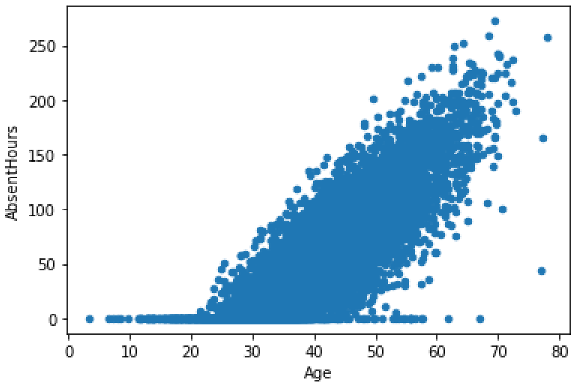
In [1]:
```python
# Musterlösung Aufg. 3 LE1 llr
```

In [2]:
```python
# libraries laden
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Read data from file 'filename.csv'
#https://www.kaggle.com/HRAnalyticRepository/absenteeism-dataset
data = pd.read_csv("employees.csv")
# Preview the first 5 lines of the loaded data
data.head()
```

Out[2]:

	EmployeeNumber	Surname	GivenName	Gender	City	JobTitle	DepartmentName	StoreLocation	Division	Age
0	1	Gutierrez	Molly	F	Burnaby	Baker	Bakery	Burnaby	Stores	32.02
1	2	Hardwick	Stephen	M	Courtenay	Baker	Bakery	Nanaimo	Stores	40.32
2	3	Delgado	Chester	M	Richmond	Baker	Bakery	Richmond	Stores	48.82
3	4	Simon	Irene	F	Victoria	Baker	Bakery	Victoria	Stores	44.59
4	5	Delvalle	Edward	M	New Westminster	Baker	Bakery	New Westminster	Stores	35.69

In [3]:
```python
#Daten anzeigen
data.plot.scatter(x='Age',y='AbsentHours')
plt.show()
```

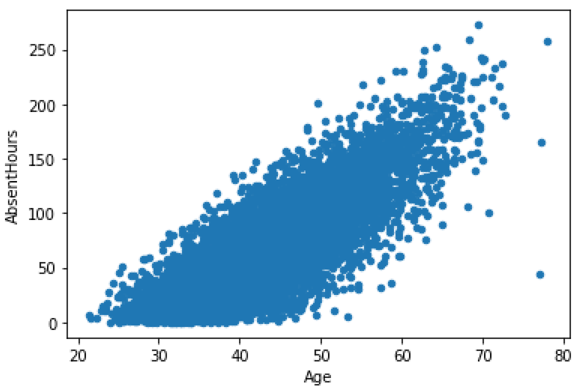


In [4]:
```python
# wir sehen Datenprobleme, da gewisse Absenzen mit 0 h eingetragen wurden.
# hier finden wir sie mittels where
index = np.where(data['AbsentHours'] == 0)
print(index)
```

```
(array([   4,    11,    12, ..., 8325, 8327, 8332]),)
```

In [5]:
```python
# hier werden unbrauchbare Zeilen entfernt
data.drop(data.index[index],inplace=True, axis=0)
```

In [6]:
```python
# Plot der bereinigten Daten
data.plot.scatter(x='Age',y='AbsentHours')
plt.show()
```



Nun benuetzen wir LinearRegression fuer einen Fit

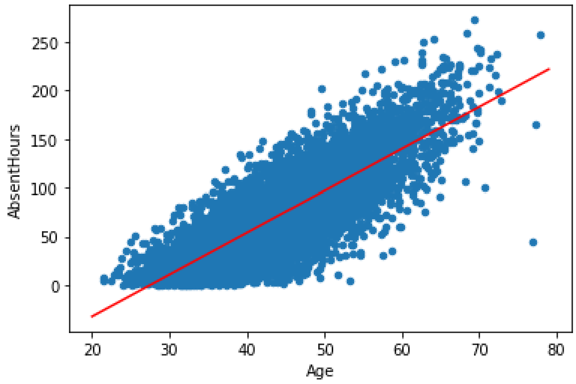
In [7]:
```
from sklearn.linear_model import LinearRegression
```

