Syntax Masters

CAPM - Capital Asset Pricing Model

Idea of the program:

Finding an entry price for investment into a chosen stock

In finance, the capital asset pricing model (CAPM) is a model used to determine a theoretically appropriate required rate of return of an asset.

1. Expected Market Return

Expected Market Return = GDP growth (real) + Expected Inflation Rate

2. Risk Premium for the Investor

Risk Premium = Expected Market Return - Risk Free Rate

3. Required Rate Of Return

Required Rate of Return = Risk Premium x Beta

In finance, the capital asset pricing model (CAPM) is a model used to determine a theoretically appropriate required rate of return of an asset.

4. Percentage Of Retained Earnings

Percentage of Retained Earnings = (Earnings per Share - Dividend) / Earnings per Share

5. Dividend Growth Rate

Growth Rate = Return on Equity(In Percent) x Percentage of Retained Earnings

In finance, the capital asset pricing model (CAPM) is a model used to determine a theoretically appropriate required rate of return of an asset.

6. Value of the Stock

Value = Dividend / (Required Rate of Return - Growth Rate)

Used Modules:

In [0]:
```
#Loading Web-Pages:
from urllib.request import urlopen #several modules for working with URLs
import requests                    #allows to send HTTP requests easily

#Handling HTML
from bs4 import BeautifulSoup      #pulling data from HTML and XML files
from lxml import html              #processing XML and HTML

#Direct Access to Financial Data
import yfinance as yf              #Yahoo! Finance market data downloader

#2D2 Module
import streamlit as st             #Streamlit turns data scripts into web apps
```

Scraping from the net

soup.select_one

function finds only the first tag that matches a selector

In [28]:
```
html = urlopen("https://www.marketwatch.com/investing/bond/tmbmkde-10y?countrycode=bx").read()
soup = BeautifulSoup(html, features="lxml")

Tenyearbond = soup.select_one('body > div.container.container--body > div.region.region--intraday > div.column.
Tenyearbond
```

Out[28]:
```
'-0.292%'
```

Scraping economic data with selection of Forcast Period

In [29]:
```
html = urlopen ("https://ec.europa.eu/info/business-economy-euro/economic-performance-and-forecasts/economic-pe
soup = BeautifulSoup(html, features="lxml")

yearGDp = 2023
if yearGDp >= 2020:
    GDPgrowth = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section > div
    Inflationrate = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section >
elif yearGDp >= 2021:
    GDPgrowth = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section > div
    Inflationrate = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section >
elif yearGDp >= 2022:
    GDPgrowth = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section > div
    Inflationrate = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section >

else:
    if yearGDp <= 2020 or yearGDp >= 2022:
        print ('Please choose a year between 2020 and 2022!')

        # pauses Streamlit with a request to choose a year within limits
        # st.warning ('Please choose a year between 2020 and 2022!')
        # st.stop()
```

```
Please choose a year between 2020 and 2022!
```

Converting scraped data

In [28]:
```
html = urlopen("https://www.marketwatch.com/investing/bond/tmbmkde-10y?countrycode=bx").read()
soup = BeautifulSoup(html, features="lxml")
Tenyearbond = soup.select_one('body > div.container.container--body > div.region.region--intraday > div.column.
print (Tenyearbond)

Tenyearbond = Tenyearbond.replace("%","")
Tenyearbond = float(Tenyearbond)/100
print ('The converted Ten Year Bond (decimal):  ' + str(Tenyearbond))
```

```
-0.292%
The converted Ten Year Bond (decimal):  -0.00292
```

Using yfinance module

Yahoo! Finance market data downloader

https://pypi.org/project/yfinance/

In [122]:
```
stock = 'AAPL'                  #Apple Inc
print (yf.Ticker(stock).dividends)
```

```
Date
1987-05-11    0.000536
1987-08-10    0.000536
1987-11-17    0.000714
1988-02-12    0.000714
1988-05-16    0.000714
                ...
2020-05-08    0.205000
2020-08-07    0.205000
2020-11-06    0.205000
2021-02-05    0.205000
2021-05-07    0.220000
Name: Dividends, Length: 71, dtype: float64
```

Calculations

In [43]:
```
#scraping from 2021 Economic Data Forecast
html = urlopen ("https://ec.europa.eu/info/business-economy-euro/economic-performance-and-forecasts/economic-pe
soup = BeautifulSoup(html, features="lxml")

GDPgrowth = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section > div > di
Inflationrate = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section > div

#converting
GDPgrowth = float(GDPgrowth.replace("," , "."))
Inflationrate = float(Inflationrate.replace("," , "."))

#calculating
# ExpectedMarketReturn = GDP growth (real) +  Expected Inflation Rate
ExpectedMarketReturn = GDPgrowth + Inflationrate
print ("This is the Expected Market Return in 2021 (Euro-Area in Per-Cent): " + str(ExpectedMarketReturn))
```

```
This is the Expected Market Return in 2021 (Euro-Area in Per-Cent): 6.699999999999999
```

