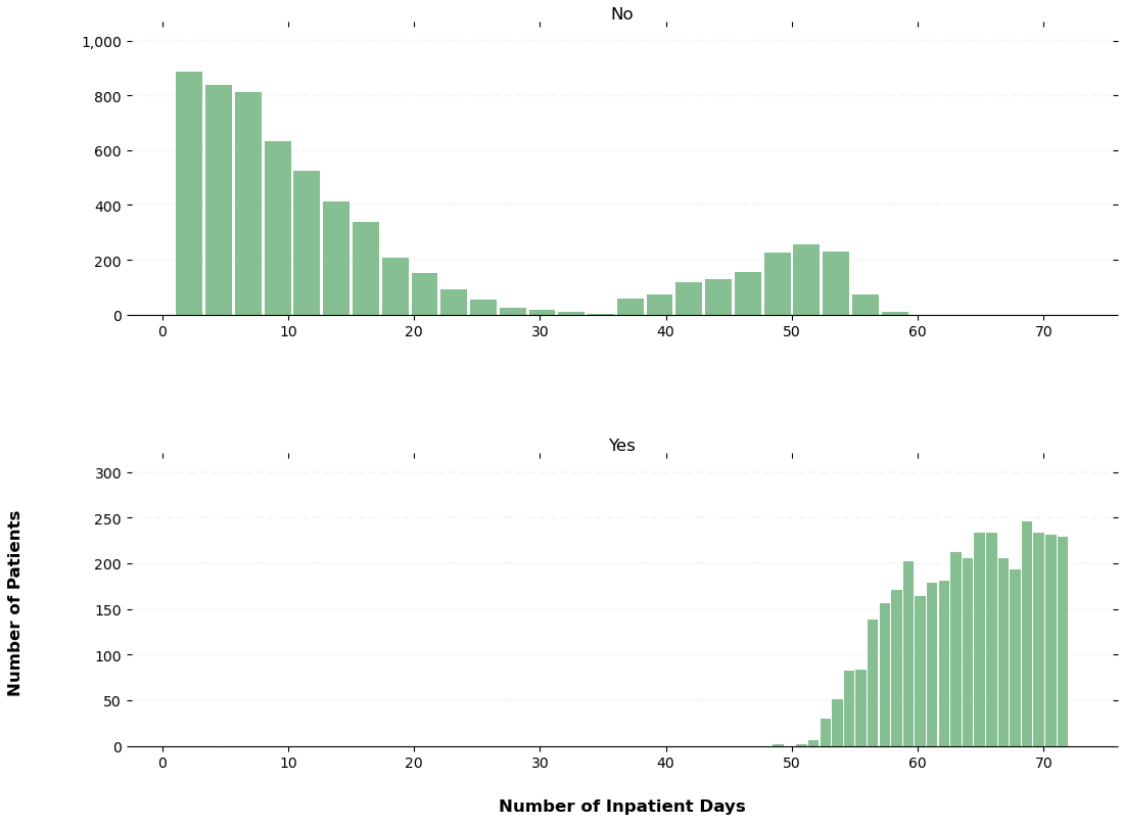
[79]:
```
# BoxPlot of Initial Days and Overweight
focus_df.boxplot(column='Initial_days', by='Overweight');
```



1.1.2 Histograms Comparing Initial Stay of Readmissions (Yes/No)

[80]:
```
#https://mode.com/example-gallery/python_histogram/

ax = focus_df.hist(column='Initial_days', by='ReAdmis', bins=25, grid=False,␣
 ↪figsize=(12,15), layout=(3,1), sharex=True, color='#86bf91', zorder=2,␣
 ↪rwidth=.9)

for i,x in enumerate(ax):

    # Despine
    x.spines['right'].set_visible(False)
    x.spines['top'].set_visible(False)
    x.spines['left'].set_visible(False)

    # Switch off ticks
    x.tick_params(axis="both", which="both", bottom="off", top="off",␣
 ↪labelbottom="on", left="off", right="off", labelleft="on")
```

```
    # Draw horizontal axis lines
    vals = x.get_yticks()
    for tick in vals:
        x.axhline(y=tick, linestyle='dashed', alpha=0.4, color='#eeeeee',␣
↪zorder=1)

    # Set x-axis label
    x.set_xlabel("Number of Inpatient Days", labelpad=20, weight='bold',␣
↪size=12)

    # Set y-axis label
    if i == 1:
        x.set_ylabel("Number of Patients", labelpad=50, weight='bold', size=12)

    # Format y-axis label
    x.yaxis.set_major_formatter(StrMethodFormatter('{x:,g}'))

    x.tick_params(axis='x', rotation=0)
```



