**Binance Client Code snippets**

Rewritten Functions:

class BinanceClient:  
 def \_\_init\_\_(self, api\_key: str, api\_secret: str, testnet: bool, futures: bool):

# this function will communicate with the Binance API  
# it permits requesting, sending and deleting info when interacting with the API  
# returns the response in a JSON format, which would be a dictionary

def \_make\_request(self, method: str, endpoint: str, data: typing.Dict):

# public endpoints  
# get the possible contracts like BTC/USDT or ETH/USDT for example  
# returns a dictionary of contracts

def get\_contracts(self) -> typing.Dict[str, Contract]:

# in charge of getting the amount of all cryptos the user has  
# returns a dict of Balance

def get\_balances(self) -> typing.Dict[str, Balance]:

# manages to get the historical candlestick, up to 1000  
# receives the contract and the time interval, 1m, 5m, 15m and so on  
# returns a list of Candle

def get\_historical\_candles(self, contract: Contract, interval: str) -> typing.List[Candle]:

def place\_order(self, contract: Contract, order\_type: str, quantity: float, side: str, price=None, tif=None) -> \  
 OrderStatus:

def \_hashing(self, query\_string: str):  
 return hmac.new(  
 self.\_secret\_key.encode('utf-8'), query\_string.encode('utf-8'), hashlib.sha256  
 ).hexdigest()  
  
def \_get\_timestamp(self):  
 return int(time.time() \* 1000)  
  
def \_dispatch\_request(self, http\_method: str):  
 session = requests.Session()  
 session.headers.update(  
 {"Content-Type": "application/json;charset=utf-8", "X-MBX-APIKEY": self.\_public\_key}  
 )  
 return {  
 "GET": session.get,  
 "DELETE": session.delete,  
 "PUT": session.put,  
 "POST": session.post,  
 }.get(http\_method, "GET")  
  
def \_send\_signed\_request(self, http\_method: str, url\_path: str, payload={}):  
 query\_string = urlencode(payload, True)  
 if query\_string:  
 query\_string = "{}&timestamp={}".format(query\_string, self.\_get\_timestamp())  
 else:  
 query\_string = "timestamp={}".format(self.\_get\_timestamp())  
  
 url = (  
 self.\_base\_url + url\_path + "?" + query\_string + "&signature=" + self.\_hashing(query\_string)  
 )  
  
 print("{} {}".format(http\_method, url))  
 params = {"url": url, "params": {}}  
 response = self.\_dispatch\_request(http\_method)(\*\*params)  
 return response.json()

def \_send\_public\_request(self, url\_path: str, payload={}):  
 query\_string = urlencode(payload, True)  
 url = self.\_base\_url + url\_path  
 if query\_string:  
 url = url + '?' + query\_string  
 print('{}'.format(url))  
 response = self.\_dispatch\_request('GET')(url=url)  
 return response.json()