In [5]:
```
%%writefile capm.py

from bs4 import BeautifulSoup
from urllib.request import urlopen
import requests
from lxml import html
import yfinance as yf
import streamlit as st

st.header('Syntax Masters')
st.image('./CAPM_Bild.jpg')
st.subheader("This program uses the CAPM (Capital Asset Pricing Model) in order to give our users the ability to

stock = st.text_input("Please write the ticker of the company that you're looking for: ")
if not stock:
    st.warning('Please input a stock ticker!')
    st.stop()
st.success('Thank you for inputting a stock ticker.')
st.write("You are looking for this Stock: ", stock)

yearGDp = st.number_input('Would You like to change the year for the Economic Data (2020-2022)? ', value=2021)
st.write(yearGDp, type(yearGDp))
#yearGDp = st.text_input ("Which year are you looking for?: ")
#yearGDp = int(yearGDp)
st.write("The Will see the Year: ", yearGDp)


headers={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
url = 'https://finance.yahoo.com/quote/'+ stock +'/key-statistics?p=' + stock
resp = requests.get(url, headers=headers, timeout=5).text
soup = BeautifulSoup(resp, features="lxml")

beta = soup.select_one('#Col1-0-KeyStatistics-Proxy > section > div.Mstart(\a\).Mend(\a\) > div.Fl(end).W\(5
#st.write("Beta " +stock+": "+\n' + beta)
companystock = soup.select_one('#quote-header-info > div.My\(6px\).Pos\(r\).smartphone_Mt\(6px\) > div.D\(ib\)
#st.write("Stock: "+stock+" "+companystock)

ReturnOnEquity = soup.select_one('#Col1-0-KeyStatistics-Proxy > section > div.Mstart(\a\).Mend(\a\) > div:nth-c
#print("Return On Equity " +stock+ ": " \n' + ReturnOnEquity)

DividendRate = soup.select_one('#Col1-0-KeyStatistics-Proxy > section > div.Mstart(\a\).Mend(\a\) > div.Fl(end
#print("Dividend Rate " +stock+ ": " \n' + DividendRate)

EPS = soup.select_one('#Col1-0-KeyStatistics-Proxy > section > div.Mstart(\a\).Mend(\a\) > div:nth-child(3) > d
#print("EPS " +stock+ ": " \n' + EPS)


html = urlopen ("https://ec.europa.eu/info/business-economy-euro/economic-performance-and-forecasts/economic-pe
soup = BeautifulSoup(html, features="lxml")

if yearGDp == 2020:
    GDPgrowth = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section > div
    #st.write("Expected GDP Growth 2020: "\n' + GDPgrowth)

    Inflationrate = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section >
    #st.write("Expected Inflation Rate 2020: "\n' + Inflationrate)
elif yearGDp == 2021:
    GDPgrowth = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section > div
    #st.write("Expected GDP Growth 2021: "\n' + GDPgrowth)

    Inflationrate = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section >
    #st.write("Expected Inflation Rate 2021:"\n' + Inflationrate)
elif yearGDp == 2022:
    GDPgrowth = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section > div
    #st.write("Expected GDP Growth 2022: " \n' + GDPgrowth)

    Inflationrate = soup.select_one('#block-system-main > div > div > div.page-content > div > div > section >
    #st.write("Inflation Rate 2022: "\n' + Inflationrate)
else:
    #st.write("Please choose a year between 2020 and 2022")
    if yearGDp <= 2020 or yearGDp >= 2022:
        st.warning("Please choose a year between 2020 and 2022!")
        st.stop()


html = urlopen("https://www.marketwatch.com/investing/bond/tmbmkde-10y?countrycode=bx").read()
soup = BeautifulSoup(html, features="lxml")

Tenyearbond = soup.select_one('body > div.container.container--body > div.region.region--intraday > div.column.
#st.write("Ten Year Bond: "\n' + Tenyearbond)

#Converting the scraped data from strings to floats
GDPgrowth = float(GDPgrowth.replace("," , "."))
st.write ('')
st.write ('The converted GDP Growth is:                ' + str(GDPgrowth) + '%')

Inflationrate = float(Inflationrate.replace("," , "."))
st.write ('The converted Inflationrate is:             ' + str(Inflationrate) + '%')

Tenyearbond = Tenyearbond.replace("%","")
Tenyearbond = float(Tenyearbond)/100
st.write ('The converted Ten Year Bond (decimal):  ' + str(Tenyearbond))

beta = beta.replace("," , ".")
beta = float(beta)
st.write ('The converted Beta (factor):                ' + str(beta))

EPS = EPS.replace("," , ".")