In [8]:
```
# Fit Alter
age          = data['Age'].values.reshape(-1,1)
absenthours =  data['AbsentHours'].values.reshape(-1,1)
model = LinearRegression().fit(age,absenthours)
# values.reshape ist notwendig wegen depreciation Warnung
```

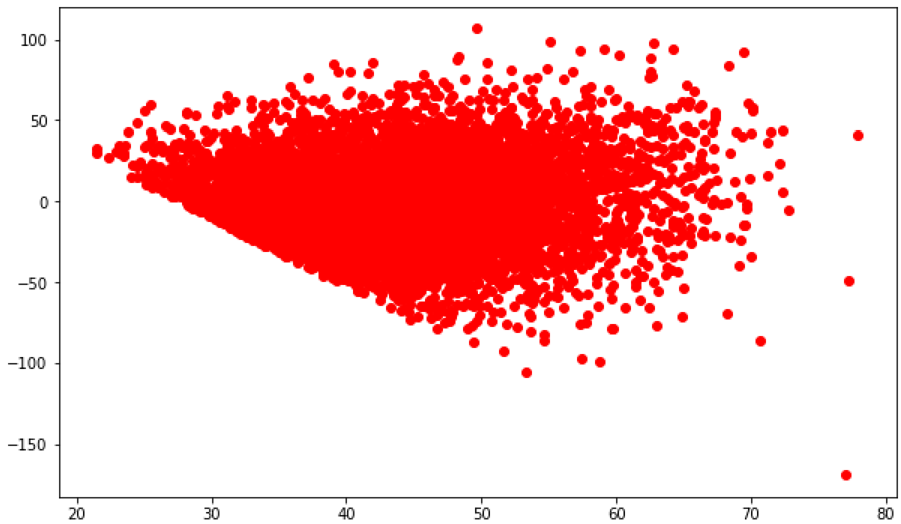
In [9]:
```
#Berechnung R2, slope und intercept
r_sq = model.score(age,absenthours)
print(r_sq)
print('intercept:', model.intercept_)
print('slope:', model.coef_)

#R2 ist 0.65, i.e. nicht perfekt (das waere 1), aber es ist nicht schlecht.
```

```
0.6493487820751211
intercept: [-118.51170857]
slope: [[4.30605265]]
```

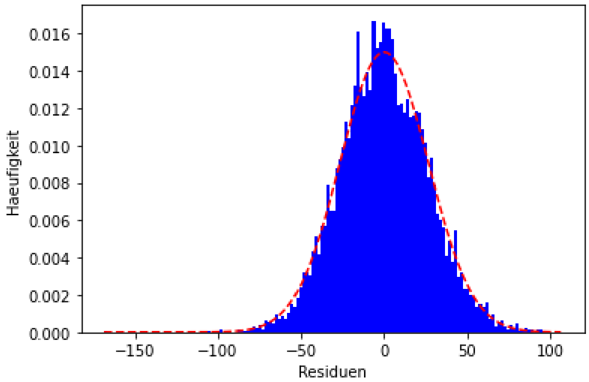
In [10]:
```
#overplot regression line
# x-Achse erstellen von 20-80
x_new = np.arange(60).reshape((-1,1))+20
y_new = model.predict(x_new)

data.plot.scatter(x='Age',y='AbsentHours')
plt.plot(x_new, y_new, color='r')
plt.show()
```



In [11]:
```
#Residuen

fig = plt.figure(figsize=(10,6))

#Vorhersage durchfuehren und Residuen berechnen
predictedabsence = model.predict(age)
resid = absenthours-predictedabsence #Residuen

plt.plot(age, resid, 'o',color='r')
plt.show()
```

