Code Lines Checker

> Author: 物联212 廖玮珑

功能

单文件

统计单文件的行数等信息，如果是python文件，则提供作业要求的信息，如果是其他文件，则只输出总行数。 结果输出到`result.json`

文件夹

获取文件夹内所有的（包括子文件夹）的py文件，并按照作业要求统计相关信息，输出到`result.json`

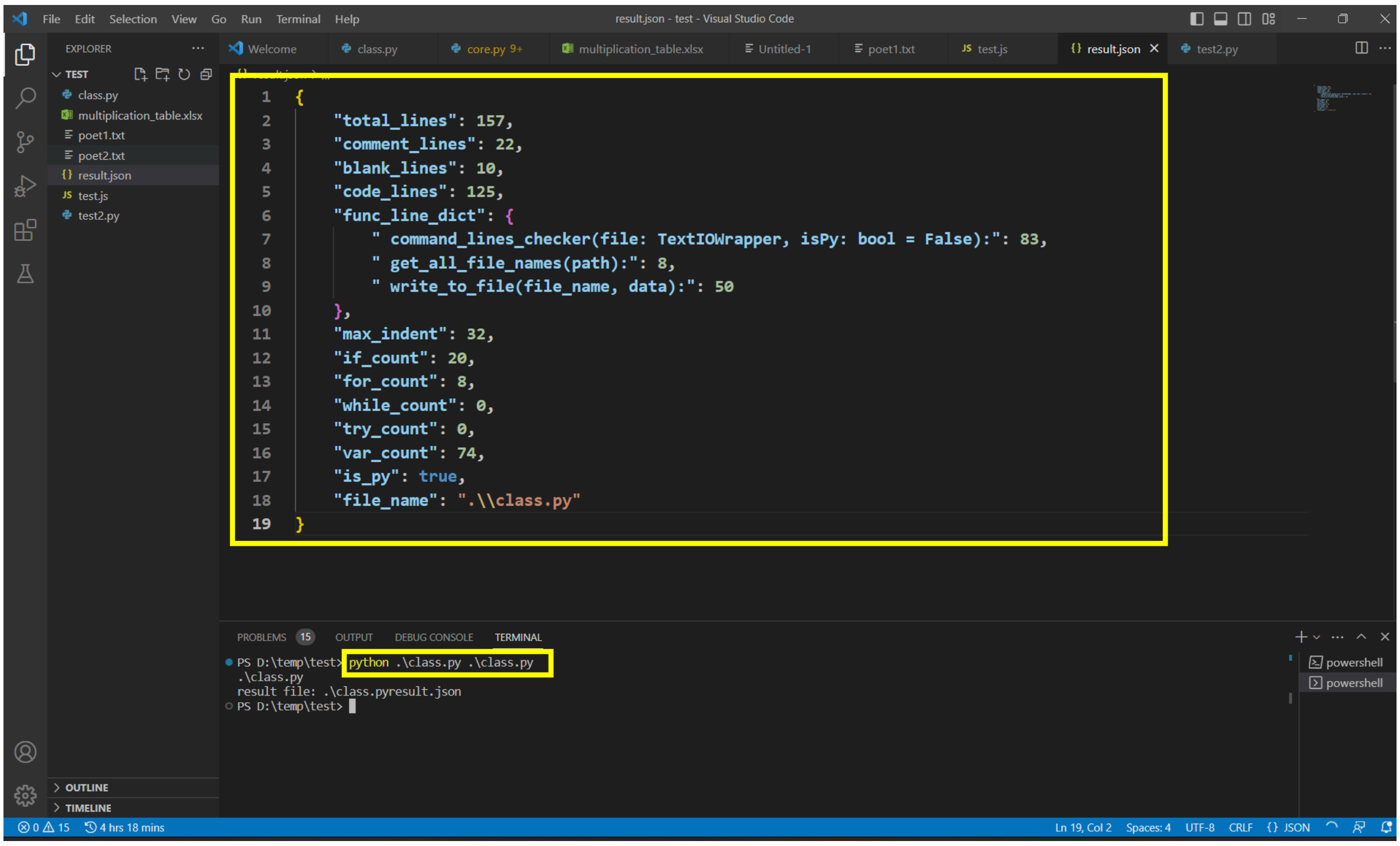
Get Start

在命令行参数加上想要统计的文件或文件夹的路径。

> 注意！文件夹后必须加上"\"，并且理论上统计文件夹功能目前仅支持Windows端。

Demo

单文件

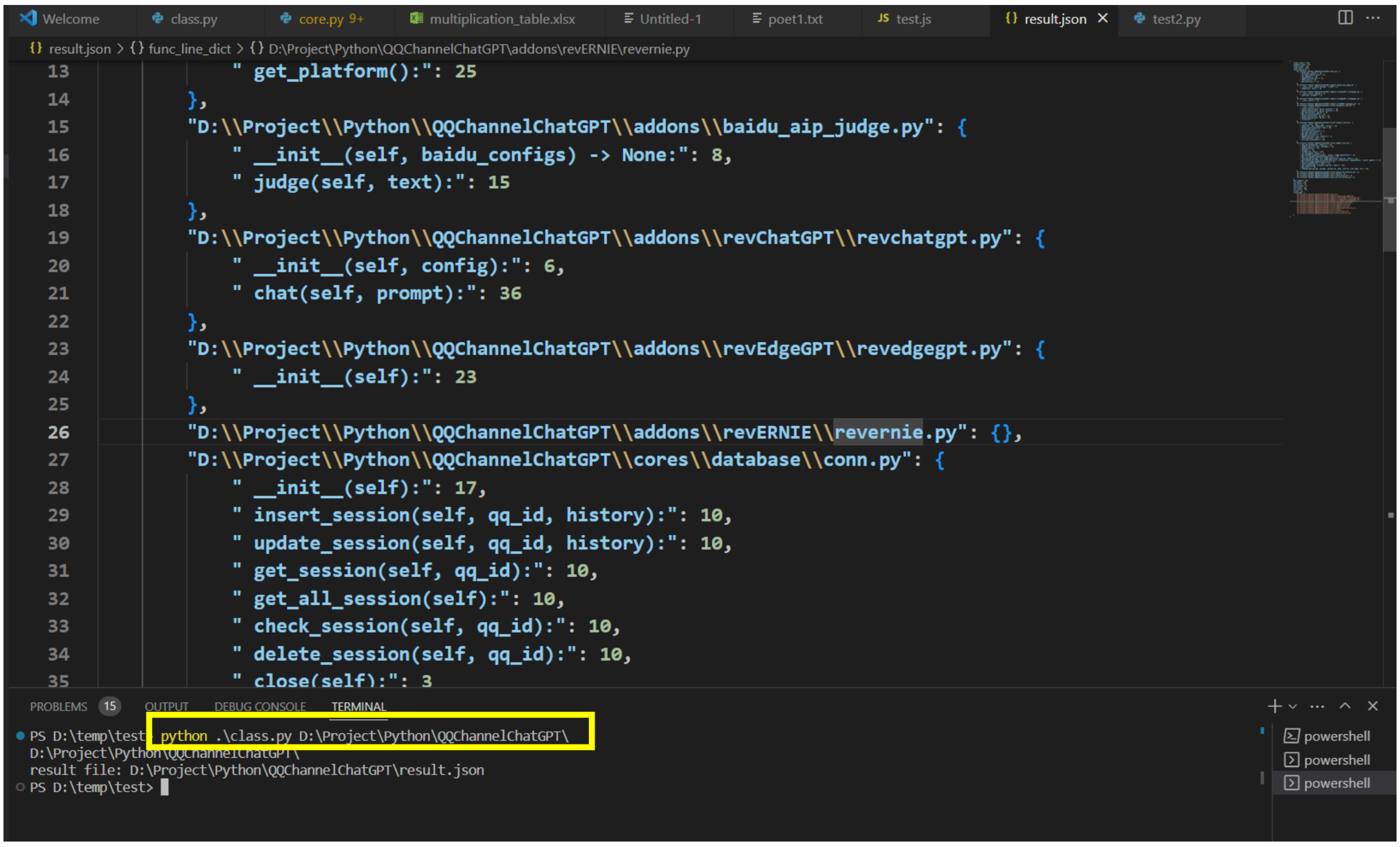


```
{
    "total_lines": 157,
    "comment_lines": 22,
    "blank_lines": 10,
    "code_lines": 125,
    "func_line_dict": {
        " command_lines_checker(file: TextIOWrapper, isPy: bool = False):": 83,
        " get_all_file_names(path):": 8,
        " write_to_file(file_name, data):": 50
    },
    "max_indent": 32,
    "if_count": 20,
    "for_count": 8,
    "while_count": 0,
    "try_count": 0,
    "var_count": 74,
    "is_py": true,
    "file_name": ".\\class.py"
}
```

```
PS D:\temp\test> python .\class.py .\class.py
.\class.py
result file: .\class.pyresult.json
PS D:\temp\test>
```

