Trabajo Práctico ITBA - Python

October 13, 2022

Gerardo Damian Gimeno
DNI: 24.296.945
gerardogimeno0772@gmail.com

```python
if __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:
            print(e)
    elif opcion=="2":
        try:
            opcion2(base_de_datos)
        except Exception as e:
            print(e)
    else:
        print("Opción incorrecta")
```

```python
def opcion1(base_de_datos):

    ticker,desde,hasta = solicitar_datos_opcion1()
    print("Pidiendo datos...")
    time.sleep(3)
    timespan="day"
    resultados_api=llamar_api_polygon(ticker,timespan,desde,hasta)
    guardar_informacion(base_de_datos,ticker,desde,hasta,resultados_api)
    print("Operación exitosa")
```

```python
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: ')
    validar_fecha(hasta)
    return ticker,desde,hasta
```

```python
def validar_fecha(fecha):
    format = "%Y-%m-%d" #valida en el
    if len(fecha) != 10:
        raise ValueError("Formato de fecha erróneo. Ingresar AAAA-MM-DD")
    if fecha[4] != '-' or fecha[7] != '-':
        raise ValueError("Formato de fecha erróneo. Ingresar AAAA-MM-DD")
    try:
        res = bool(datetime.datetime.strptime(fecha, format))
    except ValueError:
        raise ValueError("Fecha errónea")
    return True
```

```python
def llamar_api_polygon(ticker,timespan,fecha_desde,fecha_hasta):

    key = 'eWjcDZNVWFJ5RC_xwFvz8qbuS4SR1huN'
    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.json()

    if data['resultsCount'] == 0:
        raise ValueError("No existen datos a listar. Verifique los datos ingresados")
    return data['results']
```

```python
def opcion2(base_de_datos):

    opcion=mostrar_opcion2_sub_opciones()
    if opcion=="1":
        mostrar_resumen(base_de_datos)
    else:
        graficar_ticker(base_de_datos)
```

```python
def guardar_informacion(base_de_datos,ticker, fecha_desde, fecha_hasta,info_polygon):
    con = sqlite3.connect(base_de_datos)
    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['t'],stock_info['n'])
        cur.execute(sql_insert,valores)

    con.commit()
    con.close()
```

```python
def convertir_timestamp_epoc_a_fecha(t):

    t_segs = t/1000
    fecha_act_aux = datetime.datetime.fromtimestamp(t_segs)
    fecha_convert=datetime.datetime.strftime(fecha_act_aux,"%Y-%m-%d")
    return fecha_convert
```

```python
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()
```

```python
def graficar_ticker(base_de_datos):

    print("graficar_ticker")
    texto_opcion_graficar_ticker="Ingrese ticker a graficar"
    ticker_elegido=input(texto_opcion_graficar_ticker)
    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"
    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.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

    fecha_min=fechas[0]
    fecha_max=fechas[len(fechas)-1]

    titulo="Evolución ticker "+ticker_elegido+" "+fecha_min+" - "+fecha_max
    plt.title(titulo) ## Título de Gráfica

    nombre_grafico=ticker_elegido+"-"+fecha_min+"-"+fecha_max+".jpg"
    plt.savefig(nombre_grafico)

    con.close()
```

```python
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
```

```python
if __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:
            print(e)
    elif opcion=="2":
        try:
            opcion2(base_de_datos)
        except Exception as e:
            print(e)
    else:
        print("Opción incorrecta")
```

```python
def opcion1(base_de_datos):

    ticker,desde,hasta = solicitar_datos_opcion1()
    print("Pidiendo datos...")
    time.sleep(3)
    timespan="day"
    resultados_api=llamar_api_polygon(ticker,timespan,desde,hasta)
    guardar_informacion(base_de_datos,ticker,desde,hasta,resultados_api)
    print("Operación exitosa")
```

```python
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: ')
    validar_fecha(hasta)
    return ticker,desde,hasta
```

```python
def validar_fecha(fecha):
    format = "%Y-%m-%d" #valida en el
    if len(fecha) != 10:
        raise ValueError("Formato de fecha erróneo. Ingresar AAAA-MM-DD")
    if fecha[4] != '-' or fecha[7] != '-':
        raise ValueError("Formato de fecha erróneo. Ingresar AAAA-MM-DD")
    try:
        res = bool(datetime.datetime.strptime(fecha, format))
    except ValueError:
        raise ValueError("Fecha errónea")
    return True
```

```python
def llamar_api_polygon(ticker, timespan, fecha_desde, fecha_hasta):

    key = 'eWjcDZNVWFJ5RC_xwFvz8qbuS4SR1huN'
    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.json()

    if data['resultsCount'] == 0:
        raise ValueError("No existen datos a listar. Verifique los datos ingresados")

    return data['results']
```

```python
def guardar_informacion(base_de_datos,ticker, fecha_desde, fecha_hasta,info_polygon):
    con = sqlite3.connect(base_de_datos)
    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 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 (?,?,?,?,?,?,?,?,?,?)"
        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()
```

DIAGRAMA FUNCIONAL DE CONCATENACIÓN DE FUNCIONES

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.

```
if __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:
            print(e)
    elif opcion=="2":
        try:
            opcion2(base_de_datos)
        except Exception as e:
            print(e)
    else:
        print("Opción incorrecta")
```

1

```
C:\Users\ggimeno3\OneDrive\Documentos personales\Python\ITBA\Ejercicio final>python Integrador9.py
Seleccione una opción
1. Actualización de datos
2. Visualización de datos
1
```

2

```
def opcion1(base_de_datos):

    ticker,desde,hasta = solicitar_datos_opcion1()
    print("Pidiendo datos...")
    time.sleep(3)
    timespan="day"
    resultados_api=llamar_api_polygon(ticker,timespan,desde,hasta)
    guardar_informacion(base_de_datos,ticker,desde,hasta,resultados_api)
    print("Operación exitosa")
```

3

4

5

```
Ingresar ticker a pedir:AAPL
Ingrese fecha de inicio en este formato año-mes-día:  2022-05-11
Ingrese fecha de fin en este formato año-mes-día: 2022-05-12
```

```
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.