Calculate Average Initial Days when Readmission = Yes

[81]:
```
def calc_average(values):
    average = 0
    for value in values:
        average += value
    return average / len(values)

init_days_df = focus_df.loc[focus_df['ReAdmis'] == 'Yes']
init_days_readmis = calc_average(init_days_df['Initial_days'])
print(init_days_readmis)
```

```
63.85950718996995
```

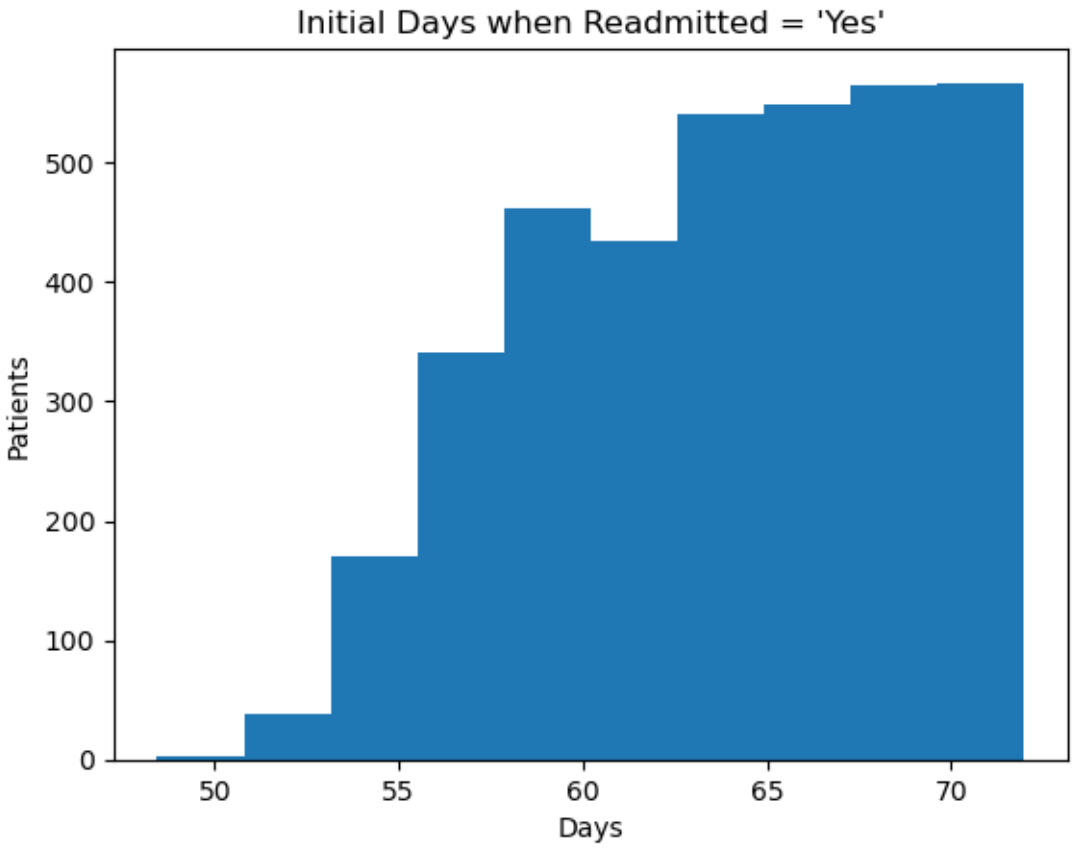
Calculate Average Initial Days when Readmission = No

