Documentation cryprocoins/api_coinmarketcap_impl/

git: https://github.com/nvinnikov/cryprocoins

UI

> launch ui.py

> then you can use help by:

> for quit:

> for other commands:

```
--a
['BTC', 'LTC', 'NMC', 'TRC', 'PPC', 'FTC', 'FRC', 'IXC', 'WDC', 'DGC', 'GLC', 'FST', 'PXC', 'MEC', 'IFC', 'XPM', 'ANC', 'CSC', 'EMD',
--h
Write coin symbol or:
(--h) - help
(--q) - quit
(--x) - example
(--a) - all coins
--x
[['Bitcoin', 'BTC'], ['Litecoin', 'LTC'], ['Namecoin', 'NMC'], ['Terracoin', 'TRC'], ['Peercoin', 'PPC'], ['Feathercoin', 'FTC'], ['Fr
Example:
BTC
Bitcoin (BTC) is a cryptocurrency
```

> then you can write a token symbol and get token info and token description:

```
--a
['BTC', 'LTC', 'NMC', 'TRC', 'PPC', 'FTC', 'FRC', 'IXC', 'WDC', 'DGC', 'GLC', 'FST', 'PXC', 'MEC', 'IFC', 'XPM', 'ANC', 'CSC', 'EMD',
BTC
Bitcoin (BTC) is a cryptocurrency . Users are able to generate BTC through the process of mining. Bitcoin has a current supply of 18,8
Bitcoin is 48,144.78758812 USD
BTC is up 0.79 over the last 24 hours.
TONCOIN
TON Coin (TONCOIN) is a cryptocurrency . TON Coin has a current supply of 0. The last known price of TON Coin is 0.74600113 USD and is
TON Coin is 0.74600113 USD
TONCOIN is down -7.21 over the last 24 hours.
SOL
Solana (SOL) is a cryptocurrency launched in 2020. Solana has a current supply of 505,530,667.6799421 with 297,840,085.56316054 in cir
Solana is 172.64128647 USD
SOL is up 3.58 over the last 24 hours.
```

QUOTES

> quotes.py

> here describe MarketFeel and Quotes classes

> in MarketFeel class describe to functions:

```
def fells(self):
        return ('BTC dominance today ' + self.q.btc_dominance_today() + '%, 24 hours change ' +
                self.q.btc_dominance_24h_percentage_change() + '%' + '\n' +
                'ETH dominance today ' + self.q.eth_dominance_today() + '%, 24 hours change ' +
                self.q.eth_dominance_24h_percentage_change() + '%' + '\n' +
                'Active ' + self.q.active_market_pairs() + ' pairs' + '\n' +
                'Total cryptocoins ' + self.q.total_cryptocurrencies() + '\n')
```

```
    def market_status(self):
        if float(self.q.btc_dominance_24h_percentage_change()) >= 0:
            return 'up'
        else:
            return 'down'
```

> feels describe crypto market today

> market_status describe about BTC dominance in crypto markets, BTC capitalisation the biggest, so we can predict crypto market about 24 hours

CRYPTOCURRENCY

> cryptocurrency.py

> here describe Cryptocurrency class

```
class Cryptocurrency(object):
    def __init__(self, symbol=None, coin_id=None):
        self.symbol = symbol
        self.coin_id = coin_id

    def _get_map(self):
        _url = 'map'
        url = base_url + cryptocurrency_url + _url
        response = requests.get(url, headers=headers)
        mp = _expt(response)
        return mp['data']

    def get_20_coins(self):
        mp = self._get_map()
        coins_map = []
        for i in mp:
            idd = []
            symbol = i['symbol']
            name = i['name']
            idd.append(name), idd.append(symbol)
            coins_map.append(idd)
            if len(coins_map) > 20:
                return coins_map
        return coins_map

    def _get_info_by_symbol(self, symbol):
        _url = 'info?symbol='
        url = base_url + cryptocurrency_url + _url + symbol
        response = requests.get(url, headers=headers)
        mp = _expt(response)
        return mp['data']

    def _get_info_by_coin_id(self, coin_id):
        _url = 'info?id='
        url = base_url + cryptocurrency_url + _url + str(coin_id)
        response = requests.get(url, headers=headers)
        mp = _expt(response)
        return mp['data']

    def get_symbol_id(self, symbol):
        mp = self._get_info_by_symbol(symbol)
        return mp[symbol]['id']

    def get_coin_description_by_symbol(self, symbol):
        mp = self._get_info_by_symbol(symbol)
        return mp[symbol]['description']

    def get_coin_description_by_id(self, coin_id):
        mp = self._get_info_by_coin_id(coin_id)
        return mp[coin_id]['description']

    def get_all_symbols(self):
        all = []
        mp = self._get_map()
        for i in mp:
            ii = i['symbol']
            all.append(ii)
        return all
```

```
    def get_market_price(self, symbol):
        string = self.get_coin_description_by_symbol(symbol)
        f = string.find('known price of') + 15 #151+15
        l = string.find('USD and is') + 3 #193+3
        return string[f:l]

    def get_market_statistic(self, symbol):
        string = self.get_coin_description_by_symbol(symbol)
        f = string.find(' USD and is ') + 9
        l = string.find(' It is currently trading')
        return symbol + ' ' + string[f:l]
```

FIAT

> fiat.py

> here describe Fiat class

> in progress