文件夹

以我的一个项目的文件夹为例



```
            " get_platform():": 25
        },
        "D:\\Project\\Python\\QQChannelChatGPT\\addons\\baidu_aip_judge.py": {
            " __init__(self, baidu_configs) -> None:": 8,
            " judge(self, text):": 15
        },
        "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revChatGPT\\revchatgpt.py": {
            " __init__(self, config):": 6,
            " chat(self, prompt):": 36
        },
        "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revEdgeGPT\\revedgegpt.py": {
            " __init__(self):": 23
        },
        "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revERNIE\\revernie.py": {},
        "D:\\Project\\Python\\QQChannelChatGPT\\cores\\database\\conn.py": {
            " __init__(self):": 17,
            " insert_session(self, qq_id, history):": 10,
            " update_session(self, qq_id, history):": 10,
            " get_session(self, qq_id):": 10,
            " get_all_session(self):": 10,
            " check_session(self, qq_id):": 10,
            " delete_session(self, qq_id):": 10,
            " close(self):": 3
```

```
PS D:\temp\test> python .\class.py D:\Project\Python\QQChannelChatGPT\
D:\Project\Python\QQChannelChatGPT\
result file: D:\Project\Python\QQChannelChatGPT\result.json
PS D:\temp\test>
```

```
{
    "total_lines": 1745,
    "comment_lines": 153,
        "blank_lines": 150,
        "code_lines": 1442,
        "func_line_dict": {
            "D:\\Project\\Python\\QQChannelChatGPT\\main.py": {
                " main(loop, event):": 19,
                " privider_chooser(cfg):": 11,
                " hot_update():": 60,
                " update_version(ver):": 31,
                " check_env():": 29,
                " get_platform():": 25
            },
            "D:\\Project\\Python\\QQChannelChatGPT\\addons\\baidu_aip_judge.py": {
                " __init__(self, baidu_configs) -> None:": 8,
                " judge(self, text):": 15
            },
            "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revChatGPT\\revchatgpt.py": {
                " __init__(self, config):": 6,
                " chat(self, prompt):": 36
            },
            "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revEdgeGPT\\revedgegpt.py": {
                " __init__(self):": 23
            },
            "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revERNIE\\revernie.py": {},
            "D:\\Project\\Python\\QQChannelChatGPT\\cores\\database\\conn.py": {
                " __init__(self):": 17,
                " insert_session(self, qq_id, history):": 10,
                " update_session(self, qq_id, history):": 10,
                " get_session(self, qq_id):": 10,
                " get_all_session(self):": 10,
                " check_session(self, qq_id):": 10,
                " delete_session(self, qq_id):": 10,
                " close(self):": 3
            },
            "D:\\Project\\Python\\QQChannelChatGPT\\cores\\openai\\core.py": {
                " __init__(self, cfg):": 19,
                " chat(self, req, image_mode = False):": 29,
                " handle_switch_key(self, req):": 30,
                " getConfigs(self):": 3,
                " save_key_record(self):": 4,
                " get_key_stat(self):": 2,
                " get_key_list(self):": 4,
                " append_key(self, key, sponsor):": 6,
                " check_key(self, key):": 17,
                " init_key_record(self):": 28
