Generic - Code Snippets

Generic Python3 code snippets.

To Do

- ☐ ST3: study & improve `subl` utility. open files & folders. custom options to open projects with customizable configurations. eg. open project in `Distraction-Free Mode` with code on side, info on the right, and sliding directory listing.
- ☐ Trim `slides.mp4`
- ☐ VLC: modify vlc.exe to create playlists from videos in directory.
- ☐ Telegram: bot/hack to send command line msg (including links, multimedia) to multiple contacts. eg. send link to blog post to friends to comment upon.
- ☐ Lookup/Ask friends if DMRC has a link similar to Beazley's `bustracker.py`
- ☐ check Re-import Modules from beazley's vids. readup `importlib` and `imp`.
- [ ] https://github.com/{username}/{repository}/blob/{branch} https://github.com/{username}/{repository}/find/{branch} eg. https://github.com/inovizz/demystifyi docker-for-devs/blob/master/tree https://github.com/inovizz/demystifying-docker-for-devs/find/master/tree
- ☒ Open folder in Explorer

```
start . // current folder
```

Regular Expressions (RegEx)

To-Do

- ☐ Curate a table of expressions with examples.
- ☐ Python: Use RegEx.
- ☐ Python: Implement RegEx in code.

Expression	Meaning
.	Any character (except newline)
\d, \D	Digit (0-9), !Digit
\w, \W	Word (a-z, A-Z, 0-9, _), !Word

Expression	Meaning
`\s`, `\S`	Whitespace (space, tab, newline), !Whitespace
`\b`, `\B`	WordBoundary, !WordBoundary
`^`, `$`	Beginning/End of String
`[],(),{}`	CharacterSet, WordGroup, #Values

- Quantifier

Expression	Meaning
`0`	>=0
`+`	>=1
`?`	0 or 1
`{n}`	Exact value (=n)
`{start, stop}`	Range of values (min, max)

Examples

- Phone Numbers +91-95774-21050 +91-94273-18798 +91-81340-14187 `\d\d.\d\d\d\d\d.\d\d\d\d\d` `\+\d\d[-]\d\d\d\d\d[-]\d\d\d\d\d` `\+\d{2}-?\d{5}-?\d{5}` +91-95774-21050 to 9577421050 `(\+\d{2})-?(\d{5})-?(\d{5})` `$2$3`
- Character from the alphabet. `[a-zA-Z]`
- Anything except characters of the alphabet. `[^a-zA-Z]`
- HTTP URLs. `https?://(www\.)?\w+\.\w+`
- Rhyming words (cat, mat, bat, rat). `at\b`

☐ Rhyming words (cat, mat, bat, rat, rattatat).

```
#!/usr/bin/env python3
import re


text = '''
abc1234!
'''
pattern = re.compile(r'(\bDoe\b)')
matches = pattern.finditer(text)
for match in matches:
    print(match)
```

```
1. Get Source & Destination directories from header.
\b(Source|Dest)\s+:\s(.+)
\b(?P<type>Source|Dest)\s+:\s(?P<dir>.+)

2. Get Error message.
# timestamp
```

```
(\d{4}/\d{2}/\d{2}\s+\d{2}:\d{2}:\d{2})
(\d{4}/\d{2}/\d{2}\s+\d{2}:\d{2}:\d{2})
\d{4}(/\d{2}){2}\s+(\d{2}:?){3}
# error
\bERROR\s+(.+)

3. Get metrics table
\b(Dir|File|Byte|Time)s\s+:(\s+[\d:]+){1,}
```

Basics

- Import/De-Import/Re-import/CheckIfImported Modules.

```
import this
del this
```

- Check All Available Local Packages.

```
pip list
```

- Clear Screen from interpreter `import subprocess; subprocess.run('cls', shell=True); del subprocess;`
- Display `dict`'s items

```
for key, value in this.__dict__.items():
    print(key, value, sep='\t')
```