```
if __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:
            print(e)
    elif opcion=="2":
        try:
            opcion2(base_de_datos)
        except Exception as e:
            print(e)
    else:
        print("Opción incorrecta")
```

```
def opcion1(base_de_datos):

    ticker,desde,hasta = solicitar_datos_opcion1()
    print("Pidiendo datos...")
    time.sleep(3)
    timespan="day"
    resultados_api=llamar_api_polygon(ticker,timespan,desde,hasta)
    guardar_informacion(base_de_datos,ticker,desde,hasta,resultados_api)
    print("Operación exitosa")
```

5

```
Ingresar ticker a pedir:AAPL
Ingrese fecha de inicio en este formato año-mes-día:  2022-05-11
Ingrese fecha de fin en este formato año-mes-día: 2022-05-12
```

6

7

```
def llamar_api_polygon(ticker, timespan, fecha_desde, fecha_hasta):

    key = 'eWjcDZNVWFJ5RC_xwFvz8qbuS4SR1huN'
    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.json()
    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

```
if __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:
            print(e)
    elif opcion=="2":
        try:
            opcion2(base_de_datos)
        except Exception as e:
            print(e)
    else:
        print("Opción incorrecta")
```

```
def opcion1(base_de_datos):

    ticker,desde,hasta = solicitar_datos_opcion1()
    print("Pidiendo datos...")
    time.sleep(3)
    timespan="day"
    resultados_api=llamar_api_polygon(ticker,timespan,desde,hasta)
    guardar_informacion(base_de_datos,ticker,desde,hasta,resultados_api)
    print("Operación exitosa")
```

8

```
def guardar_informacion(base_de_datos,ticker, fecha_desde, fecha_hasta,info_polygon):
    con = sqlite3.connect(base_de_datos)
    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['t'],stock_info['n'])
        cur.execute(sql_insert,valores)
```

9

```
def convertir_timestamp_epoc_a_fecha(t):

    t_segs = t/1000
    fecha_act_aux = datetime.datetime.fromtimestamp(t_segs)
    fecha_convert=datetime.datetime.strftime(fecha_act_aux,"%Y-%m-%d")
    return fecha_convert
```

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

```
if __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:
            print(e)
    elif opcion=="2":
        try:
            opcion2(base_de_datos)
        except Exception as e:
            print(e)
    else:
        print("Opción incorrecta")
```

```
def opcion2(base_de_datos):

    opcion=mostrar_opcion2_sub_opciones()
    if opcion=="1":
        mostrar_resumen(base_de_datos)
    else:
        graficar_ticker(base_de_datos)
```

```
C:\Users\ggimeno3\OneDrive\Documentos personales\Python\ITBA\Ejercicio final>python tp_polygon.py
Seleccione una opción
1. Actualización de datos
2. Visualización de datos
Ingrese opcion:2
Visualización de datos - Seleccione una opción
1. Resumén
2. Grafico de Ticker1
AAPL - 2021-12-01 <-> 2022-09-19
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
```

```python
if __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:
            print(e)
    elif opcion=="2":
        try:
            opcion2(base_de_datos)
        except Exception as e:
            print(e)
    else:
        print("Opción incorrecta")
```

```python
def opcion2(base_de_datos):

    opcion=mostrar_opcion2_sub_opciones()
    if opcion=="1":
        mostrar_resumen(base_de_datos)
    else:
        graficar_ticker(base_de_datos)
```

```
C:\Users\ggimeno3\OneDrive\Documentos personales\Python\ITBA\Ejercicio final>python tp_polygon.py
Seleccione una opción
1. Actualización de datos
2. Visualización de datos
Ingrese opcion:2
Visualización de datos - Seleccione una opción
1. Resumén
2. Grafico de Ticker1
AAPL - 2021-12-01 <-> 2022-09-19
TSLA - 2022-05-02 <-> 2022-07-01
```

```python
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()
```

```
if __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:
            print(e)
    elif opcion=="2":
        try:
            opcion2(base_de_datos)
        except Exception as e:
            print(e)
    else:
        print("Opción incorrecta")
```

```
def opcion2(base_de_datos):

    opcion=mostrar_opcion2_sub_opciones()
    if opcion=="1":
        mostrar_resumen(base_de_datos)
    else:
        graficar_ticker(base_de_datos)
```

```
def graficar_ticker(base_de_datos):

    print("graficar_ticker")
    texto_opcion_graficar_ticker = "Ingrese el acrónimo del ticker a graficar. Ejemplo AAPL\nNombre del Ticker: "
    ticker_elegido = input(texto_opcion_graficar_ticker)
    con = sqlite3.connect(dase_de_datos)
    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')  # Leyenda 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()
```