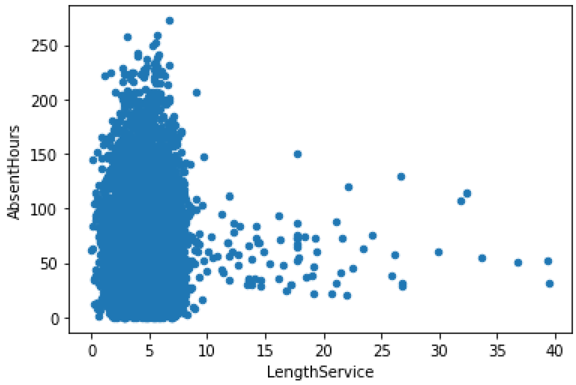


In [12]:
```
# Test ob Residuen normalverteilt via Histogramm
#import matplotlib.mlab as mlab -> funktioniert in neueren python Versionen nicht mehr
from scipy.stats import norm # neu statt mlab

n, bins, patches = plt.hist(resid, bins=150, facecolor='blue',stacked=True,density=True)
plt.xlabel('Residuen')
plt.ylabel('Haeufigkeit')

mu = np.average(resid)
sigma = np.std(resid)

#normalverteilung mit diesem Mittelwert und Standardabweichung
y = norm.pdf(bins, mu, sigma)
plt.plot(bins, y, 'r--')

plt.show()

#dieses Histogram ist gut genug, um als normalverteilt zu gelten.
```



Fit der Laenge der Beschaeftigung

In [13]:
```
#plot of service length
data.plot.scatter(x='LengthService',y='AbsentHours')
plt.show()
# man sieht, dass wohl keine Korrelation existiert
```



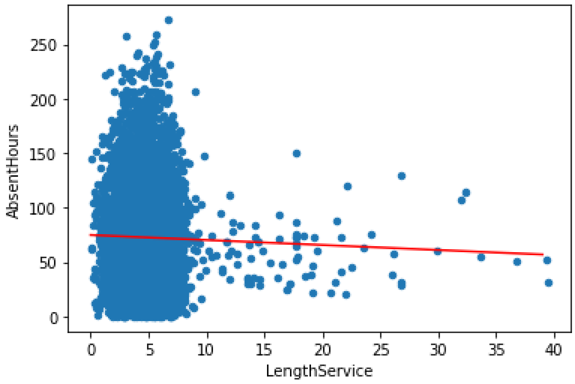
In [14]:
```
service = data['LengthService'].values.reshape(-1,1)
model2  = LinearRegression().fit(service, absenthours)
```

In [15]:
```
#Berechnung R2, slope und intercept
r_sq = model2.score(service, absenthours)
print(r_sq)
print('intercept:', model2.intercept_)
print('slope:', model2.coef_)

#R2 ist 0.001, i.e. das Modell ist sehr schlecht.
```

```
0.0005010210095048873
intercept: [74.99104238]
slope: [[-0.46065175]]
```

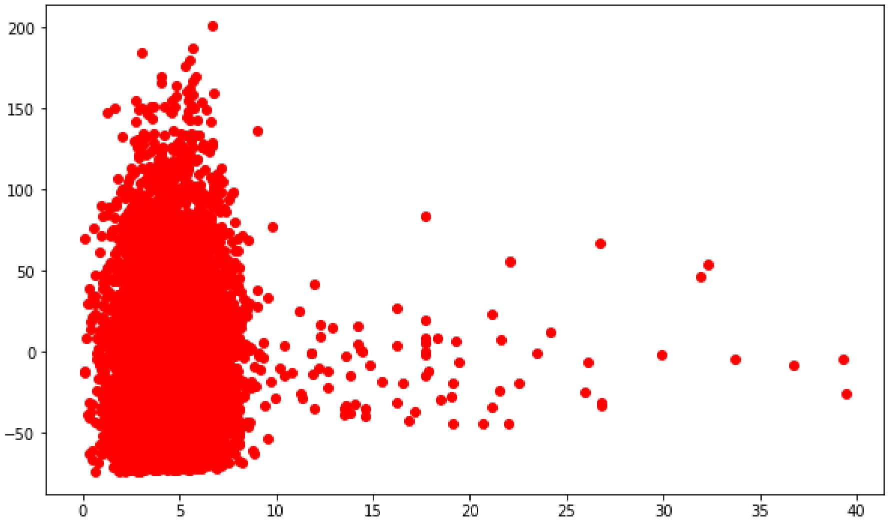
In [16]:

```
#overplot regression line
# X Achse erstellen fuer 0-40
x_new2 = np.arange(40).reshape((-1,1))
y_new2 = model2.predict(x_new2)

data.plot.scatter(x='LengthService',y='AbsentHours')
plt.plot(x_new2, y_new2, color='r')
plt.show()
```

