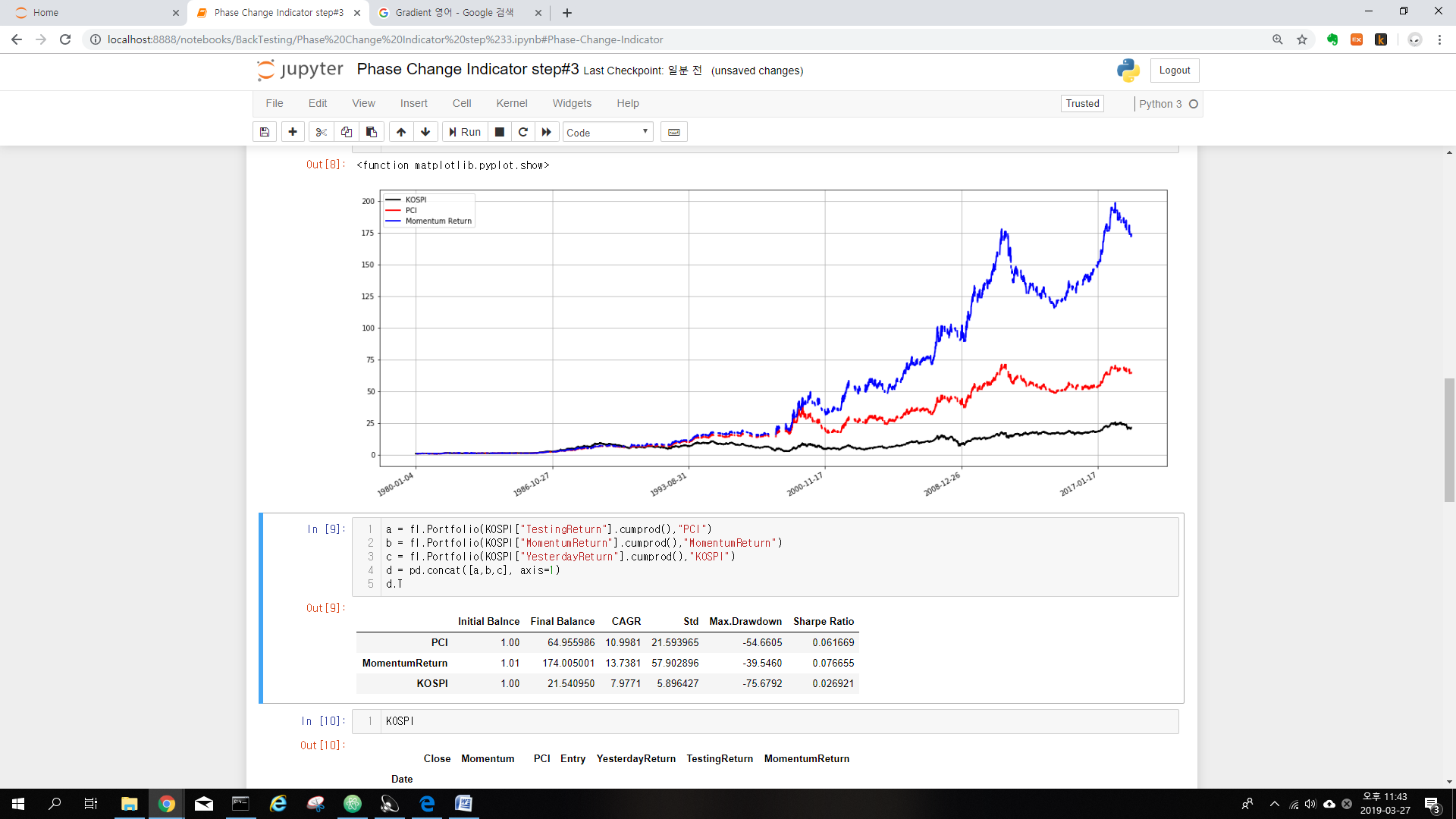
Phase Change Indicator

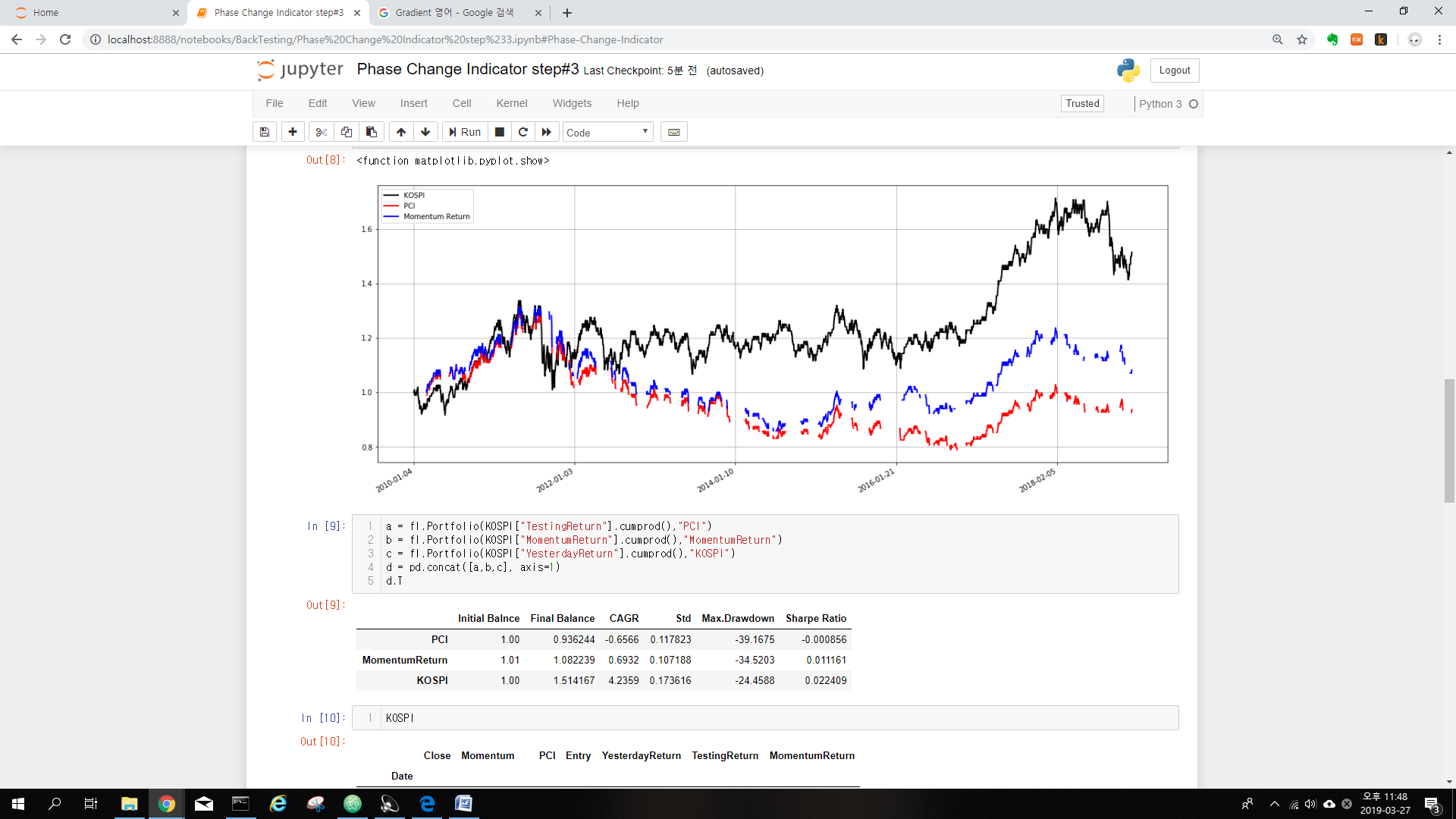
1. 일별데이타로 백테스팅(PCI 산출기간 20일)



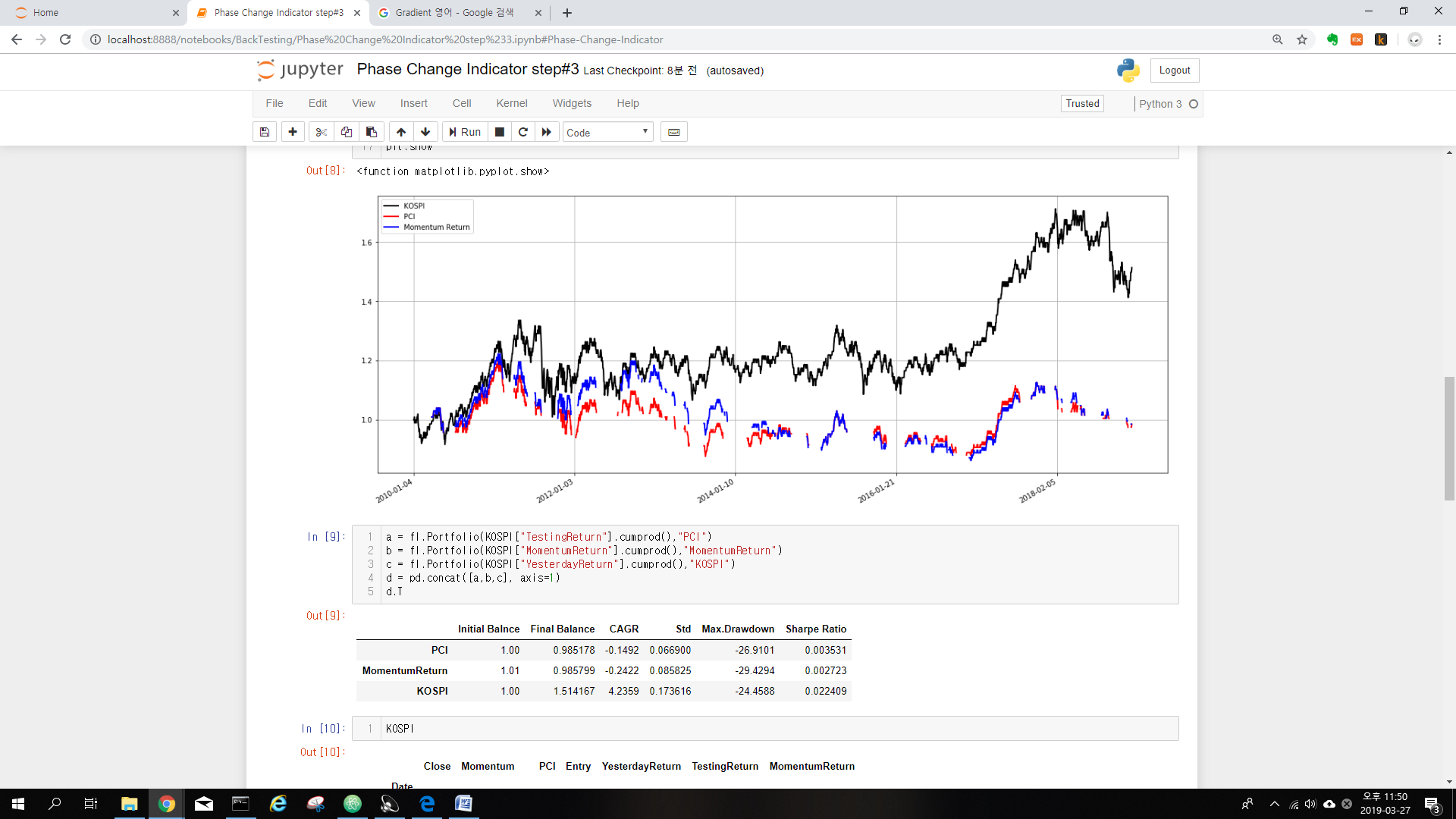
뭔 이런 결과가…

단순 모멘텀보다 성과가 좋지못하다.

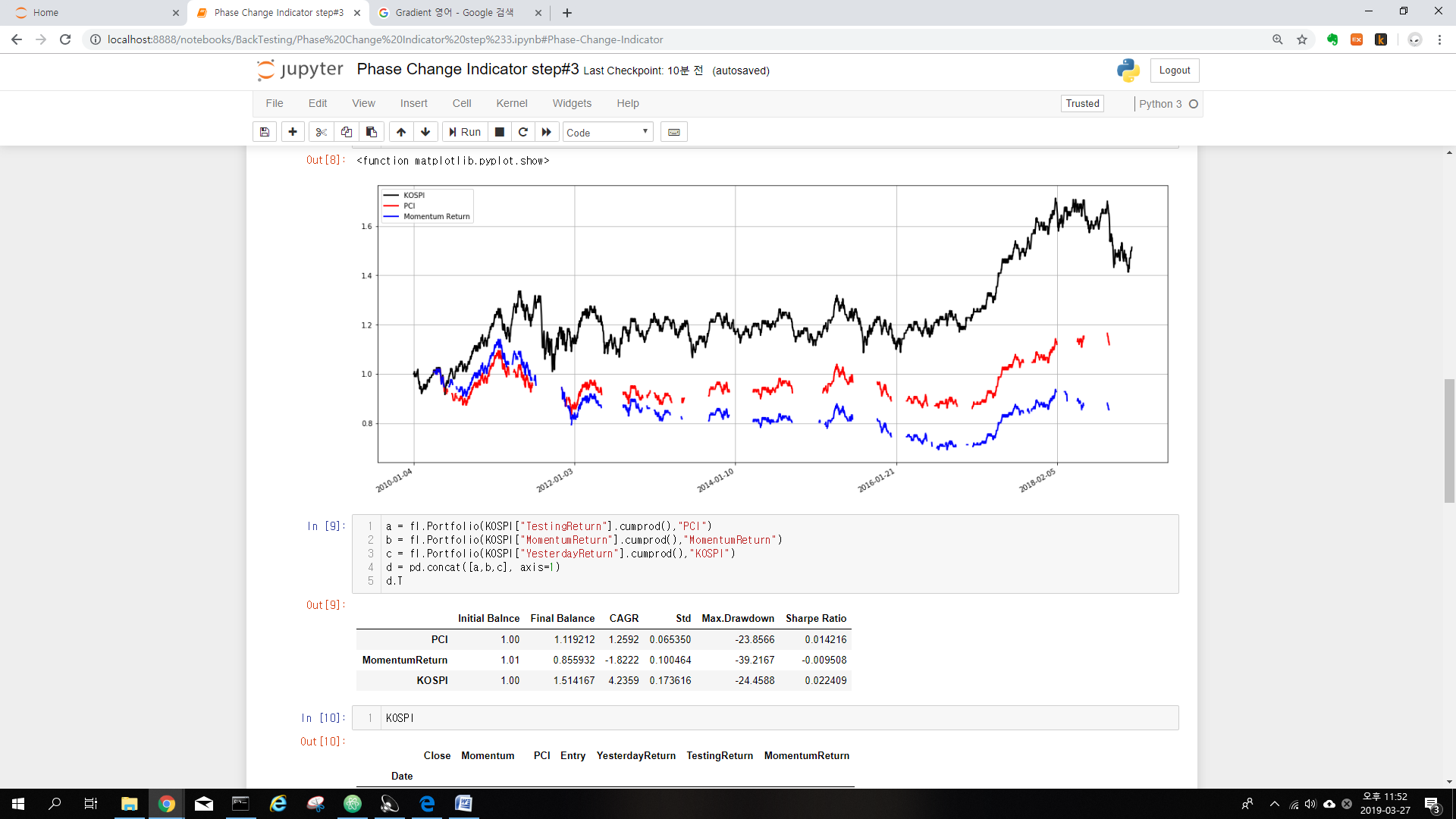
MDD도 크고, 수익률도 떨어지고



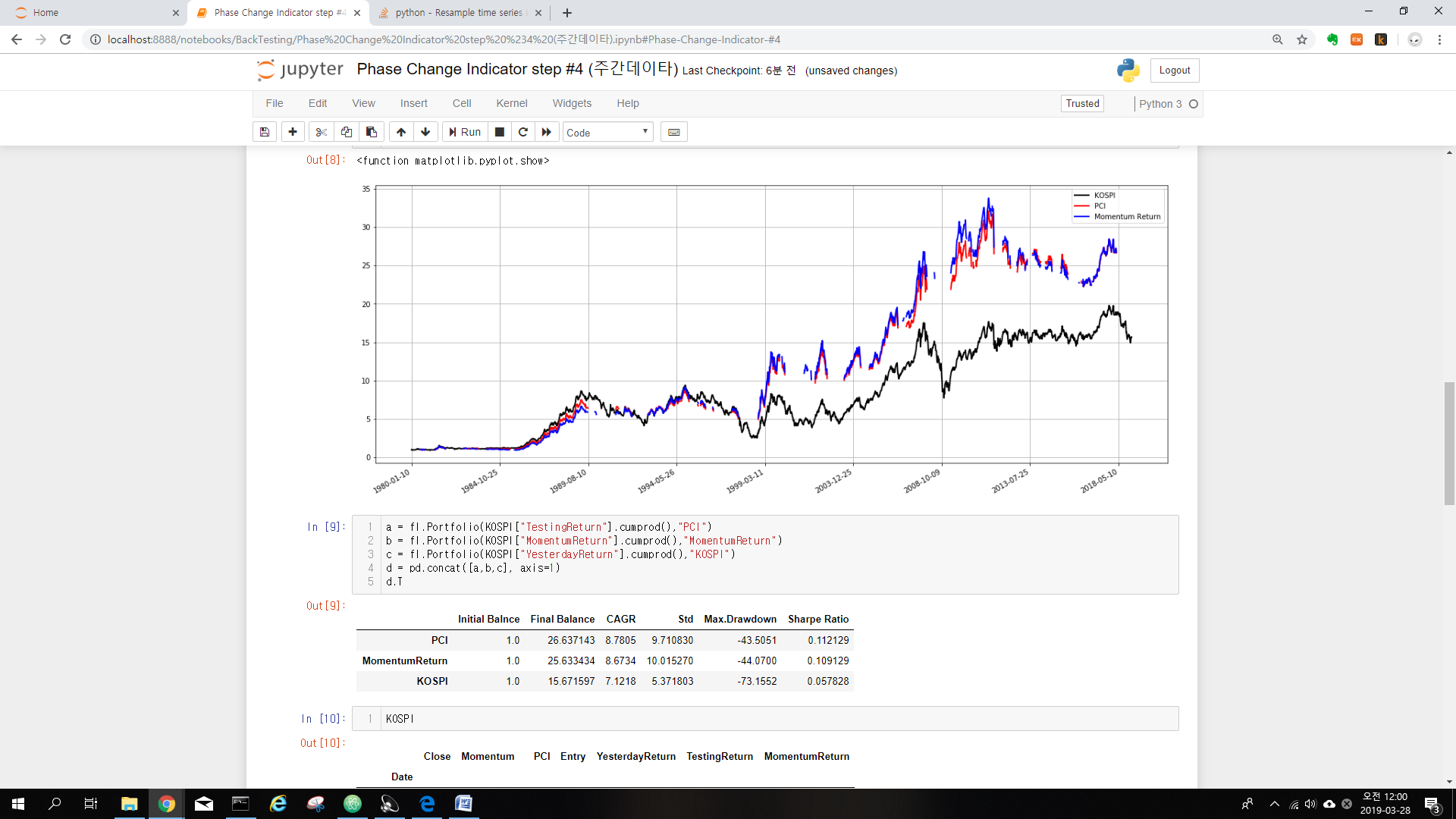
1. 일별데이타로 백테스팅(PCI 산출기간 40일)



1. 일별데이타로 백테스팅(PCI 산출기간 60일)



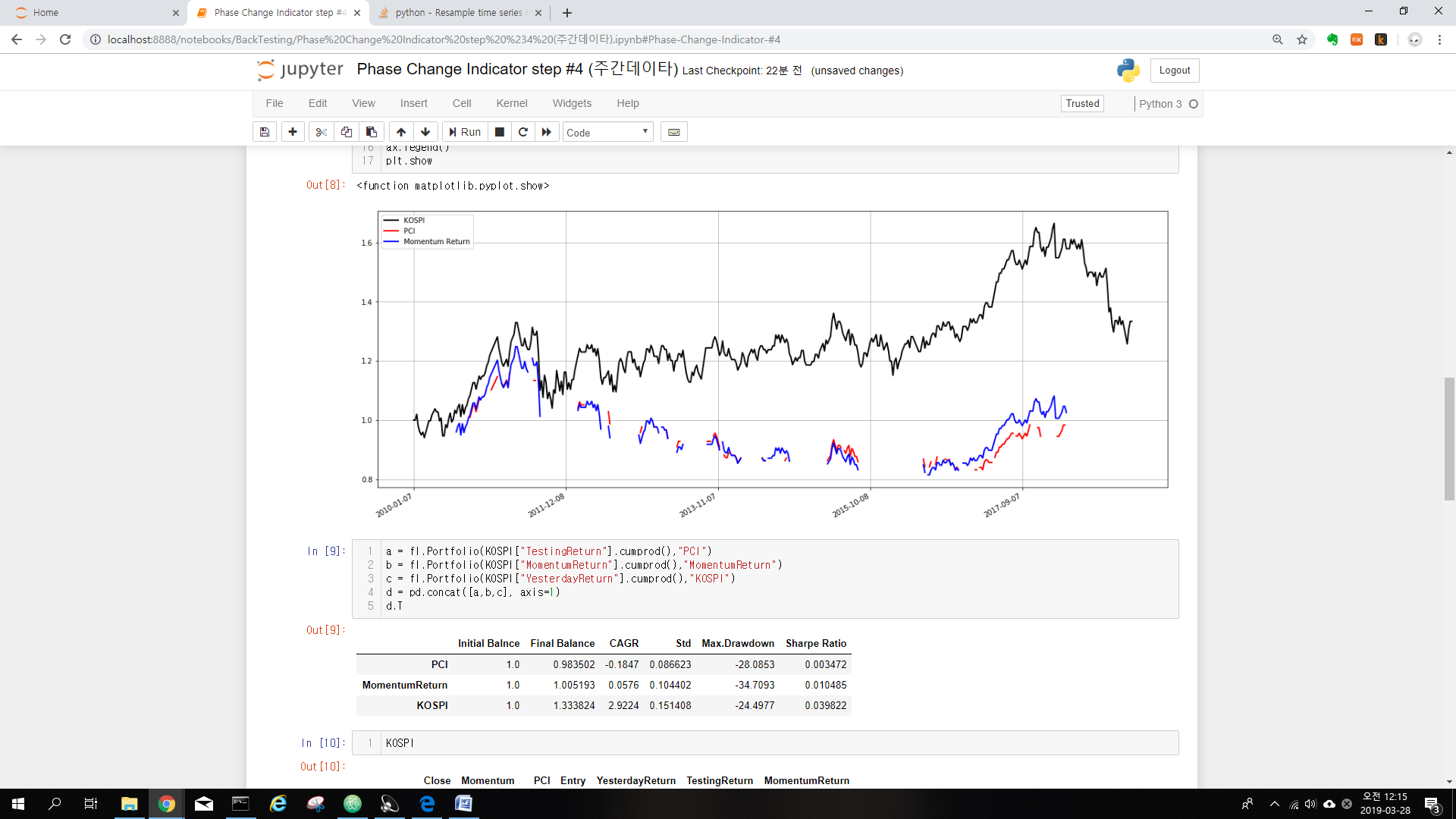
1. 주간데이터로 백테스팅(PCI 산출기간 24주)



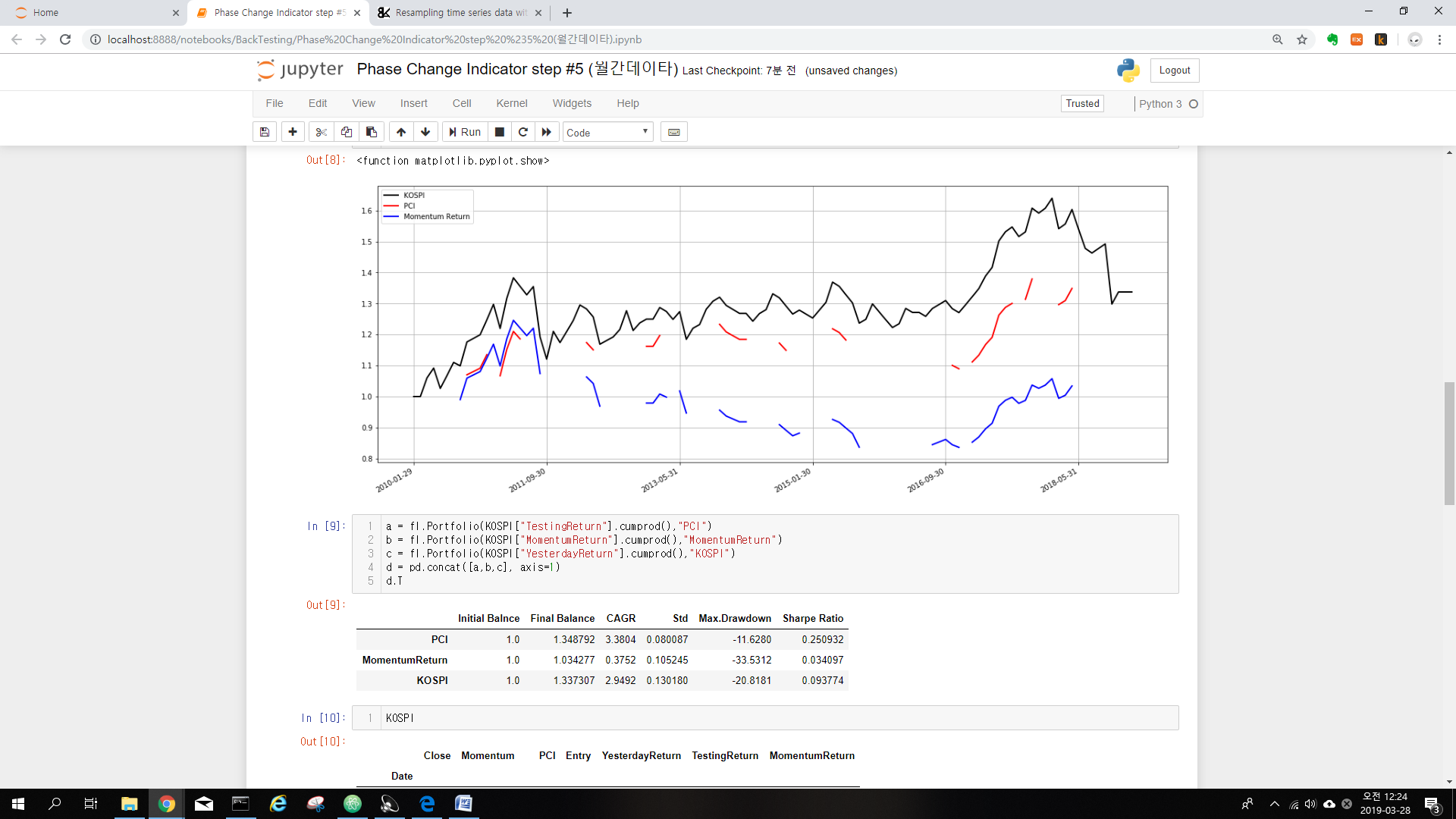
주간데이터로 백테스팅 했을뿐인데 일별데이타와 결과가 확달라진다.

역시 모멘텀(추세추종)은 최소 주간데이타 이상으로 raw data를 만들고 테스트해야 하는 것 같다. 단기 daytrading은 아닌 것 같다.

2010년 시작하면 전략은 망한다.

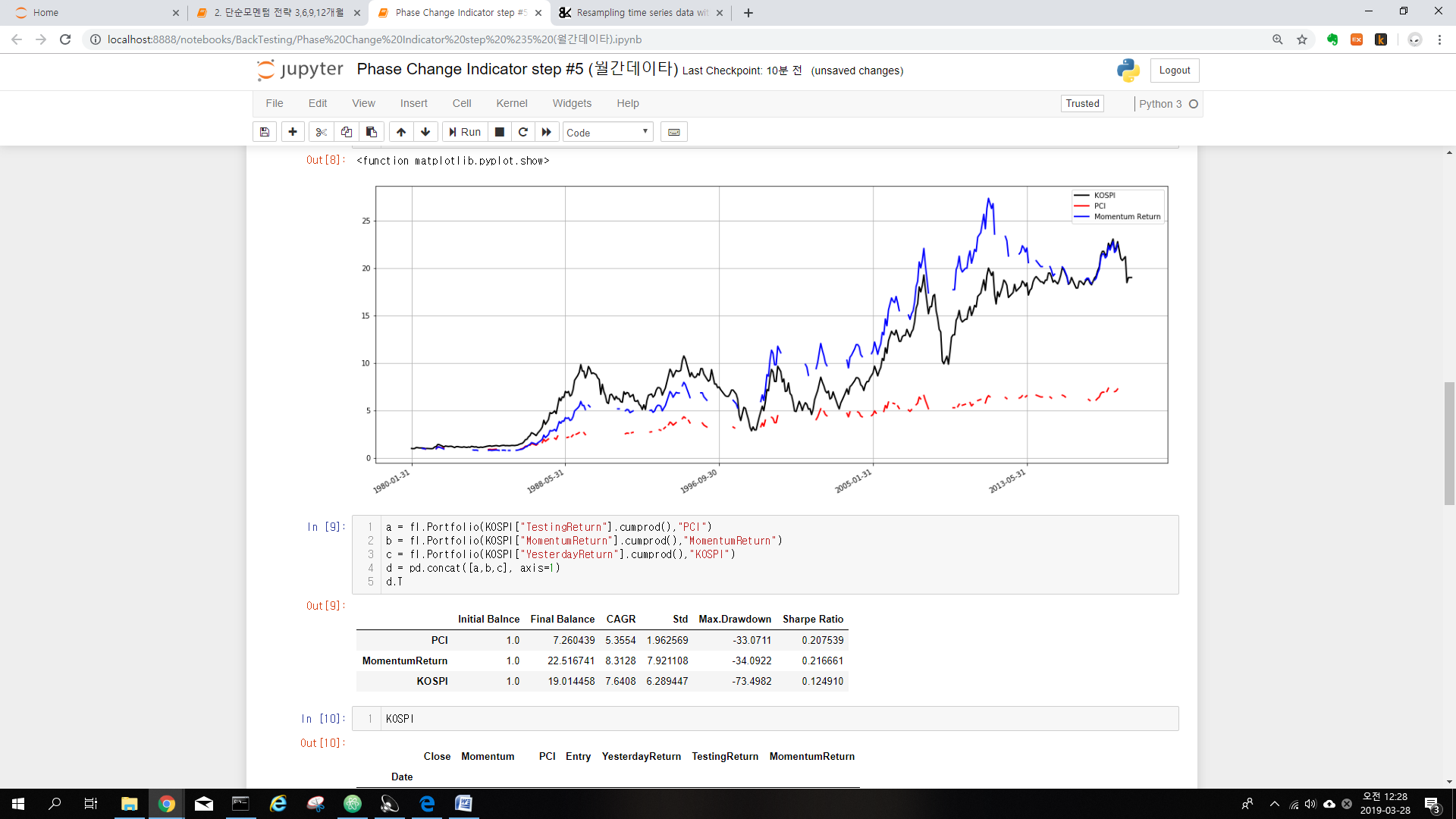


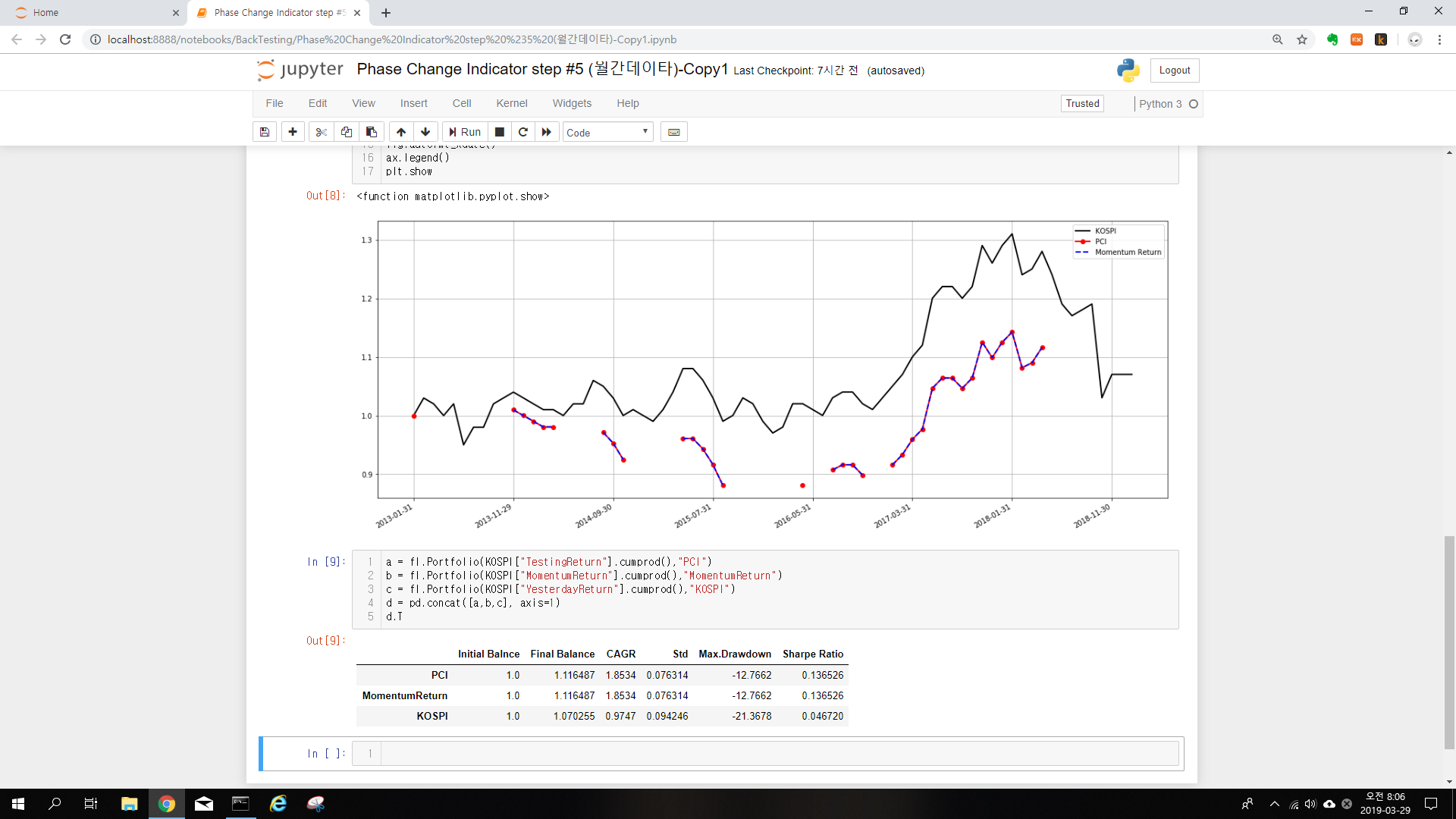
1. 월간데이터로 백테스팅 (PCI 산출기간 6개월)



2010년부터 결과이고…

처음부터 시작하면



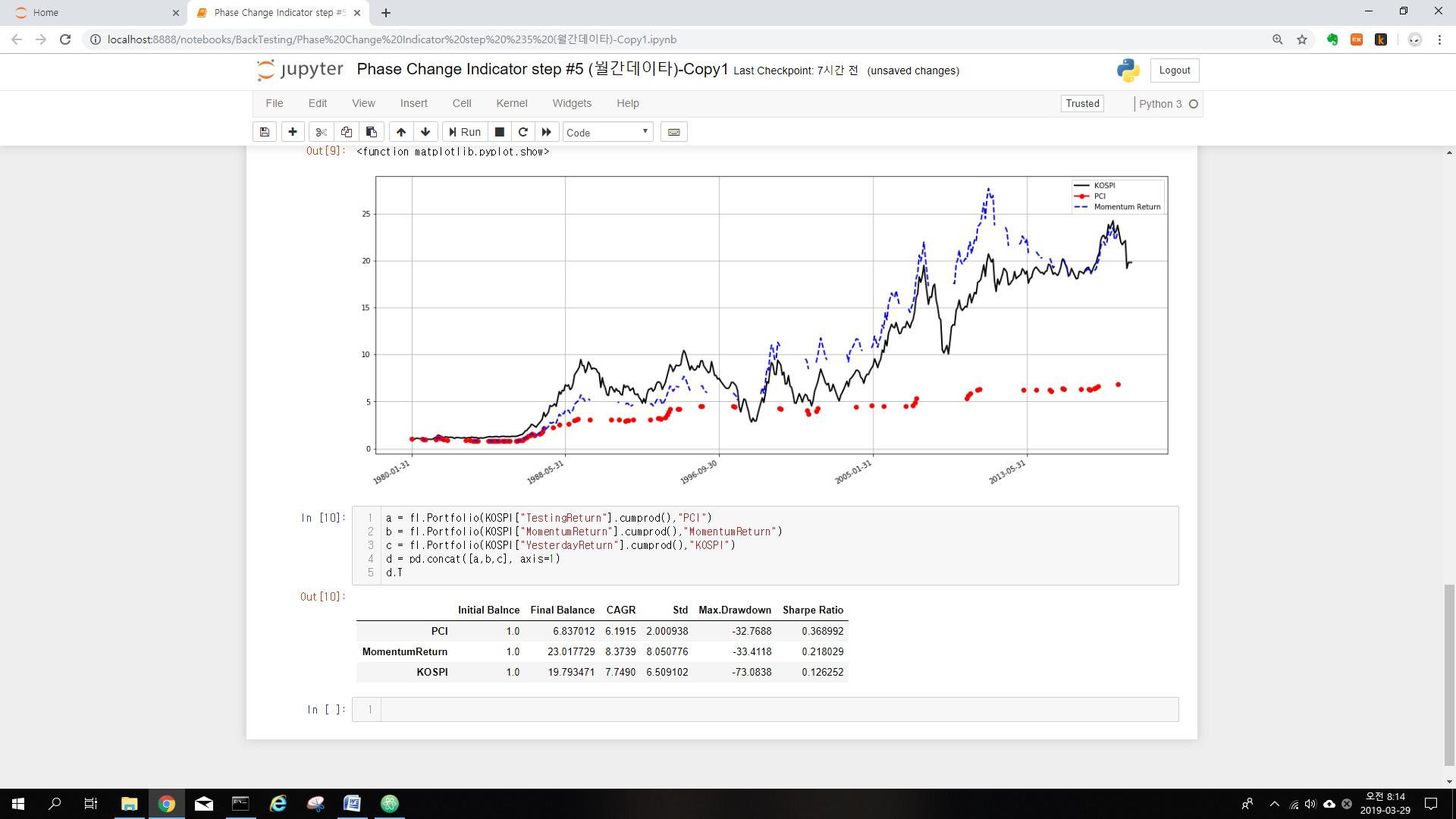


재미있는 결과다.

코드를 이렇게 했더니 단순모멘텀과 똑 같은 결과가 나왔다. 왜? 이런 일이 생길까?

* 단순모멘텀도, PCI도 같은 결과가 나와 놀랍기도 하지만 결국은 같은 로직이라는 것

그러나, 처음부터 백테스팅하면 단순모멘텀보다 못한성과가 나온다

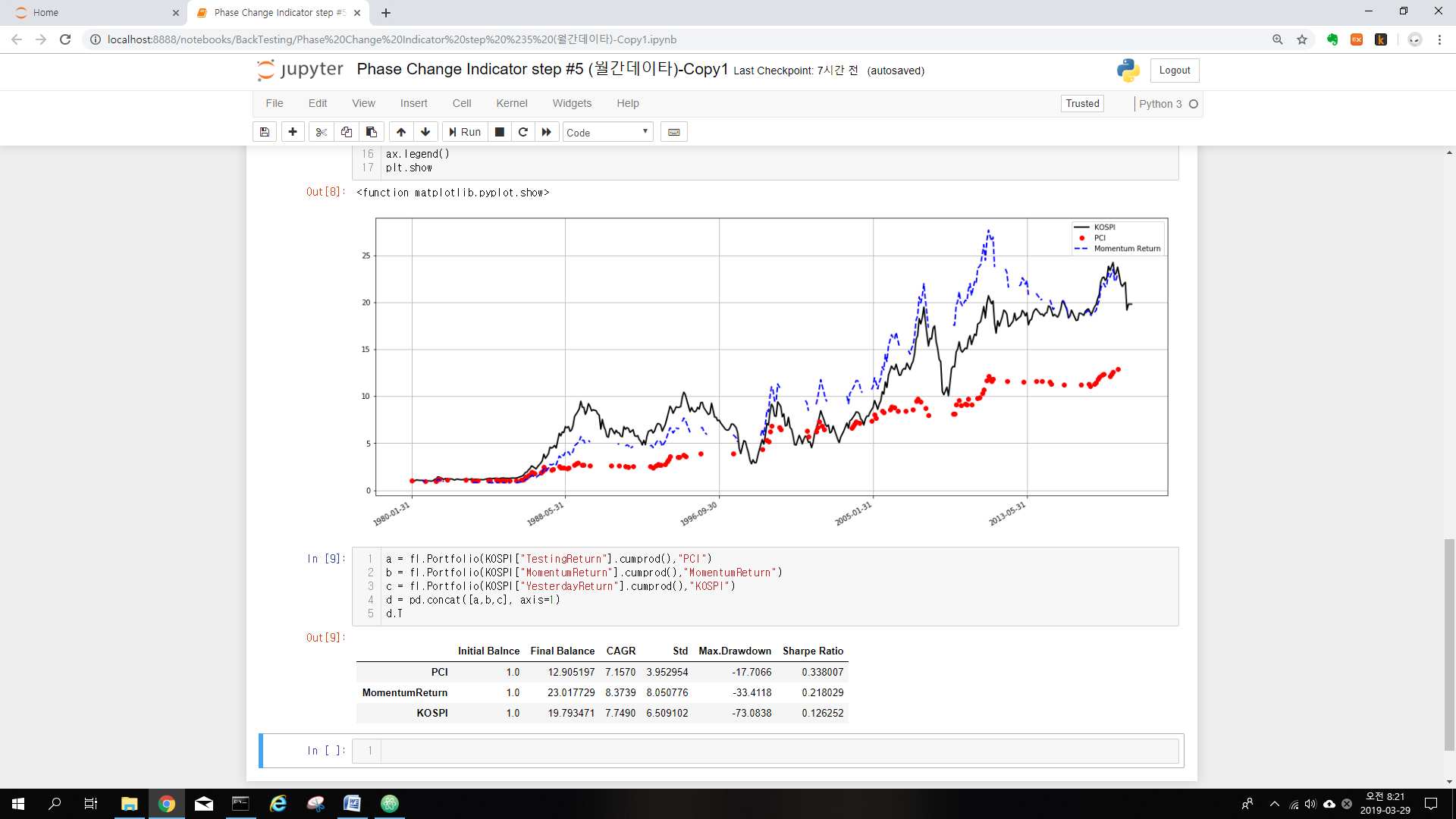


|  |
| --- |
| # Momentum > 0 이고, PCI < 20 일때 매수  # Momentum < 0 이고, PCI > 80 일때 매도  entry = []  for i in range(len(KOSPI)):  if KOSPI["Momentum"][i] > 0 and KOSPI["PCI"][i] < 30: # 하락장에서 상승으로 전환, 횡보에서 상승으로 전환  #entry.append("Buy")  entry.append(1)  elif KOSPI["Momentum"][i] < 0 and KOSPI["PCI"][i] > 70: # 상승장에서 하락으로 전환, 횡보에서 하락으로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] < 0 and KOSPI["PCI"][i] < 30: # 하락장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] > 0 and KOSPI["PCI"][i] > 70: # 상승장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  else:  entry.append(np.NaN) # 20과 80사이는 상태를 정의할 수 없다. 아래 코드에서 이전 상태를 유지 한다.  KOSPI["Entry"] = entry  KOSPI["Entry"][0] = 0 # 첫번째 값을 "0"으로 세팅하여 현재 매도 상태임을 명시  #KOSPI["Entry"] = KOSPI["Entry"].fillna(method="ffill") # NaN은 횡보임으로 앞의 값으로 채움 |

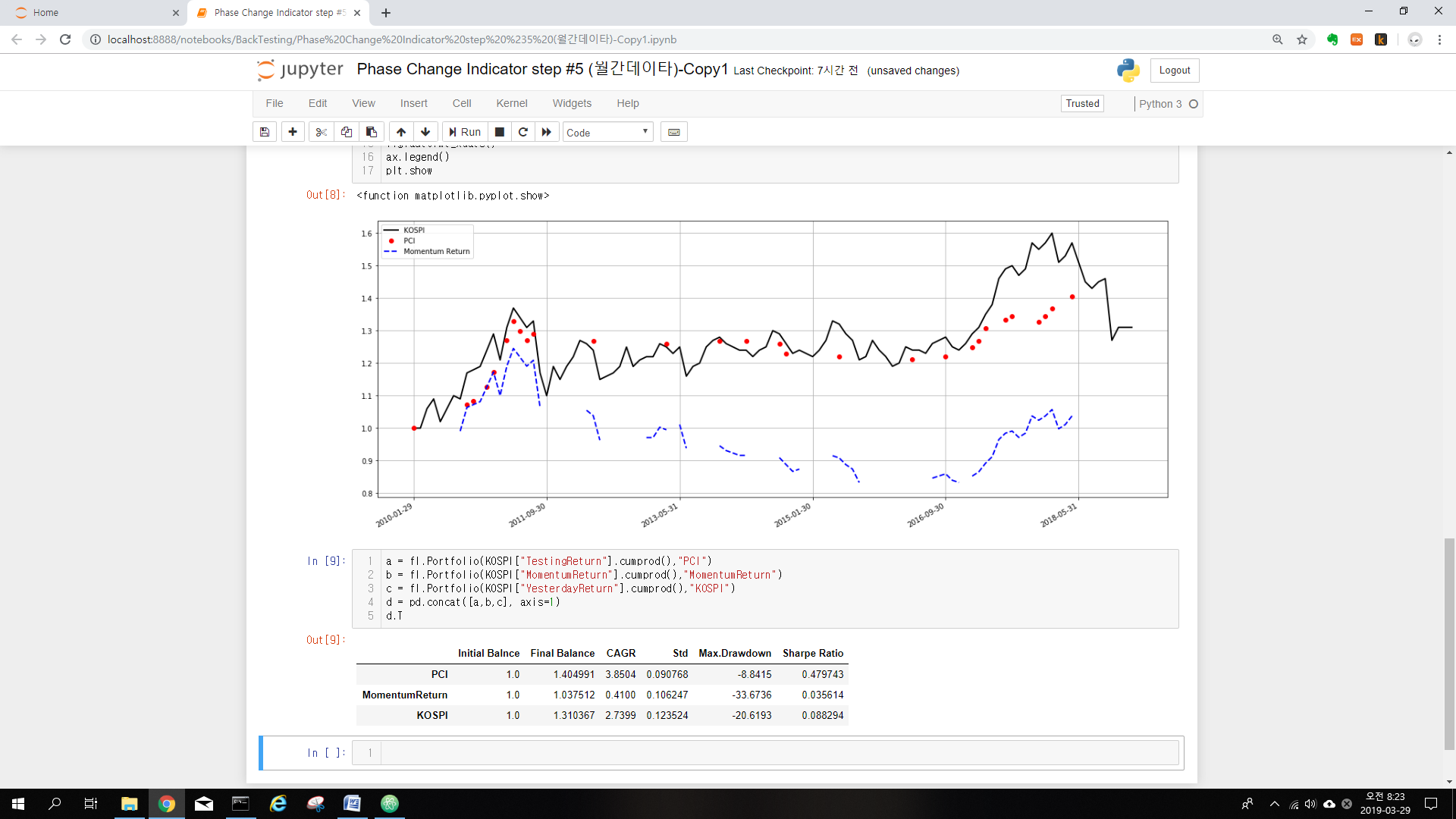
|  |
| --- |
| # 절대모멘텀  def SimpleMomentum(data,period):  # shift를 사용하여 원화는 달의 수익률을 구한다  # 당월주가 > n 개월 전 주가(n개월 모멘텀 > 0) --> 주식매수  # 당월주가 < n 개월 전 주가(n개월 모멘텀 < 0) --> 주식매도  result = np.where(data > data.shift(period),1,np.NaN)  return pd.Series(result,index=data.index) |

|  |
| --- |
| # PCI 구하기  def PhaseChangeIndicator(data,Period=20):  P = Period - 1  result = 0  for i in range(Period+1):  result += np.where(data > data.shift(i),1,0)  result = result/Period  #data["Momentum"] = data - data.shift(Period)  data["Momentum"] = result  PCI = [0] \* P # PCI 저장할 리스트  for ind in range(P,len(data)):  sumUpDi = 0.0  sumDownDi = 0.0  Gradient = []  Deviation = []  for ind2 in range(Period):  # ex) 5일 첫 Close + 5일 모멘텀 \* 0/4, 1/4, 2/4, 3/4, 4/4  val2 = data["Close"].iloc[ind-P] + (data["Momentum"].iloc[ind]\*(ind2)/P)  Gradient.append(np.round(val2,4))  for ind3 in range(Period):  val3 = abs(data["Close"].iloc[ind-P+ind3] - Gradient[ind3])  if data["Close"].iloc[ind-P+ind3] > Gradient[ind3]:  sumUpDi += val3  elif data["Close"].iloc[ind-P+ind3] < Gradient[ind3]:  sumDownDi += val3  else:  None  #ZeroDivisionError 방지코자 if문 사용함  if sumUpDi != 0:  a = sumUpDi / (sumUpDi+sumDownDi) \* 100  PCI.append(np.round(a,2))  else:  PCI.append(0)  return pd.Series(PCI, index=data.index) |

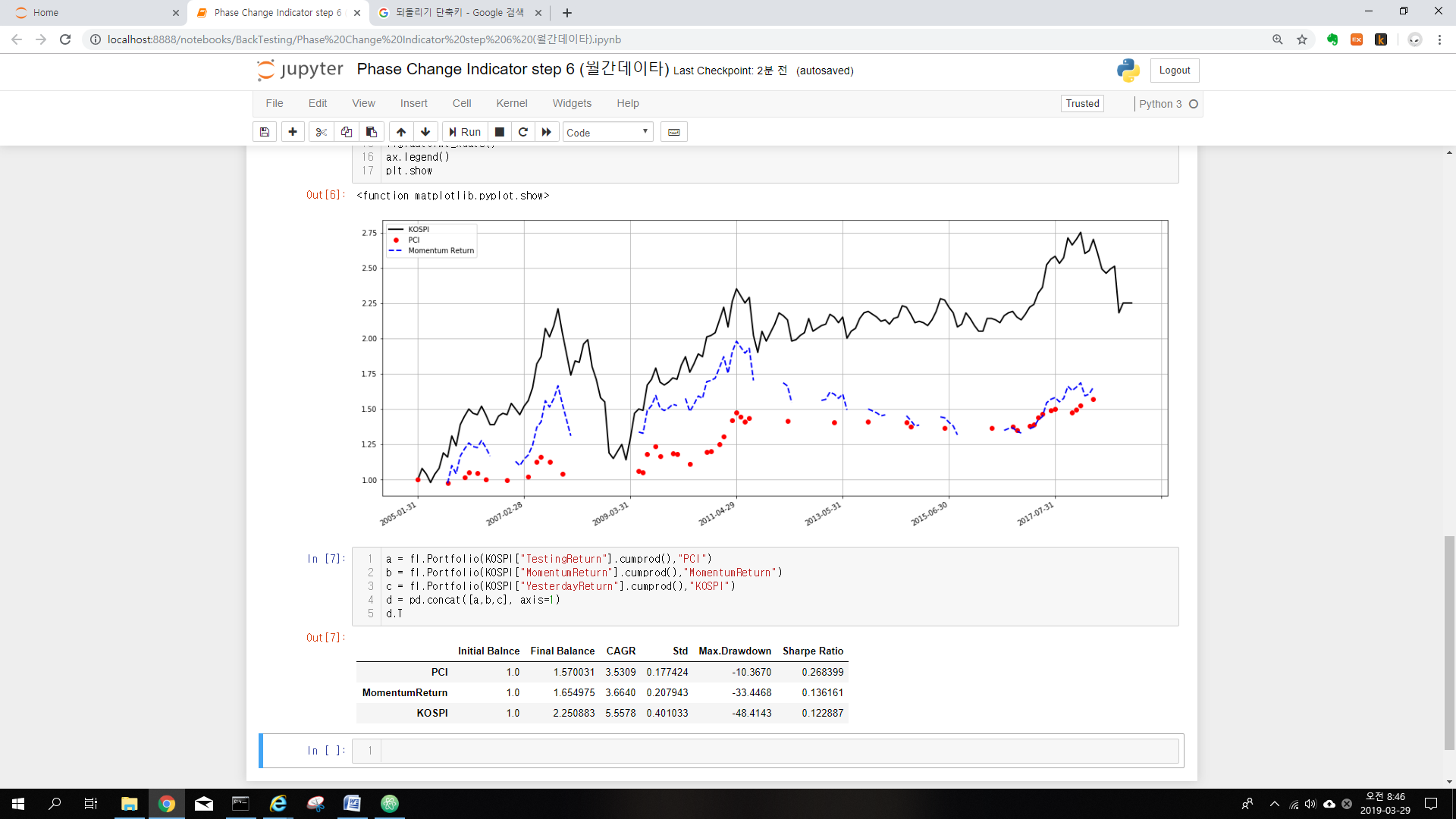
**[[ 월간 데이터 백테스팅 – Original version 수식]]**



KODEX ETF가 출시된 2010년부터 하면, MDD가 엄청떨어진다.



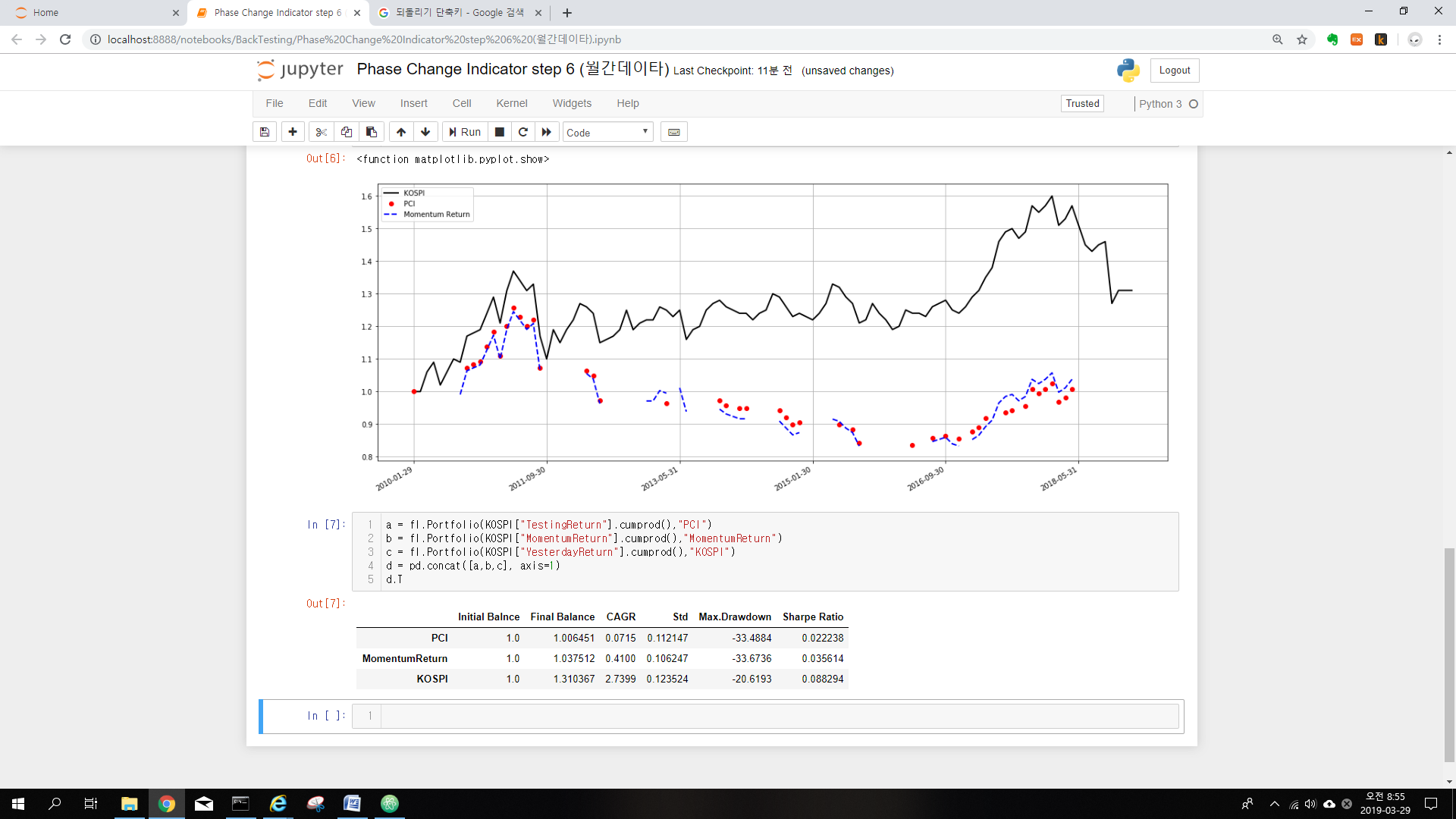
2005년부터 사용하면



|  |
| --- |
| # Momentum > 0 이고, PCI < 20 일때 매수  # Momentum < 0 이고, PCI > 80 일때 매도  entry = []  for i in range(len(KOSPI)):  if KOSPI["Momentum"][i] > 0 and KOSPI["PCI"][i] < 30: # 하락장에서 상승으로 전환, 횡보에서 상승으로 전환  #entry.append("Buy")  entry.append(1)  elif KOSPI["Momentum"][i] < 0 and KOSPI["PCI"][i] > 70: # 상승장에서 하락으로 전환, 횡보에서 하락으로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] < 0 and KOSPI["PCI"][i] < 30: # 하락장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] > 0 and KOSPI["PCI"][i] > 70: # 상승장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  else:  entry.append(np.NaN) # 20과 80사이는 상태를 정의할 수 없다. 아래 코드에서 이전 상태를 유지 한다.  KOSPI["Entry"] = entry  KOSPI["Entry"][0] = 0 # 첫번째 값을 "0"으로 세팅하여 현재 매도 상태임을 명시  #KOSPI["Entry"] = KOSPI["Entry"].fillna(method="ffill") # NaN은 횡보임으로 앞의 값으로 채움 |

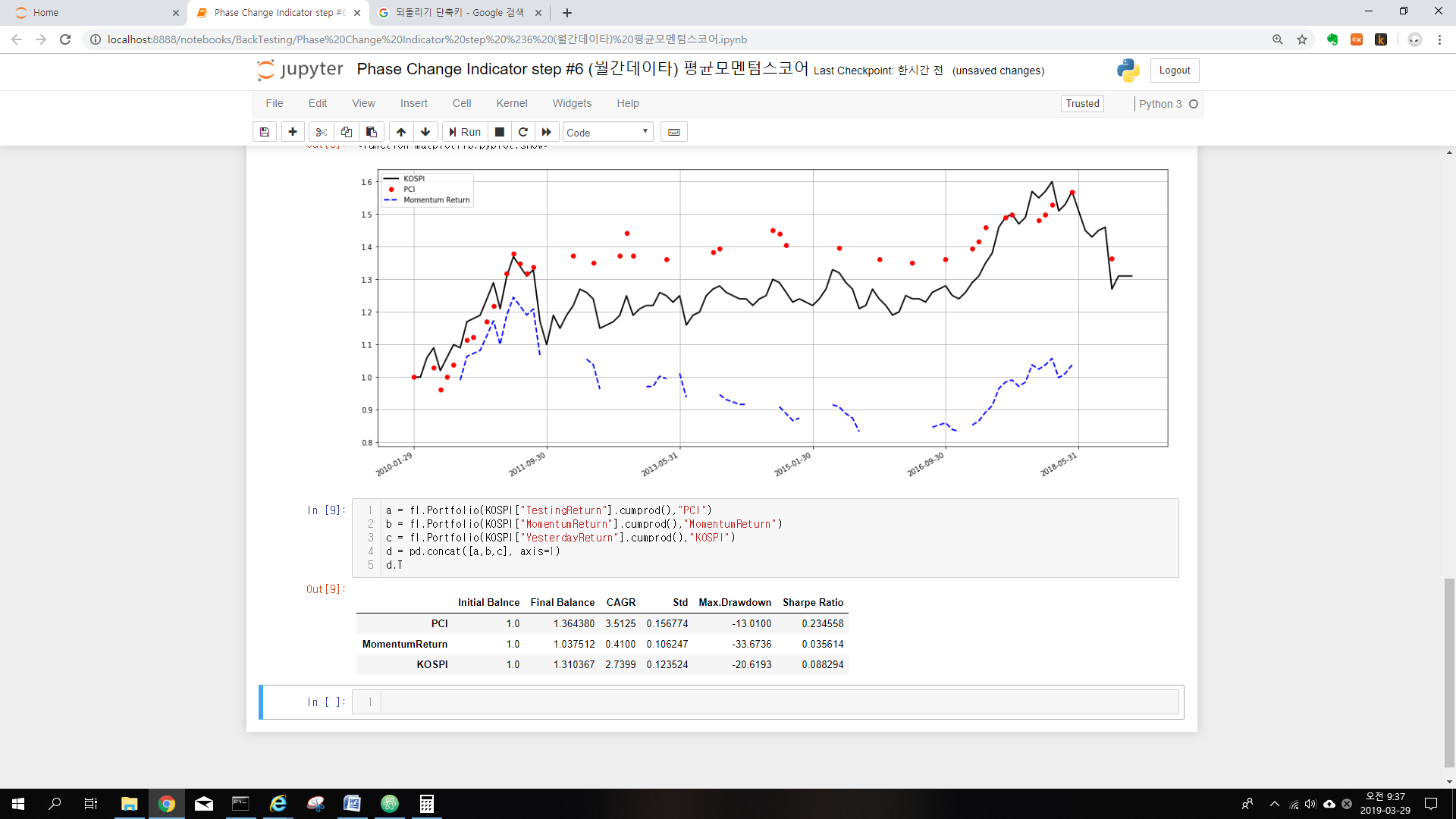
|  |
| --- |
| # PCI 구하기  def PhaseChangeIndicator(data,Period=20):  P = Period - 1  '''  result = 0  for i in range(Period+1):  result += np.where(data > data.shift(i),1,0)  result = result/Period  data["Momentum"] = result  '''    data["Momentum"] = data - data.shift(Period)  PCI = [0] \* P # PCI 저장할 리스트  for ind in range(P,len(data)):  sumUpDi = 0.0  sumDownDi = 0.0  Gradient = []  Deviation = []  for ind2 in range(Period):  # ex) 5일 첫 Close + 5일 모멘텀 \* 0/4, 1/4, 2/4, 3/4, 4/4  val2 = data["Close"].iloc[ind-P] + (data["Momentum"].iloc[ind]\*(ind2)/P)  Gradient.append(np.round(val2,4))  for ind3 in range(Period):  val3 = abs(data["Close"].iloc[ind-P+ind3] - Gradient[ind3])  if data["Close"].iloc[ind-P+ind3] > Gradient[ind3]:  sumUpDi += val3  elif data["Close"].iloc[ind-P+ind3] < Gradient[ind3]:  sumDownDi += val3  else:  None  #ZeroDivisionError 방지코자 if문 사용함  if sumUpDi != 0:  a = sumUpDi / (sumUpDi+sumDownDi) \* 100  PCI.append(np.round(a,2))  else:  PCI.append(0)  return pd.Series(PCI, index=data.index) |

횡보일 때 진인하는 조건을 추가하면, 단순모멘텀과 비슷하다

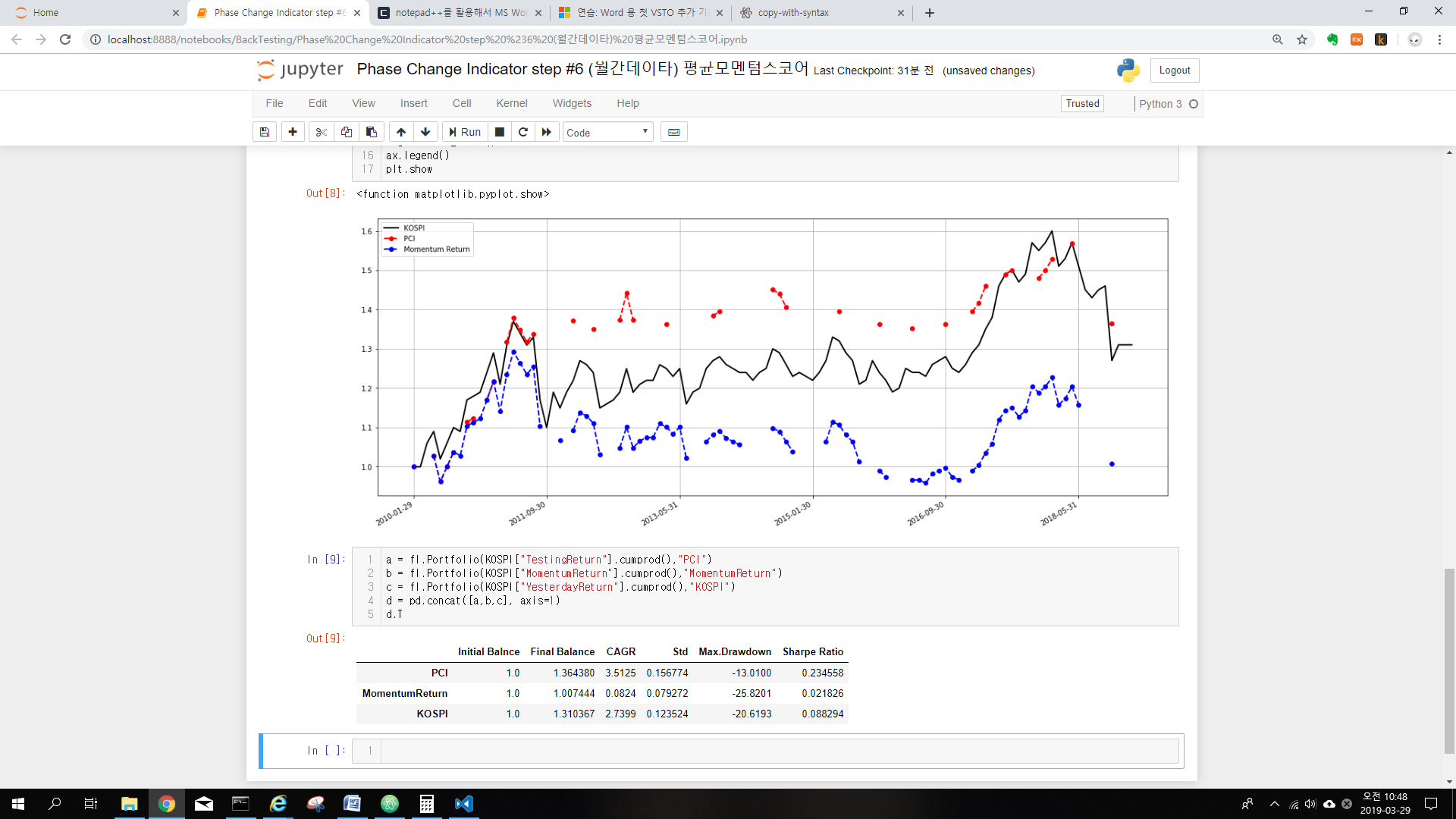


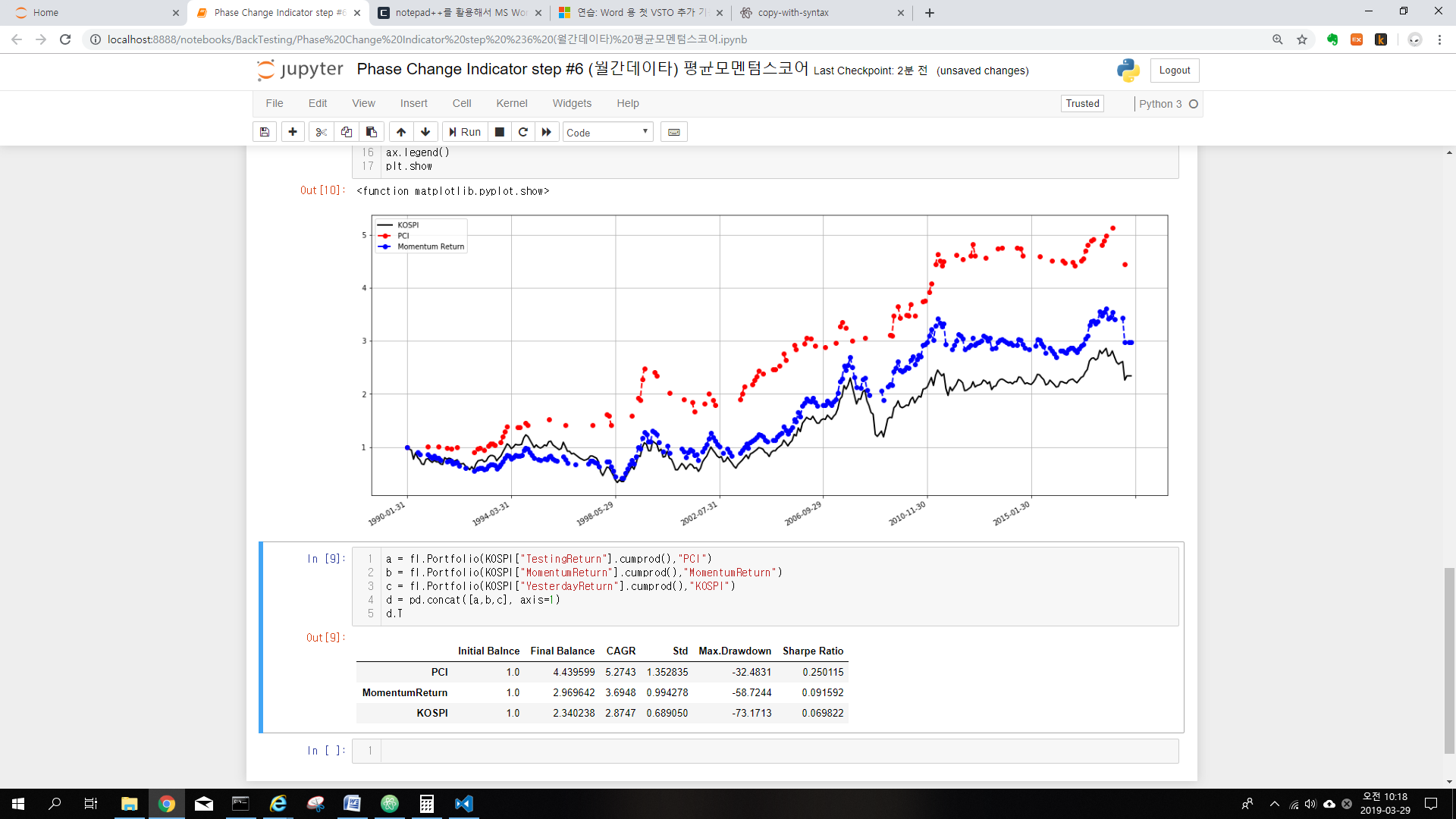
|  |
| --- |
| # Momentum > 0 이고, PCI < 20 일때 매수  # Momentum < 0 이고, PCI > 80 일때 매도  entry = []  for i in range(len(KOSPI)):  if KOSPI["Momentum"][i] > 0 and KOSPI["PCI"][i] < 30: # 하락장에서 상승으로 전환, 횡보에서 상승으로 전환  #entry.append("Buy")  entry.append(1)  elif KOSPI["Momentum"][i] < 0 and KOSPI["PCI"][i] > 70: # 상승장에서 하락으로 전환, 횡보에서 하락으로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] < 0 and KOSPI["PCI"][i] < 30: # 하락장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] > 0 and KOSPI["PCI"][i] > 70: # 상승장에서 횡보로 전환  #entry.append("Sell")  entry.append(1)  else:  entry.append(np.NaN) # 20과 80사이는 상태를 정의할 수 없다. 아래 코드에서 이전 상태를 유지 한다.  KOSPI["Entry"] = entry  KOSPI["Entry"][0] = 0 # 첫번째 값을 "0"으로 세팅하여 현재 매도 상태임을 명시  #KOSPI["Entry"] = KOSPI["Entry"].fillna(method="ffill") # NaN은 횡보임으로 앞의 값으로 채움 |

**[평균모멘텀 스코어를 사용한 전략 성과]**



단순평균모멘텀과 비교하기 - 6개월





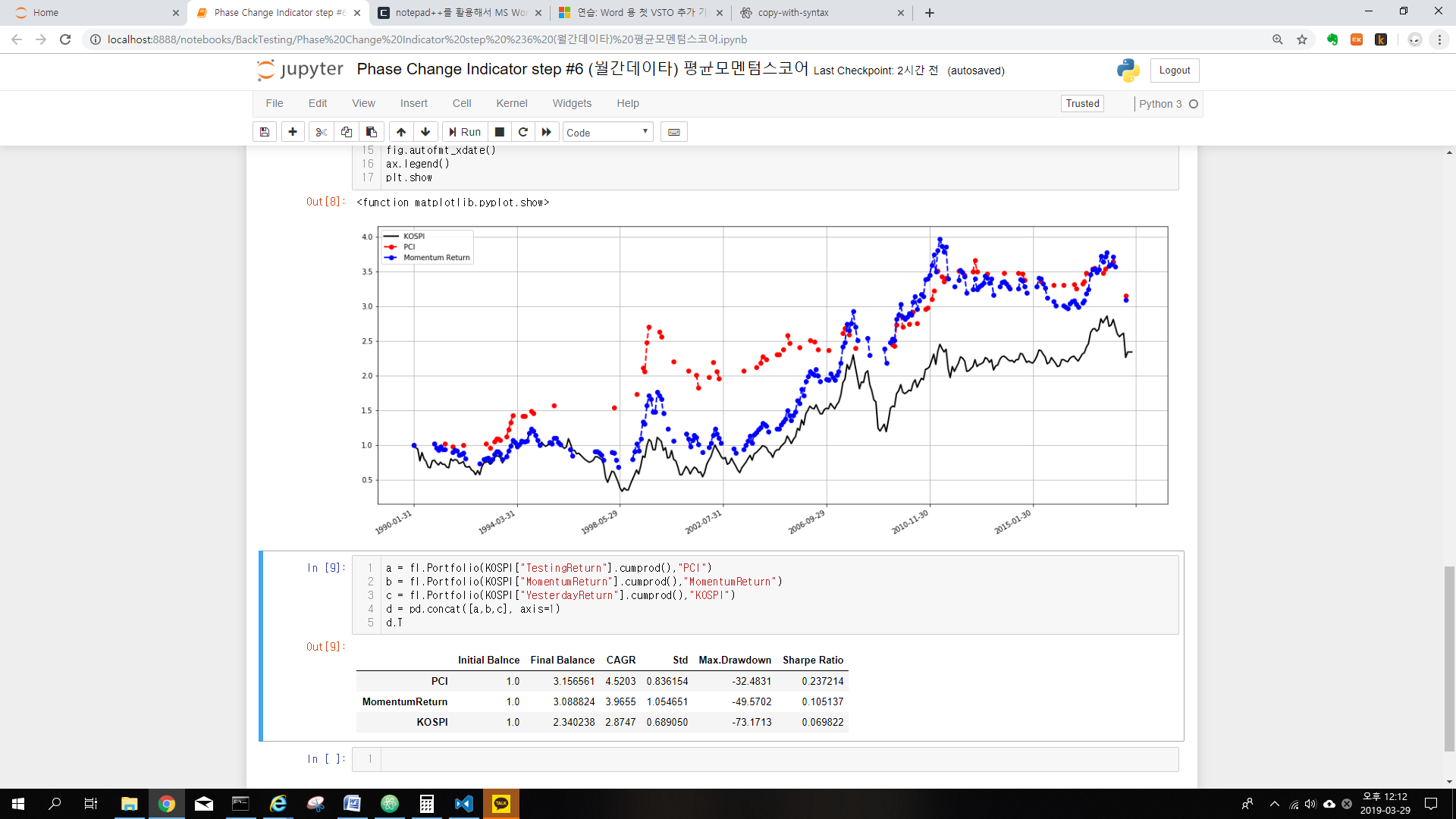
**Very Good**

|  |
| --- |
| # PCI 구하기  def PhaseChangeIndicator(data,Period=20):  P = Period - 1  #Momentum = AverageMomentumScore(data,Period)  Momentum = data - data.shift(Period)  PCI = [0] \* P # PCI 저장할 리스트  for ind in range(P,len(data)):  sumUpDi = 0.0  sumDownDi = 0.0  Gradient = []  Deviation = []  for ind2 in range(Period):  # ex) 5일 첫 Close + 5일 모멘텀 \* 0/4, 1/4, 2/4, 3/4, 4/4  val2 = data.iloc[ind-P] + (Momentum.iloc[ind]\*(ind2)/P)  Gradient.append(np.round(val2,4))  for ind3 in range(Period):  val3 = abs(data.iloc[ind-P+ind3] - Gradient[ind3])  if data.iloc[ind-P+ind3] > Gradient[ind3]:  sumUpDi += val3  elif data.iloc[ind-P+ind3] < Gradient[ind3]:  sumDownDi += val3  else:  None  #ZeroDivisionError 방지코자 if문 사용함  if sumUpDi != 0:  a = sumUpDi / (sumUpDi+sumDownDi) \* 100  PCI.append(np.round(a,2))  else:  PCI.append(0)  return pd.Series(PCI, index=data.index) |

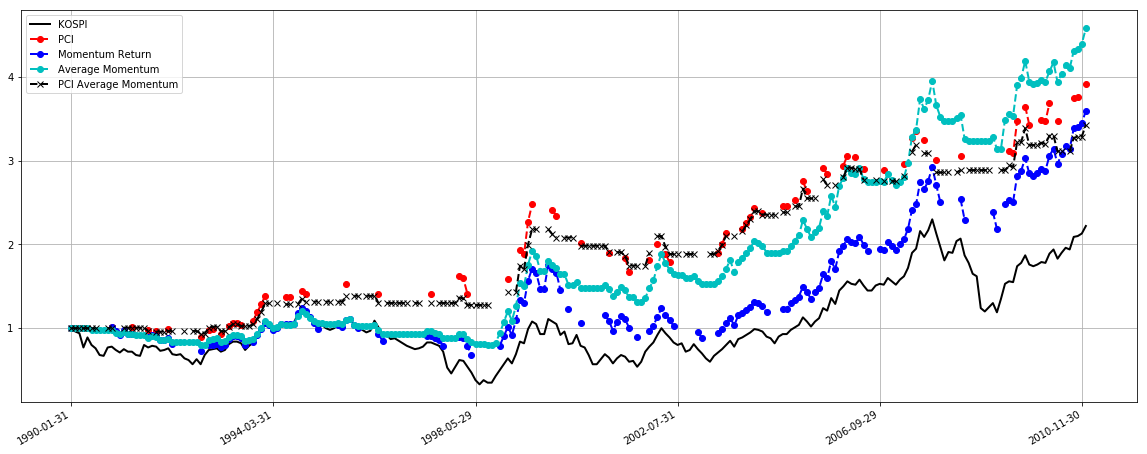
|  |
| --- |
| # Momentum > 0 이고, PCI < 20 일때 매수  # Momentum < 0 이고, PCI > 80 일때 매도  entry = []  for i in range(len(KOSPI)):  if KOSPI["Momentum"][i] > 0.2 and KOSPI["PCI"][i] < 30: # 하락장에서 상승으로 전환, 횡보에서 상승으로 전환  #entry.append("Buy")  entry.append(1)  elif KOSPI["Momentum"][i] < 0.2 and KOSPI["PCI"][i] > 70: # 상승장에서 하락으로 전환, 횡보에서 하락으로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] < 0.2 and KOSPI["PCI"][i] < 30: # 하락장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] > 0.2 and KOSPI["PCI"][i] > 70: # 상승장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  else:  entry.append(np.NaN) # 20과 80사이는 상태를 정의할 수 없다. 아래 코드에서 이전 상태를 유지 한다.  KOSPI["Entry"] = entry  KOSPI["Entry"][0] = 0 # 첫번째 값을 "0"으로 세팅하여 현재 매도 상태임을 명시  #KOSPI["Entry"] = KOSPI["Entry"].fillna(method="ffill") # NaN은 횡보임으로 앞의 값으로 채움 |

0.2보다크다 조건을 넣은 것은 단순히 6개월중 하나만 모멘텀”1”이어도 1/6=0.17 값으로 진입함으로 진입 조건을 높게 설정함.

하락/상승을 20, 80으로 수정후는 (30, 70보다 결과 좋지 못함)



**[ 최종 성과 분석 ]**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Initial Balnce | Final Balance | CAGR | Std | Max.Drawdown | Sharpe Ratio |
| PCI | 1.0 | 3.920769 | 6.7224 | 0.884822 | -32.4831 | 0.290327 |
| PCI AVG | 1.0 | 3.426670 | 6.0401 | 0.771886 | -19.8399 | 0.193866 |
| AverageMomentum | 1.0 | 4.580319 | 7.5155 | 1.037604 | -33.7247 | 0.156628 |
| MomentumReturn | 1.0 | 3.594867 | 6.2823 | 0.743104 | -49.5702 | 0.140397 |
| KOSPI | 1.0 | 2.220066 | 3.8708 | 0.445444 | -73.1713 | 0.079559 |

***파이썬 코드***

|  |
| --- |
| # coding: utf-8  # # Phase Change Indicator #6 (월간데이타)  # ##### 평균모멘텀스코어로 테스  # In[1]:  import pandas as pd  import numpy as np  # Finance Library import  from Library import FinanceLibrary as fl  # In[2]:  df = pd.read\_csv("..\\Data\\KOSPI\_Daily\_1980data.csv", encoding="utf-8")  df.index = pd.to\_datetime(df["Date"])  df\_close = df["Close"]  # 월간데이터 구하기  df\_close = df\_close.resample("BM").\_upsample(None).interpolate()  # 데이타 기간  #df\_close = df\_close["1990":"2010"]  #df\_close = df\_close["2010":]  df\_close = df\_close["1988":]  df\_close = np.round(df\_close/df\_close[0],2) #re-scale  KOSPI = pd.DataFrame(df\_close, index=df\_close.index)  # In[3]:  # fl.PhaseChangeIndicator(종가,기간)  Period = 6  PCI = fl.PhaseChangeIndicator(df\_close,Period)  KOSPI["PCI"] = PCI  # PCI에 사용하는 모멘텀은 "현재 종가" - "전일 종가"  #KOSPI["Momentum"] = KOSPI["Close"] - KOSPI["Close"].shift(Period)  KOSPI["Momentum"] = fl.AverageMomentumScore(df\_close,Period)  # In[4]:  # Momentum > 0 이고, PCI < 20 일때 매수  # Momentum < 0 이고, PCI > 80 일때 매도  entry = []  entry\_condition = 0.2  for i in range(len(KOSPI)):  if KOSPI["Momentum"][i] > entry\_condition and KOSPI["PCI"][i] < 30: # 하락장에서 상승으로 전환, 횡보에서 상승으로 전환  #entry.append("Buy")  entry.append(1)  elif KOSPI["Momentum"][i] < entry\_condition and KOSPI["PCI"][i] > 70: # 상승장에서 하락으로 전환, 횡보에서 하락으로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] < entry\_condition and KOSPI["PCI"][i] < 30: # 하락장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  elif KOSPI["Momentum"][i] > entry\_condition and KOSPI["PCI"][i] > 70: # 상승장에서 횡보로 전환  #entry.append("Sell")  entry.append(0)  else:  entry.append(np.NaN) # 20과 80사이는 상태를 정의할 수 없다. 아래 코드에서 이전 상태를 유지 한다.  KOSPI["Entry"] = entry  KOSPI["Entry"][0] = 0 # 첫번째 값을 "0"으로 세팅하여 현재 매도 상태임을 명시  #KOSPI["Entry"] = KOSPI["Entry"].fillna(method="ffill") # NaN은 횡보임으로 앞의 값으로 채움  # In[5]:  KOSPI["YesterdayReturn"] = np.round(fl.YesterdayReturn(KOSPI["Close"]),4)  KOSPI["YesterdayReturn"].iloc[0] = 1  avg = fl.AverageMomentumScore(df\_close,Period)  # 진입조건 이상일때 진입한다  avg = avg.apply(lambda x : 1 if x > entry\_condition else np.NaN)  KOSPI["MomentumReturn"] = np.round(KOSPI["YesterdayReturn"] \* avg.shift(1),4)  KOSPI["MomentumReturn"].iloc[0] = 1  #cumprod 할때 "0"이 있으면 무조건 수익률이 0으로 나옴으로 "0"을 제거하는 코드 필요함.  KOSPI["MomentumReturn"] = KOSPI["MomentumReturn"].apply(lambda x: np.NaN if x == 0 else x)  # 매무 중요 진입 시그널은 어제것을 사용한다. 왜? 수익이 "어제 진입해서 오늘 확증"되는 수익률 임으로  KOSPI["TestingReturn"] = np.round(KOSPI["YesterdayReturn"] \* KOSPI["Entry"].shift(1),4)  #cumprod 할때 "0"이 있으면 무조건 수익률이 0으로 나옴으로 "0"을 제거하는 코드 필요함.  KOSPI["TestingReturn"] = KOSPI["TestingReturn"].apply(lambda x: np.NaN if x == 0 else x)  KOSPI["TestingReturn"].iloc[0] = 1  # 평균모멘텀스코어 PCI  pci\_avg\_retrun = np.round(KOSPI["YesterdayReturn"] \* KOSPI["Entry"].shift(1) \* KOSPI["Momentum"].shift(1),4)  #cumprod 할때 "0"이 있으면 무조건 수익률이 0으로 나옴으로 "0"을 제거하는 코드 필요함.  KOSPI["PCIReturn"] = pci\_avg\_retrun + (1 - KOSPI["Momentum"].shift(1) \* KOSPI["Entry"].shift(1))  KOSPI["PCIReturn"] = KOSPI["PCIReturn"].apply(lambda x: np.NaN if x == 0 else x)  KOSPI["PCIReturn"].iloc[0] = 1  # 평균모멘텀스코어 누적수익률  average\_score = fl.AverageMomentumScore(df\_close,Period)  average\_return = KOSPI["YesterdayReturn"] \* average\_score.shift(1) # 모멘텀 수익률 \* 평균스코어만큼 진입  KOSPI["AverageMomentum"] = average\_return + (1-average\_score.shift(1)) # 평균모멘텀 수익률 = 평균모멘텀 수익률 + 현금(1-average\_score)  KOSPI["AverageMomentum"] = KOSPI["AverageMomentum"].apply(lambda x: np.NaN if x == 0 else x)  KOSPI["AverageMomentum"].iloc[0] = 1  # In[6]:  get\_ipython().run\_line\_magic('matplotlib', 'inline')  import matplotlib.pyplot as plt  plt.rcParams["figure.figsize"] = (20,8)  plt.rcParams['lines.linewidth'] = 2  plt.rcParams['lines.color'] = 'r'  plt.rcParams['axes.grid'] = True  formatter = fl.MyFormatter(KOSPI.index)  fig, ax = plt.subplots()  ax.xaxis.set\_major\_formatter(formatter)  ax.plot(np.arange(len(KOSPI)), KOSPI["YesterdayReturn"].cumprod(), color="Black", label="KOSPI")  ax.plot(np.arange(len(KOSPI)), KOSPI["TestingReturn"].cumprod(), color="Red", label="PCI", marker="o", linestyle="--")  ax.plot(np.arange(len(KOSPI)), KOSPI["MomentumReturn"].cumprod(), color="Blue", linestyle="--", marker="o", label="Momentum Return")  ax.plot(np.arange(len(KOSPI)), KOSPI["AverageMomentum"].cumprod(), color="C", linestyle="--",marker="o", label="Average Momentum")  ax.plot(np.arange(len(KOSPI)), KOSPI["PCIReturn"].cumprod(), color="K", linestyle="--",marker="x", label="PCI Average Momentum")  fig.autofmt\_xdate()  ax.legend()  plt.show  # In[7]:  a = fl.Portfolio(KOSPI["TestingReturn"].cumprod(),"PCI")  b = fl.Portfolio(KOSPI["PCIReturn"].cumprod(),"PCI AVG")  c = fl.Portfolio(KOSPI["AverageMomentum"].cumprod(),"AverageMomentum")  d = fl.Portfolio(KOSPI["MomentumReturn"].cumprod(),"MomentumReturn")  e = fl.Portfolio(KOSPI["YesterdayReturn"].cumprod(),"KOSPI")  f = pd.concat([a,b,c,d,e], axis=1)  f.T  # In[8]:  get\_ipython().run\_line\_magic('matplotlib', 'inline')  import matplotlib.pyplot as plt  plt.rcParams["figure.figsize"] = (20,8)  plt.rcParams['lines.linewidth'] = 2  plt.rcParams['lines.color'] = 'r'  plt.rcParams['axes.grid'] = True  formatter = fl.MyFormatter(KOSPI.index)  fig, ax = plt.subplots()  ax.xaxis.set\_major\_formatter(formatter)  a\_dd,a\_mdd = fl.MDD(KOSPI["YesterdayReturn"].cumprod())  a = a\_dd.apply(lambda x:x if x < 0 else 0)  b\_dd,b\_mdd = fl.MDD(KOSPI["TestingReturn"].cumprod())  b = b\_dd.apply(lambda x:x if x < 0 else 0)  c\_dd,c\_mdd = fl.MDD(KOSPI["MomentumReturn"].cumprod())  c = c\_dd.apply(lambda x:x if x < 0 else 0)  d\_dd,d\_mdd = fl.MDD(KOSPI["AverageMomentum"].cumprod())  d = d\_dd.apply(lambda x:x if x < 0 else 0)  e\_dd,e\_mdd = fl.MDD(KOSPI["PCIReturn"].cumprod())  e = e\_dd.apply(lambda x:x if x < 0 else 0)  #ax.plot(np.arange(len(KOSPI)), a, color="Black", label="KOSPI")  #ax.plot(np.arange(len(KOSPI)), a\_mdd)  ax.plot(np.arange(len(KOSPI)), b, color="Red", label="PCI", marker="o", linestyle="--")  #ax.plot(np.arange(len(KOSPI)),b\_mdd)  #ax.plot(np.arange(len(KOSPI)), c, color="Blue", linestyle="--", marker="o", label="Momentum Return")  #ax.plot(np.arange(len(KOSPI)), d, color="C", linestyle="--",marker="o", label="Average Momentum")  ax.plot(np.arange(len(KOSPI)), e, color="K", linestyle="--",marker="x", label="PCI Average Momentum")  #ax.plot(np.arange(len(KOSPI)),e\_mdd)  fig.autofmt\_xdate()  ax.legend()  plt.show  # In[9]:  KOSPI["Close"].plot() |