Email

[TUTORIAL]

imgurpython

[LINK] client_id = 125f7c9d3bcfdde client_secret = 24fd72c0cd485d1713bc069d02cabe46d4df3309

```
#! /usr/bin/env python3
import configparser
from imgurpython import ImgurClient


config = configparser.ConfigParser()
config.read('auth.ini')
client_id = config.get('credentials', 'client_id')
client_secret = config.get('credentials', 'client_secret')

client = ImgurClient(client_id, client_secret)
```

urllib

Read Ch12.

```
Uses:
1. Fix the `imgurpython` code to avoid Selenium, and use std lib instead.
2. Solve this problem:
- [] Scrap problem statements off CodeChef website.
    - Log into account using credentials.
    - Scrap problem statements.
    - Scrap problem solutions.
month = ['JAN', 'FEB', 'MARCH', 'APRIL', 'MAY', 'JUNE', 'JULY', 'AUG', 'SEPT', 'OCT', 'NOV',
division = ['A', 'B']
URL = f'https://www.codechef.com/{month}{division}'
https: // www.codechef.com / SEPT19A
```

wsgiref

Simple webpage

```
#! /usr/bin/env python3
from wsgiref.simple_server import make_server

html = b'''
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>CRTejaswi</title>
    </head>
    <body>
        <h1> Dancing In The Rain </h1>
        <p align="left">
            <video src="https://i.imgur.com/hukaORO.mp4" height="360" controls preload></vi
        </p>
    </body>
    </html>
'''


def app(environment, start_response):
    status = '200 OK'
    headers = [('Content-type', 'text/html; charset=utf-8')]
    start_response(status, headers)
    return [html]
```

```
with make_server('', 8000, app) as httpd:
    print('Serving on Port 8000 ...')
    httpd.serve_forever()
```

```
Serving on Port 8000 ...
127.0.0.1 - - [30/Sep/2019 21:34:51] "GET/HTTP/1.1" 200 416
```

flask

Basic Application

v1 > CHANGES: Added routes, GET/POST json & form data, QueryString.

```
#! /usr/bin/env python3
import time
from flask import Flask, jsonify, request, redirect, url_for
app = Flask(__name__)
app.config['DEBUG'] = True


@app.route('/', defaults={'name': 'User'})
@app.route('/usr/<name>')
def index(name):
    return f'<h1>Welcome, {name}!</h1>'


@app.route('/json')
def json():
    return jsonify({'name': 'Chaitanya Tejaswi', 'age': 22, 'availability': [1, 2, 3, 4, 5]}


@app.route('/query', methods=['GET', 'POST'])
def query():
    name, location = request.args.get('name'), request.args.get('location')
    return f'<h1>Query Page</h1><p align="left">Hi, {name} from {location}!</p>'


@app.route('/form')
def form():
    return '''
        <h1>Form Fillup</h1>
        <form method="POST" action="/resultForm">
            Name:
                <input type="text" name="name"> <br>
            Location:
                <input type="text" name="location"> <br>
```

```
            <input type="submit" value="Submit">
        </form>
    '''


@app.route('/resultForm', methods=['GET', 'POST'])
def resultForm():
    if request.method == 'GET':
        # Redirecting ...
        time.sleep(2)
        return redirect(url_for('index'))
    elif request.method == 'POST':
        name, location = request.form['name'], request.form['location']
        return f'<h1>Form Results</h1><p align="left">Hi, {name} from {location}!</p>'


@app.route('/resultJson', methods=['POST'])
def resultJson():
    data = request.get_json()
    name, age, location = data['name'], data['age'], data['location']
    return jsonify(f'<h1>JSON Results</h1><p align="left">Hi, {name}_{age} from {location}!<


if __name__ == '__main__':
    app.run()
```