[82]:
```
def calc_average(values):
    average = 0
    for value in values:
        average += value
    return average / len(values)

init_days_df = focus_df.loc[focus_df['ReAdmis'] == 'No']
init_days_df_readmis = calc_average(init_days_df['Initial_days'])
print(init_days_df_readmis)
```

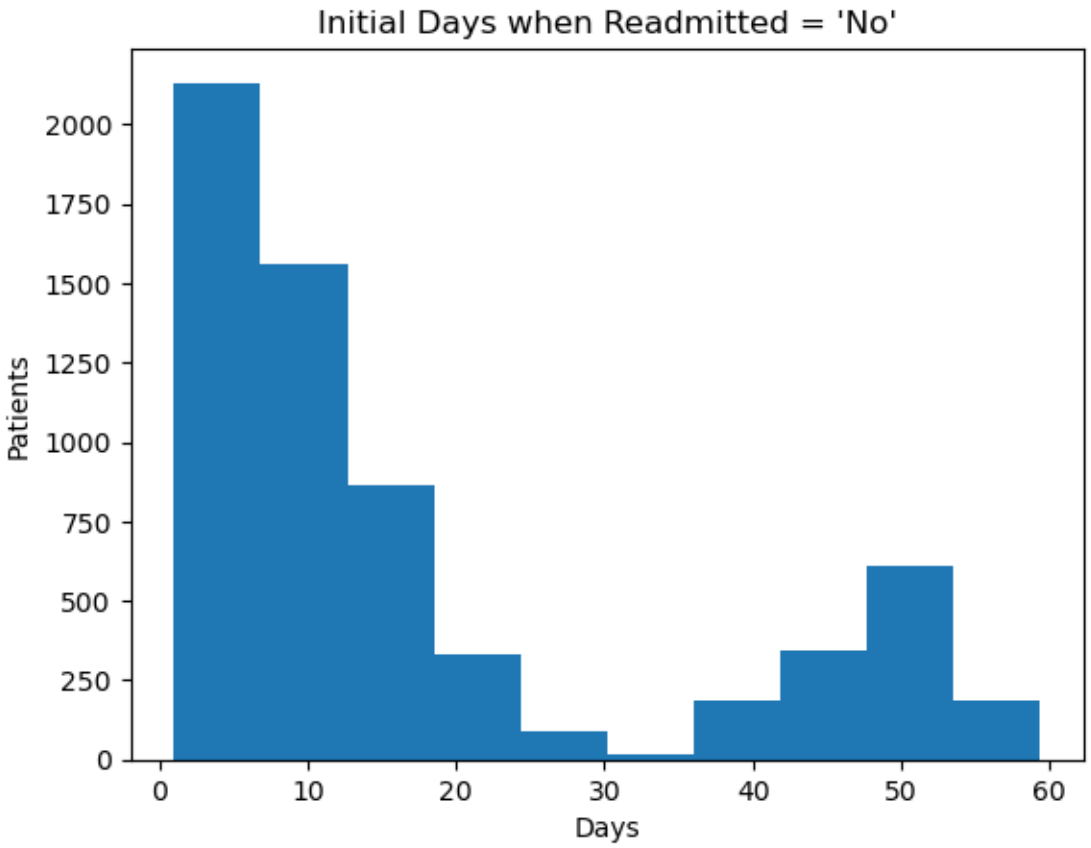
```
17.4146992227966
```

Histogram of Readmissions and Initial Days

[83]:
```
# Bivariate: When readmissions did happen?
readmis_df = focus_df.loc[focus_df['ReAdmis'] == 'Yes', 'Initial_days'].plot.
 ↪hist(bins=10)

plt.xlabel('Days')
plt.ylabel('Patients')
plt.title('Initial Days when Readmitted = \'Yes\'');
```



[84]:
```
# Bivariate: When readmissions didn't happen
readmis_df = focus_df.loc[focus_df['ReAdmis'] == 'No', 'Initial_days'].plot.
 ↪hist(bins=10)

plt.xlabel('Days')
plt.ylabel('Patients')
plt.title('Initial Days when Readmitted = \'No\'');
```


Initial Days when Readmitted = 'No'
Patients
2000
1750
1500
1250
1000
750
500
250
0
0
10
20
30
40
50
60
Days