PaperWare.org

do.py

v3.5

133 line nordic design custom search syntax.

searches.txt

```
YouTube                   (yt)  https://www.youtube.com/results?search_query=michael+jackson
Bing                      (bi)  http://www.bing.com/search?q=michael+jackson&qs=n&form=QBLH&sp=-1&pq=michael+jackson&sc=8-15

PowerShell Tutorials      (ps)  do powershell tutorial yt
Thunderstruck AC/DC       (ac)  do thunderstruck yt

YouTube Podcast Search    (pod) do __ podcast yt
Default Search            (oo)  do __ bi

My First Custom Parser    (cus) my_custom_parser.py
My First Script           (scr) test.txt

Open In Chrome tab        (chr) chrome.py
Open in Firefox tab       (fox) fox.py
```

github.com/taext/do

```
# Do 3.5 (January 6 2017)

# What's New: Debugging parse_search_url()
# Next Action: Decide where to list -> str regarding piping
import re, sys, importlib


def modules():

    file_lines = [line.rstrip() for line in open('searches.txt')]
    return(file_lines)


def build_command_dict():

    c_dict = {}
    search_file_lines = modules()

    for search_line in search_file_lines:

        search_line = search_line.rstrip()
        m = re.search('\(([a-zA_Z0_9]+)\)', search_line)
        # parse searches.txt line for command variable: "- cmd -"

        if m:
            do_command = m.group(1)                         # get do command from match above
            command_name = search_line.split(" (")[0]       # get command name from first part of " - " split
            search_url = search_line.split(") ")[-1]        # get search URL from last part of " - " split

            c_dict[do_command] = (command_name, search_url)  # add to result dictionary

    return(c_dict)

command_dict = build_command_dict()


def parse_command_string(do_command_string):


    do_search_terms = do_command_string.split(" ")
    do_commands = build_command_dict()

    new_search_terms = []
    do_commands_found = []
    for element in do_search_terms:

        if element in do_commands:
            do_commands_found.append(element)
        else:
            new_search_terms.append(element)

    result_dict = {}
    result_dict['search_terms'] = new_search_terms
    result_dict['commands'] = do_commands_found

    return(result_dict)


def parse_search_url(url_string, search_term):

    url_string = url_string.lstrip()

    link_match = re.search('http', url_string)              # test for search URL
    do_match = re.search('^do ', url_string)                # test for do command
    variable_match = re.search('\s\_\_\s', url_string)      # test for variable character __
    parser_match = re.search('\.py$', url_string)           # test for custom_parser.py
    script_match = re.search('[a-zA-Z]\.txt', url_string)   # test for script_file.txt

    if script_match:

        result = parse_script(url_string)

        return(result)

    if parser_match:

        custom_parser_name = url_string[:-3]

        result = use_custom_parser(custom_parser_name, search_term)

        return(result)


    if variable_match:

        padded_search_term = " " + search_term + " "
        url_string = url_string.replace(" __ ", padded_search_term)

    if link_match:

        m = re.search('([Mm]ichael(.+?)[Jj]ackson)', url_string)
        if m:

            match_string = m.group(1)
            separating_char = m.group(2)

            new_search_term = search_term.replace(' ', separating_char)
            new_string = url_string.replace(match_string, new_search_term)

            return(new_string)

    if do_match:

        m = re.search('do (.+?$)', url_string)
        string_without_do = m.group(1)              # remove 'do ' part of string

        result = launch(string_without_do)

        return(result[0])

    else:
        print("url_string didn't get categorized by any filter, in parse_search_url(): " + url_string)
```

```
def launch(do_string):

    commands = command_dict
    elements = parse_command_string(do_string)
    search_terms = elements['search_terms']
    command_terms = elements['commands']
    search_string = ""

    for word in search_terms:
        search_string += word + " "

    search_string = search_string.rstrip()

    result_list = []

    if command_terms == []:
        command_terms.append('oo')

    while command_terms:

        try:
            last_command = command_terms.pop(0)     # get first argument command
        except:
                last_command = 'oo'

        try:
            url = commands[last_command][-1]        # use URL corresponding to last_command from commands dict
        except:
            print("Bad command?")

        result = parse_search_url(url, search_string)  # put search term(s) from do_string
        result_list.append(result)                     # into search URL from searches.txxt

    return(result_list)


def argv_to_string():

    arg_list = sys.argv[1:]

    arg_string = ""
    for arg in arg_list:
        arg_string += arg + " "

    arg_string2 = arg_string.rstrip()

    return arg_string2


def use_custom_parser(parser_module_name, arg_string):

    custom_parser = importlib.import_module(parser_module_name)
    result = custom_parser.run(arg_string)

    return(result)


def printify(result_passed):

    for item in result_passed:
        print(item)


def parse_script(filename):

    f = open(filename)
    lines = f.readlines()

    result_list = []
    for line in lines:
        strip_line = line.rstrip()
        do_result = launch(strip_line)
        result_list.append(do_result[0])

    return(result_list)


def parse_pipe(do_string):

    elements = do_string.split(",")
    result = ""

    while elements:

        do_string_build = result + elements.pop(0)
        result = launch(do_string_build)[0]

    return(result)


def go(do_string):

    pipe_match = re.search('\,', do_string)

    if pipe_match:

        result = parse_pipe(do_string)
        return(result)

    else:
        result = launch(do_string)
        return(result)



if __name__ == '__main__':

    arg_string = argv_to_string()
    result = go(arg_string)

    if isinstance(result, list):
        result = printify(result)
    else:
        print(result)
```