v2.1 > CHANGES: Added database query.

```
#! /usr/env/bin python3
import time
import sqlite3
from flask import Flask, jsonify, request, redirect, url_for, g
app = Flask(__name__)
app.config['DEBUG'] = True


@app.route('/', defaults={'name': 'User'})
@app.route('/usr/<name>')
def index(name):
    return f'<h1>Welcome, {name}!</h1>'


@app.route('/json')
def json():
    return jsonify({'name': 'Chaitanya Tejaswi', 'age': 22, 'availability': [1, 2, 3, 4, 5]}
```

```
@app.route('/form')
def form():
    return '''
        <h1>Form Fillup</h1>
        <form method="POST" action="/resultForm">
            Name:
                <input type="text" name="name"> <br>
            Location:
                <input type="text" name="location"> <br>
            <input type="submit" value="Submit">
        </form>
    '''


@app.route('/resultForm', methods=['GET', 'POST'])
def resultForm():
    if request.method == 'GET':
        # Redirecting ...
        time.sleep(2)
        return redirect(url_for('index'))
    elif request.method == 'POST':
        name, location = request.form['name'], request.form['location']
        return f'<h1>Form Results</h1><p align="left">Hi, {name} from {location}!</p>'


@app.route('/resultJson', methods=['POST'])
def resultJson():
    data = request.get_json()
    name, age, location = data['name'], data['age'], data['location']
    return jsonify(f'<h1>JSON Results</h1><p align="left">Hi, {name}_{age} from {location}!<


@app.route('/query', methods=['GET', 'POST'])
def query():
    name, location = request.args.get('name'), request.args.get('location')
    return f'<h1>Query Page</h1><p align="left">Hi, {name} from {location}!</p>'


'''
Database Implementation
    Query a list of books from database
'''


def connect_db(filename):
    connection = sqlite3.connect(filename)
```

```
    # connection.row_factory = sqlite3.Row
    return connection


def get_db(filename):
    if not hasattr(g, 'sqlite3'):
        g.db = connect_db(filename)
    return g.db


@app.teardown_appcontext
def close_db(error):
    if hasattr(g, 'db'):
        g.db.close()


@app.route('/queryDB', methods=['GET', 'POST'])
def queryDB():
    db = get_db('books.db')
    cursor = db.cursor()
    cursor.execute('SELECT * FROM books')
    results = cursor.fetchall()
    return f'<h1>DB Query: Results</h1>{results}'


if __name__ == '__main__':
    app.run()
```

v2.2 > CHANGES: Revamped Form to make a Book Entry. Book can be queried at `/queryDB`.

```
#! /usr/env/bin python3
import time
import sqlite3
from flask import Flask, jsonify, request, redirect, url_for, g
app = Flask(__name__)
app.config['DEBUG'] = True


@app.route('/', defaults={'name': 'User'})
@app.route('/usr/<name>')
def index(name):
    return f'<h1>Welcome, {name}!</h1>'


@app.route('/json')
def json():
```

```
    return jsonify({'name': 'Chaitanya Tejaswi', 'age': 22, 'availability': [1, 2, 3, 4, 5]}


@app.route('/form')
def form():
    return '''
        <h1>Book Entry</h1>
        <form method="POST" action="/resultForm">
            Title:
                <input type="text" name="title"> <br>
            Author:
                <input type="text" name="author"> <br>
            Pages:
                <input type="number" name="pages" maxlength="5"> <br>
            Publication Year:
                <input type="number" name="year" maxlength="5"> <br>
            <input type="submit" value="Submit">
        </form>
    '''


@app.route('/resultForm', methods=['GET', 'POST'])
def resultForm():
    if request.method == 'GET':
        # Redirecting ...
        time.sleep(2)
        return redirect(url_for('index'))
    elif request.method == 'POST':
        book = request.form.to_dict()
        db = get_db('books.db')
        cursor = db.cursor()
        cursor.execute("INSERT INTO books VALUES (?,?,?,?)", (book['title'], book['author'],
        db.commit()
        return f'<h1>Inserted!</h1><h2>Details</h2>\n{book}'


@app.route('/resultJson', methods=['POST'])
def resultJson():
    data = request.get_json()
    name, age, location = data['name'], data['age'], data['location']
    return jsonify(f'<h1>JSON Results</h1><p align="left">Hi, {name}_{age} from {location}!<


@app.route('/query', methods=['GET', 'POST'])
def query():
    name, location = request.args.get('name'), request.args.get('location')
```

```
    return f'<h1>Query Page</h1><p align="left">Hi, {name} from {location}!</p>'


'''
Database Implementation
    Query a list of books from database
'''


def connect_db(filename):
    connection = sqlite3.connect(filename)
    # connection.row_factory = sqlite3.Row
    return connection


def get_db(filename):
    if not hasattr(g, 'sqlite3'):
        g.db = connect_db(filename)
    return g.db


@app.teardown_appcontext
def close_db(error):
    if hasattr(g, 'db'):
        g.db.close()


@app.route('/queryDB', methods=['GET', 'POST'])
def queryDB():
    db = get_db('books.db')
    cursor = db.cursor()
    cursor.execute("SELECT * FROM books")
    results = cursor.fetchall()
    return f'<h1>DB Query: Results</h1>{results}'


if __name__ == '__main__':
    app.run()
```

Resources

[Async Tasks with Flask & Redis] [Async Tasks with Redis] [Django v Flask]

Implementing JSON Tokens

[VIDEO]

What Day Is It?

v1

Calculate Day-Of-Week for A Given Date

```
#!/usr/bin/env python3
import enum


class Week(enum.Enum):
    Sunday = 0
    Monday = 1
    Tuesday = 2
    Wednesday = 3
    Thursday = 4
    Friday = 5
    Saturday = 6


def day1(fulldate):
    '''
    Calculate day of week for a given day.
    Works for any date satisfying:
        2000 < year < 3000
    '''
    year, month, date = fulldate.split('-', maxsplit=2)
    year, month, date = int(year), int(month), int(date)

    # YEAR offset
    year %= 1000
    k = (year % 28)
    if k % 4 == 0:
        k = k // 4 - 1
    else:
        k = k // 4
    offset = (year + k) % 7

    Q, R = month // 2, month % 2
    X = 0
    if month == 1:
        offset -= 1
        Q, R = 1, 0
    if month == 2:
        R = 1
    if month == 9 or month == 11:
        X = 1
```

```
    return (offset + date + 5 * (Q - 1) + 2 * R + X) % 7


if __name__ == '__main__':
    date = input('Date (YYYY-MM-DD): ')
    for day in Week:
        if day.value == day1(date):
            print(day.name)
```

What Time Is It?

v1 [BROKEN]

Calculate time across multiple timezones.

```
#!/usr/bin/env python3
import enum


class TimeZoneA(enum.Enum):
    '''
    TimeZone     GMT    IST
    Delhi      +5:30      0
    Toronto    -4     -9:30
    NewYork    -4     -9:30
    Moscow     +3     -2:30
    London     +1     -4:30
    Tokyo      +9     +3:30
    Sydney     +11    +5:30
    '''
    Delhi = 0
    Toronto = -9.5
    NewYork = -9.5
    Moscow = -2.5
    London = -4.5
    Tokyo = 3.5
    Sydney = 5.5


class TimeZoneB(enum.Enum):
    '''
    TimeZone     GMT    IST
    Delhi      +5:30      0
    Toronto    -4     +2:30 (-)
    NewYork    -4     +2:30 (-)
    Moscow     +3     +9:30 (-)
```

```
    London      +1      +7:30 (-)
    Tokyo       +9      +3:30
    Sydney      +11     +5:30
    '''
    Delhi = 0
    Toronto = 2.5
    NewYork = 2.5
    Moscow = 9.5
    London = 7.5
    Tokyo = 3.5
    Sydney = 5.5


def timeNow(fullTime):
    hour, minute = fullTime.split(':', maxsplit=1)
    hour, minute = int(hour), int(minute)
    hour += minute / 60

    # TimeA = TimeZoneA()
    # for place in TimeA:
    #     place.value += hour
    return TimeZoneA


if __name__ == '__main__':
    time = input('Time (HH:MM): ')
    times = timeNow(time)
    for place in times:
        print(f'{place.name:10}: {place.value:5}')
```

```
Time (HH:MM): 10:10
Delhi     :     0
Toronto   :  -9.5
Moscow    :  -2.5
London    :  -4.5
Tokyo     :   3.5
Sydney    :   5.5
```