EPS = float(EPS)
st.write ('The converted Earnings per share (EPS): ' + str(EPS))

DividendRate = DividendRate.replace("," , ".")
DividendPerShare = float(DividendRate)                 #!!! DividendRate is yahoo.finance is dividend per share!
st.write ('The converted Dividend per share:           ' + str(DividendPerShare))

ReturnOnEquity = ReturnOnEquity.replace("%", "")
ReturnOnEquity = ReturnOnEquity.replace("," , ".")
ReturnOnEquity = float(ReturnOnEquity)/100
st.write ('The Return on equity (decimal):             ' + str(ReturnOnEquity))
st.write ('')


#CALCULATIONS
# Calculate Expected Market Return
# ExpectedMarketReturn = GDP growth (real) +  Expected Inflation Rate
ExpectedMarketReturn = GDPgrowth + Inflationrate
st.write("This is the Expected Market Return:                  " + str(ExpectedMarketReturn))

# calculate  Risk Premium for the Investor.
# Risk Premium= ExpMarketReturn - RiskFreeRate
Riskpremium = ExpectedMarketReturn - Tenyearbond
st.write("This is the Risk Premium for the investor:           " + str(Riskpremium))

# calculate Required Rate Of Return
# RequiredRateOfReturn = RiskPremium x Beta
RequiredReturnRate = Riskpremium * beta / 100
st.write("This is the Required Rate of Return (decimal):       " + str(RequiredReturnRate))

# calculate Percentage Of Retained Earnings
# PercentageOfRetainedEarnings = (EarningsPerShare - Dividend) / EarningsPerShare
PercentRetainedEarnings = (EPS - DividendPerShare) / EPS
st.write("This is the Percentage Of Retained Earnings (decimal): " + str(PercentRetainedEarnings))

# calculate Dividend Growth Rate
# growthRate = ReturnOnEquity * PercentageOfRetainedEarnings
DivGrowthRate = ReturnOnEquity * PercentRetainedEarnings
st.write("This is the Dividend Growth Rate (decimal):          " + str(DivGrowthRate))


# calculate Value of the Stock
# Value = dividend / (required rate of return - growth rate)
if DivGrowthRate > (RequiredReturnRate * 0.85):
    st.write ('!!!')
    st.write("The Dividend Growth Rate is to close (85%) or higher than your Required Return Rate.')
    st.write ('The CAPM Model will not work in this case.')
    st.write ('The dividents in the last years were: ')
    st.write (yf.Ticker(stock).dividends)
    st.bar_chart(yf.Ticker(stock).dividends)
    DivGrowthRate = st.text_input ('Please examine the company and type in which Dividend Growth Rate You would
    if not DivGrowthRate:
        st.warning('Please enter your estimate for the Dividend Growth Rate!')
        st.stop()
    st.success('Thank you for your selection!')
DivGrowthRate = float (DivGrowthRate)
StockValue = DividendPerShare / (RequiredReturnRate - DivGrowthRate)
st.write("This is the Stock Value:                             " + str(StockValue))
st.write ("The actual price for " +stock+": "\n' + companystock)
```

```
Overwriting capm.py
```

Challenges

- Web Scraping didn't work and a header had to be added to simulate a Web Browser Request

In [0]:
```
stock = 'AAPL'
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr
url = 'https://finance.yahoo.com/quote/'+ stock +'/key-statistics?p=' + stock
resp = requests.get(url, headers=headers, timeout=5).text
soup = BeautifulSoup(resp, features="lxml")
#soup
```

Challenges

- Web Scraping didn't work and a header had to be added to simulate a Webbrowser Request
- Calculations could not be carried out as key figures did not provide plausible results due to the crisis or "special" dividend strategy of the companies

Conditions:

- All information is given "before" taxes
- Currency is EUR as all Economic Forcast data is based on Euro-Area and Currency Risk Calculations are not included
- Evaluation is restricted to European Companies (growing with European Market)
- Restricted to "Value Companies" as CAPM will not give reliable results on "Growth Companies" or companies with negative financial results

Further development

- Make the Economic Forecast truly global
- Include Currency Risk Calculations
- Develop the scraping method more flexible and analytic to be ready for any Webpage Changes
- Include more interactions with the user to let her adjust the calculated data (like Growth Rate example)
- Include more models and other evaluation tools