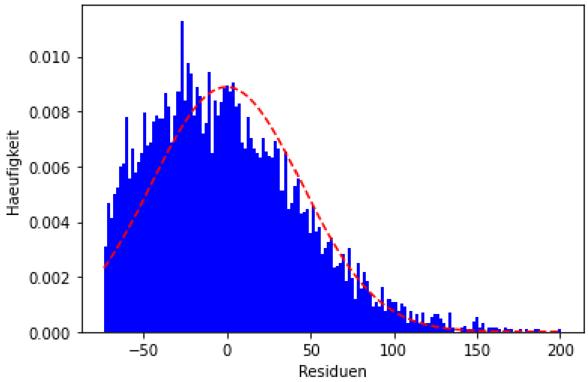


In [17]:

```
#Residuen

fig = plt.figure(figsize=(10,6))

#Residuen berechnen
predictedabsence2 = model2.predict(service)
resid2 = absenthours-predictedabsence2

plt.plot(service, resid2, 'o',color='r')
plt.show()
```



In [18]:
```
n, bins, patches = plt.hist(resid2, 150, facecolor='blue',stacked=True,density=True)
plt.xlabel('Residuen')
plt.ylabel('Haeufigkeit')

mu = np.average(resid2)
sigma = np.std(resid2)

#normalverteilung mit diesem Mittelwert und Standardabweichung
y = norm.pdf(bins, mu, sigma)
plt.plot(bins, y, 'r--')

plt.show()

#dieses Histogram ist nicht normalverteilt.
```



Mittels OLS kann man mehr Output zur Qualitaet der Fits darstellen. Die Regression ist identisch.

In [19]:
```
#advanced regression (advantage: provides summary of fit)
import statsmodels.api as sm

#add b0 (die Konstante ist by default nicht inbegriffen)
age = sm.add_constant(age)
model = sm.OLS(absenthours, age)  #y zuerst!
results = model.fit()
print(results.summary())
# Der Output ist aehnlich zu R. Man sieht z.B., dass man 7016 Datenpunkte hat,
# 7014 degrees of freedom (d.h. 7016-Steigung-Intercept), und wie dann in LE2
# und LE3 erklaert wird, auch den t-Wert und die probability (P>|t|).
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.649
Model:                            OLS   Adj. R-squared:                  0.649
Method:                 Least Squares   F-statistic:                 1.299e+04
Date:                Fri, 15 Oct 2021   Prob (F-statistic):               0.00
Time:                        16:52:34   Log-Likelihood:                -32973.
No. Observations:                7016   AIC:                         6.595e+04
Df Residuals:                    7014   BIC:                         6.596e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -118.5117      1.709    -69.364      0.000    -121.861    -115.162
x1             4.3061      0.038    113.968      0.000       4.232       4.380
==============================================================================
Omnibus:                       34.863   Durbin-Watson:                   1.985
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               47.129
Skew:                           0.051   Prob(JB):                     5.83e-11
Kurtosis:                       3.388   Cond. No.                         243.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [20]:
```
#zweite Regression (mit der anderen Variablen)
service = sm.add_constant(service)
model = sm.OLS(absenthours, service)  #y zuerst!
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     3.516
Date:                Fri, 15 Oct 2021   Prob (F-statistic):             0.0608
Time:                        16:52:34   Log-Likelihood:                -36648.
No. Observations:                7016   AIC:                         7.330e+04
Df Residuals:                    7014   BIC:                         7.331e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         74.9910      1.279     58.640      0.000      72.484      77.498
x1            -0.4607      0.246     -1.875      0.061      -0.942       0.021
==============================================================================
Omnibus:                      470.857   Durbin-Watson:                   2.036
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              568.673
Skew:                           0.681   Prob(JB):                    3.27e-124
Kurtosis:                       3.298   Cond. No.                         12.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]: