Trabajo Práctico ITBA - Python

October 13, 2022

Gerardo Damian Gimeno DNI: 24.296.945 gerardogimeno0772@gmail.com

```
ef validar fecha(fecha):
 name__ == '__main__':
                                                                                   def solicitar datos opcion1():
                                                                                                                                                                        format = "%Y-%m-%d" #valida en el
menu_principal()
                                                                                                                                                                        if len(fecha) != 10:
                                                                                    /ticker = input('Ingresar ticker a pedir:')
                                                                                                                                                                         raise ValueError("Formato de fecha erróneo. Ingresar AAAA-MM-DD")
                                                                                     desde = input('Ingrese fecha de inicio en este formato año-mes-día: ')
                                                                                                                                                                        if fecha[4] != '-' or fecha[7] != '-':
def menu_principal():
                                                                                     hasta = input('Ingrese fecha de fin en este formato año-mes-día: ')
                                                                                                                                                                         raise ValueError("Formato de fecha erróneo. Ingresar AAAA-MM-DD")
  base_de_datos="tp_polygon.db"
                                                                                     validar fecha(hasta)
  menu="Seleccione una opción\n1. Actualización de datos\n2. Visualización de datos"
                                                                                                                                                                         res = bool(datetime.datetime.strptime(fecha, format))
                                                                                     return ticker, desde, hasta
  opcion=input(menu)
                                                                                                                                                                        except ValueError:
                                                                                                                                                                         raise ValueError("Fecha errónea")
                                                                                                                                                                        return True
  if opcion=="1":
                                         def opcion1(base_de_datos):
    try:
      opcion1(base_de_datos)
                                                                                                                def llamar_api_polygon(ticker,timespan,fecha_desde,fecha_hasta):
                                           ticker,desde,hasta = soficitar_datos_opcion1()
    except Exception as e:
                                           print("Pidiendo datos...")
      print(e)
                                                                                                                   key = 'eWjcDZNVWFJ5RC_xwFvz8qbuS4SR1huN'
  elif opcion=="2":
                                           time.sleep(3)
                                                                                                                   polygon stocks url = f"https://api.polygon.io/v2/aggs/ticker/{ticker}/range/1/{timespan}/{fecha desde}/{fecha hasta}?unadjusted=true&apiKey={key}"
                                           timespan="day"
    try:
                                                                                                                   session = requests.Session()
                                           resultados_api=llamar_api_polygon(ticker,timespan,desde,hasta)
      popcion2(base_de_datos)
                                                                                                                   r = session.get(polygon_stocks_url)
                                           guardar_informacion(base de datos,ticker,desde,hasta,resultados api)
    except Exception as e:
                                                                                                                   data = r.json()
      print(e)
                                           print("Operación exitosa")
                                                                                                                   if data['resultsCount'] == 0:
    print("Opción incorrecta")
                                                                                                                     raise ValueError("No existen datos a listar. Verifique los datos ingresados")
                                                                                                                   return data['results']
                                                                             def guardar_informacion(base_de_datos,ticker, fecha_desde, fecha_hasta,info_polygon):
                                                                                con = sqlite3.connect(base de datos)
def opcion2(base_de_datos):
                                                                                cur = con.cursor()
                                                                                sql create=""CREATE TABLE IF NOT EXISTS stocks
  opcion=mostrar_opcion2_sub_opciones()
                                                                                       (nro_consulta integer primary key autoincrement, ticker text, fecha date, v real, vw real, o real, c real, h real, l real, t integer, n integer)
  if opcion=="1":
     mostrar_resumen(base_de_datos)
                                                                                cur.execute(sql_create)
   else:
                                                                                for stock info in info polygon:
                                                                                  sql insert="INSERT INTO stocks(ticker, fecha, v,vw,o, c,h,l,t,n ) VALUES (?,?,?,?,?,?,?,?)"
     graficar_ticker(base_de_datos)
                                                                                  fecha_insert=convertir_timestamp_epoc_a_fecha(stock_info['t'])
                                                                                  valores=(ticker, fecha_insert,stock_info['v'],stock_info['vw'],stock_info['o'],stock_info['c'],stock_info['h'],stock_info['l'],stock_info['t'],stock_info['n'])
                                                                                  cur.execute(sql insert,valores)
                                                                                con.commit()
                                                                                                                                                                                 def convertir_timestamp_epoc_a_fecha(t):
                                                                                con.close()
                                                                                                                                                                                    t segs = t/1000
       def mostrar_resumen(base_de_datos):
                                                                                                                                                                                   fecha_act_aux = datetime.datetime.fromtimestamp(t_segs)
                                                                                                                def graficar_ticker(base_de_datos):
                                                                                                                                                                                   fecha_convert=datetime.datetime.strftime(fecha_act_aux,"%Y-%m-%d")
            con = sqlite3.connect(base_de_datos)
                                                                                                                                                                                   return fecha_convert
                                                                                                                   print("graficar ticker")
            cur = con.cursor()
                                                                                                                   texto_opcion_graficar_ticker="Ingrese ticker a graficar"
            sql_select_group_by="SELECT ticker,MIN(fecha),MAX(fecha) FROM stocks GROUP BY ticker"
                                                                                                                   ticker elegido=input(texto_opcion_graficar_ticker)
            cur.execute(sql_select_group_by)
                                                                                                                   con = sqlite3.connect(base_de_datos)
                                                                                                                   cur = con.cursor()
                                                                                                                   sql select ticker="SELECT fecha,c FROM stocks WHERE ticker=:id ticker ORDER BY fecha ASC"
            for (ticker,fecha_min,fecha_max) in cur.fetchall():
                                                                                                                   cur.execute(sql_select_ticker,{"id_ticker":ticker_elegido})
              linea=ticker+" - "+fecha min+" <-> "+fecha max
                                                                                                                   datos=cur.fetchall()
               print(linea)
                                                                                                                   fechas=[x[0] \text{ for } x \text{ in datos }]
                                                                                                                   valores_cierre=[x[1] for x in datos]
            con.close()
                                                                                                                   plt.plot(fechas, valores_cierre)
                                                                                                                   plt.ylabel('Valor al cierre') # Leyenda en el eje y
                                                                                                                   plt.xticks(rotation=45)
                                                                                                                   plt.xlabel('Fecha') # Leyenda en el eje x
def mostrar_opcion2_sub_opciones():
                                                                                                                   fecha min=fechas[0]
   menu_opcion2="Visualización de datos - Seleccione una opción\n1. Resumén\n2. Grafico de Ticker"
                                                                                                                   fecha max=fechas[len(fechas)-1]
   sub opcion2=input(menu opcion2)
   if sub_opcion2!="1" and sub_opcion2!="2":
                                                                                                                   titulo="Evolución ticker "+ticker_elegido+" "+fecha_min+" - "+fecha_max
     raise ValueError("Número de opción incorrecta")
                                                                                                                   plt.title(titulo) ## Título de Gráfica
                                                                                                                   nombre_grafico=ticker_elegido+"-"+fecha_min+"-"+fecha_max+".jpg"
   return sub_opcion2
                                                                                                                   plt.savefig(nombre grafico)
                                                                                                                   con.close()
```

```
def solicitar datos opcion1():
    name == ' main ':
                                                                                                                                                                                     def validar fecha(fecha):
                                                                                                                   ticker = input('Ingresar ticker a pedir:')
  menu principal()
                                                                                                                                                                                        format = "%Y-%m-%d" #valida en el
                                                                                                                   desde = input('Ingrese fecha de inicio en este formato año-mes-día: ')
                                                                                                                                                                                        if len(fecha) != 10:
                                                                                                                   hasta = input('Ingrese fecha de fin en este formato año-mes-día: ')
 def menu principal():
                                                                                                                                                                                          raise ValueError("Formato de fecha erróneo. Ingresar AAAA-MM-DD")
                                                                                                                   validar fecha(hasta)
   base de datos="tp polygon.db"
                                                                                                                   return ticker, desde, hasta
                                                                                                                                                                                        if fecha[4] != '-' or fecha[7] != '-':
   menu="Seleccione una opción\n1. Actualización de datos\n2. Visualización de datos"
                                                                                                                                                                                          raise ValueError("Formato de fecha erróneo. Ingresar AAAA-MM-DD")
   opcion=input(menu)
                                                                                                                                                                                          res = bool(datetime.datetime.strptime(fecha, format))
   if opcion=="1":
                                                                                                                                                                                        except ValueError:
      try:
                                               def opcion1(base de datos):
                                                                                                                                                                                          raise ValueError("Fecha errónea")
        opcion1(base de datos)
                                                                                                                                                                                       return True
                                                  ticker, desde, hasta = solicitar datos opcion1()
      except Exception as e:
                                                  print("Pidiendo datos...")
        print(e)
                                                  time.sleep(3)
   elif opcion=="2":
                                                  timespan="day"
      try:
                                                  resultados api=llamar api polygon(ticker,timespan,desde,hasta)
                                                                                                                        def llamar api polygon(ticker, timespan, fecha desde, fecha hasta):
        opcion2(base de datos)
                                                  guardar informacion(base de datos, ticker, desde, hasta, resultados api)
      except Exception as e:
                                                  print("Operación exitosa")
                                                                                                                          key = 'eWjcDZNVWFJ5RC xwFvz8qbuS4SR1huN'
        print(e)
                                                                                                                          polygon stocks url = f"https://api.polygon.io/v2/aggs/ticker/{ticker}/range/1/{timespan}/{fecha desde}/{fecha hasta}?unadjusted=true&apiKey={key}"
   else:
                                                                                                                          session = requests.Session()
      print("Opción incorrecta")
                                                                                                                          r = session.get(polygon stocks url)
                                                                                                                          data = r.json()
                                                                                                                          if data['resultsCount'] == 0:
def guardar_informacion(base_de_datos,ticker, fecha_desde, fecha_hasta,info_polygon):
                                                                                                                            raise ValueError("No existen datos a listar. Verifique los datos ingresados")
  con = sqlite3.connect(base de datos)
  cur = con.cursor()
 sgl create=""CREATE TABLE IF NOT EXISTS stocks
                                                                                                                          return data['results']
        (nro_consulta integer primary key autoincrement, ticker text, fecha date, v real, v real, o real, c real, h real, l real, t integer, n int
 cur.execute(sql create)
  for stock info in info polygon:
   sql insert="INSERT INTO stocks(ticker, fecha, v,vw,o, c,h,l,t,n ) VALUES (?,?,?,?,?,?,?,?,?)"
                                                                                                                                                                DIAGRAMA FUNCIONAL DE CONCATENACIÓN DE FUNCIONES
   fecha insert=convertir timestamp epoc a fecha(stock info['t'])
   valores=(ticker, fecha insert,stock info['v'],stock info['vw'],stock info['o'],stock info['c'],stock info['h'],stock info['l'],stock info['t'],stock info['n'])
   cur.execute(sql insert,valores)
```

con.commit() con.close()

Como se solicitan los datos a la API y se almacenan en SQLITE3

DIAGRAMA FUNCIONAL DE CONCATENACIÓN DE FUNCIONES Como se solicitan los datos a la API continuación.

```
name == ' main ':
menu principal()
def menu principal():
  base de datos="tp polygon.db"
  menu="Seleccione una opción\n1. Actualización de datos\n2. Visualización de datos"
  opcion=input(menu)
                                rs\ggimeno3\OneDrive\Documentos personales\Python\ITBA\Ejercicio final>python Integrador9.py
  if opcion=="1":
                             Actualización de datos
                             Visualización de datos
    try:
      opcion1(base de datos)
    except Exception as e:
                                         def opcion1(base de datos):
      print(e)
  elif opcion=="2":
                                           ticker, desde, hasta = solicitar datos opcion1()
                                           print("Pidiendo datos...")
    try:
                                           time.sleep(3)
      opcion2(base de datos)
                                           timespan="day"
    except Exception as e:
                                           resultados api=llamar api polygon(ticker,timespan,desde,hasta)
      print(e)
                                           guardar informacion(base de datos, ticker, desde, hasta, resultados api)
  else:
                                           print("Operación exitosa")
    print("Opción incorrecta")
                                                     def solicitar datos opcion1():
                                                       ticker = input('Ingresar ticker a pedir:')
                                                        desde = input('Ingrese fecha de inicio en este formato año-mes-día: ')
                                                        hasta = input('Ingrese fecha de fin en este formato año-mes-día: ')
                                                        return ticker, desde, hasta
```

DIAGRAMA FUNCIONAL DE CONCATENACIÓN DE FUNCIONES Como se realiza el get a la API.

```
name == ' main ':
   menu principal()
  def menu_principal():
     base de datos="tp polygon.db"
     menu="Seleccione una opción\n1. Actualización de datos\n2. Visualización de datos"
     opcion=input(menu)
     if opcion=="1":
       try:
          opcion1(base de datos)
       except Exception as e:
                                               def opcion1(base de datos):
          print(e)
     elif opcion=="2":
                                                 ticker,desde,hasta = solicitar_datos_opcion1()
                                                 print("Pidiendo datos...")
       try:
                                                 time.sleep(3)
          opcion2(base_de_datos)
                                                                                                   ngrese fecha de inicio en este formato año-mes-día: 2022-05-11
                                                 timespan="day"
       except Exception as e:
                                                 resultados api=llamar_api_polygon(ticker,timespan,desde,hasta)-
          print(e)
                                                 guardar informacion(base de datos, ticker, desde, hasta, resultados api)
     else:
                                                 print("Operación exitosa")
       print("Opción incorrecta")
def llamar_api_polygon(ticker, timespan, fecha_desde, fecha_hasta):
 key = 'eWjcDZNVWFJ5RC xwFvz8gbuS4SR1huN'
 polygon_stocks_url = f"https://api.polygon.io/v2/aggs/ticker/{ticker}/range/1/{timespan}/{fecha_desde}/{fecha_hasta}?unadjusted=true&apiKey={key}"
 session = requests.Session()
 r = session.get(polygon stocks url)
  data = r.ison()
 if data['resultsCount'] == 0:
   raise ValueError("No existen datos a listar. Verifique los datos ingresados")
 return data['results']
```

DIAGRAMA FUNCIONAL DE CONCATENACIÓN DE FUNCIONES Dar formato de fecha y guardado de info en SQLITE3

```
name == ' main ':
 menu principal()
def menu principal():
   base de datos="tp_polygon.db"
   menu="Seleccione una opción\n1. Actualización de datos\n2. Visualización de datos"
   opcion=input(menu)
   if opcion=="1":
     try:
        opcion1(base de datos)
     except Exception as e:
                                              def opcion1(base de datos):
        print(e)
   elif opcion=="2":
                                                ticker, desde, hasta = solicitar datos opcion1()
                                                print("Pidiendo datos...")
     try:
                                                time.sleep(3)
        opcion2(base de datos)
                                                timespan="day"
     except Exception as e:
                                                resultados api=llamar_api_polygon(ticker,timespan,desde,hasta)
        print(e)
                                                guardar informacion(base de datos,ticker,desde,hasta,resultados api)
   else:
                                                print("Operación exitosa") <
     print("Opción incorrecta")
                                                                                             def convertir_timestamp_epoc_a_fecha(t):
                                                                                               t \text{ segs} = t/1000
                                                                                               fecha act aux = datetime.datetime.fromtimestamp(t segs)
def guardar informacion(base de datos, ticker, fecha desde, fecha hasta, info polygon):
                                                                                               fecha convert=datetime.datetime.strftime(fecha act aux,"%Y-%m-%d")
  con = sqlite3.connect(base de datos)
                                                                                               return fecha convert
 cur = con.cursor()
 sql create=""CREATE TABLE IF NOT EXISTS stocks
         (nro_consulta integer primary key autoincrement, ticker text, fecha date, v real, vw real, o real, c real, h real, l real, t integer, n integer)
 cur.execute(sql create)
 for stock info in info polygon:
    sql insert="INSERT INTO stocks(ticker, fecha, v,vw,o, c,h,l,t,n ) VALUES (?,?,?,?,?,?,?,?,?)"
    fecha insert=convertir timestamp epoc a fecha(stock info['t'])
    valores=(ticker, fecha insert, stock info['v'], stock info['vw'], stock info['o'], stock info['c'], stock info['h'], stock info['l'], stock info['l'], stock info['n'])
    cur.execute(sql insert, valores)
```

DIAGRAMA FUNCIONAL DE CONCATENACIÓN DE FUNCIONES Menues de visualización de datos y Grafica.

```
_name___ == '__ main ':
 menu principal()
def menu_principal():
  base de datos="tp polygon.db"
  menu="Seleccione una opción\n1. Actualización de datos\n2. Visualización de datos"
  opcion=input(menu)
  if opcion=="1":
                                                               def opcion2(base de datos):
    try:
       opcion1(base de datos)
                                                                 opcion=mostrar_opcion2_sub_opciones() ◆
     except Exception as e:
                                                                 if opcion=="1":
       print(e)
                                                                   mostrar_resumen(base de datos)
  elif opcion=="2":
                                                                 else:
     try:
                                                                   graficar_ticker(base_de_datos)
       opcion2(base de datos)
     except Exception as e:
       print(e)
                                               sers\ggimeno3\OneDrive\Documentos personales\Python\ITBA\Ejercicio final>python tp_polygon.py
  else:
     print("Opción incorrecta")
                                              Grafico de Ticker1
                                             TSLA - 2022-05-02 <-> 2022-07-01
def mostrar opcion2 sub opciones():
  menu opcion2="Visualización de datos - Seleccione una opción\n1. Resumén\n2. Grafico de Ticker"
  sub opcion2=input(menu opcion2)
  if sub opcion2!="1" and sub opcion2!="2":
    raise ValueError("Número de opción incorrecta")
  return sub opcion2
```

```
_name__ == ' _ main _ ':
menu principal()
def menu principal():
  base_de_datos="tp_polygon.db"
  menu="Seleccione una opción\n1. Actualización de datos\n2. Visualización de datos"
  opcion=input(menu)
  if opcion=="1":
                                                                   def opcion2(base_de_datos):
    try:
       opcion1(base de datos)
                                                                      opcion=mostrar_opcion2_sub_opciones()
    except Exception as e:
                                                                      if opcion=="1":
       print(e)
                                                                        mostrar_resumen(base de datos) -
  elif opcion=="2":
                                                                      else:
    try:
                                                                        graficar ticker(base de datos)
       opcion2(base de datos)
    except Exception as e:
       print(e)
                                                       \Users\ggimeno3\OneDrive\Documentos personales\Python\ITBA\Ejercicio final>python tp_polygon.py
  else:
                                                       Actualización de datos
    print("Opción incorrecta")
                                                        Visualización de datos
                                                       sualización de datos - Seleccione una opción
                                                      APL - 2021-12-01 <-> 2022-09-19
                                                      TSLA - 2022-05-02 <-> 2022-07-01
   def mostrar_resumen(base_de_datos):
       con = sqlite3.connect(base de datos)
        cur = con.cursor()
       sql select group by="SELECT ticker, MIN(fecha), MAX(fecha) FROM stocks GROUP BY ticker"
        cur.execute(sql select group by)
        for (ticker, fecha min, fecha max) in cur. fetchall():
          linea=ticker+" - "+fecha min+" <-> "+fecha max
          print(linea)
        con.close()
```

```
name == ' main ':
menu_principal()
                                                                                           def opcion2(base_de_datos):
def menu principal():
 base de datos="tp polygon.db"
 menu="Seleccione una opción\n1. Actualización de datos\n2. Visualización de datos"
                                                                                              opcion=mostrar_opcion2_sub_opciones()
 opcion=input(menu)
                                                                                              if opcion=="1":
                                                                                                 mostrar resumen(base de datos)
  if opcion=="1":
                                                                                              else:
    try:
      opcion1(base de datos)
                                                                                                 graficar_ticker(base de datos) ←
    except Exception as e:
      print(e)
 elif opcion=="2":
                                  def graficar ticker(base de datos):
    try:
      opcion2(base de datos)
                                    print("graficar ticker")
    except Exception as e:
                                    texto opcion graficar ticker = "Ingrese el acrónimo del ticker a graficar. Ejemplo AAPL\nNombre del Ticker: "
      print(e)
                                    ticker elegido = input(texto opcion graficar ticker)
  else:
                                    con = sqlite3.connect(dase de datos)
    print("Opción incorrecta")
                                    cur = con.cursor()
                                    sql select ticker = "SELECT fecha,c FROM stocks WHERE ticker=:id ticker ORDER BY fecha ASC"
                                    cur.execute(sql select ticker, {"id ticker": ticker elegido})
                                    datos = cur.fetchall()
                                    fechas = [x[0]] for x in datos
                                    valores cierre = [x[1] for x in datos]
                                    plt.figure(figsize=(15, 10))
                                    plt.plot(fechas, valores cierre)
                                    plt.ylabel('Valor al cierre') # Leyenda en el eje y
                                    plt.xticks(rotation=45)
                                    plt.xlabel('Fecha') # Levenda en el eje x
                                    fecha min = fechas[0]
                                    fecha max = fechas[len(fechas) - 1]
                                    titulo = "Evolución ticker" + ticker_elegido + " desde " + fecha_min + " al " + fecha_max
                                    plt.title(titulo) ## Título de Gráfica
                                    # plt.show() #Descomentar para mostrar en pantalla
                                    nombre_grafico = ticker_elegido + "-" + fecha_min + "-" + fecha_max + ".jpg"
                                    plt.savefig(nombre_grafico)
                                    con.close()
```