            },
            "D:\\Project\\Python\\QQChannelChatGPT\\cores\\qqbot\\core.py": {
                " new_sub_thread(func, args=()):": 22,
                " toggle_count(at: bool, message):": 20,
                " dump_history():": 24,
                " upload():": 38,
                " initBot(cfg, prov):": 145,
                " run_bot(appid, token):": 9,
                " get_chatGPT_response(context, request, image_mode=False):": 23,
                " get_rev_ChatGPT_response(prompts_str):": 32,
                " send_qq_msg(message, res, image_mode=False, msg_ref = None):": 18,
                " get_prompts_by_cache_list(cache_data_list, divide=False, paging=False, size=5, page=1):": 1
                " get_user_usage_tokens(cache_list):": 9,
                " check_frequency(id) -> bool:": 21,
                " oper_msg(message, at=False, msg_ref = None):": 253,
                " get_stat():": 28,
                " command_oper(qq_msg, message, session_id, name, user_id, user_name, at):": 152
        },
        "D:\\Project\\Python\\QQChannelChatGPT\\cores\\qqbot\\personality.py": {},
        "D:\\Project\\Python\\QQChannelChatGPT\\util\\log.py": {},
        "D:\\Project\\Python\\QQChannelChatGPT\\util\\unfit_words.py": {},
        "D:\\Project\\Python\\QQChannelChatGPT\\util\\errors\\errors.py": {}
    },
    "max_indent": 32,
    "if_count": 114,
    "for_count": 20,
    "while_count": 6,
    "try_count": 36,
    "var_count": 494,
    "is_py": true,
    "file_name": [
        "D:\\Project\\Python\\QQChannelChatGPT\\main.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\addons\\baidu_aip_judge.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revChatGPT\\revchatgpt.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revEdgeGPT\\revedgegpt.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\addons\\revERNIE\\revernie.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\cores\\database\\conn.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\cores\\openai\\core.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\cores\\qqbot\\core.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\cores\\qqbot\\personality.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\util\\log.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\util\\unfit_words.py",
        "D:\\Project\\Python\\QQChannelChatGPT\\util\\errors\\errors.py"
    ]
}
```

解释

Key解释

key	解释
total_lines	总行数
comment_lines	注释行数
blank_lines	空行数
code_lines	有效代码行数
func_line_dict	每个函数的行数
max_indent	最大缩进
if_count	if数
for_count	for数
while_count	while数
try_count	try数

key	解释
var_count	变量数
is_py	是否是py文件
file_name	文件名

关键代码解释

```
if __name__ == '__main__':
    # 得到参数
    args = sys.argv[1:]
    for i in args:
        print(i)
        # 检查是否是文件夹，如果是的话，则只统计.py文件，且得到文件夹下（包含子文件夹）的所有py文件
        if i.endswith('\\'):
            # 是文件夹
            total_list = [] # 用于存放所有文件的统计结果
            # 得到文件夹下的所有文件
            file_names = get_all_file_names(i)
            for j in file_names:
                if j.endswith('.py'):
                    with open(j, 'r', encoding='utf-8') as f:
                        # print("file name: " + f.name)
                        total_list.append(command_lines_checker(f, True))
            # 汇总结果
            total_dict = {}
            max_intent = 0
            for k in total_list:
                for key, value in k.items():
                    if key not in total_dict:
                        # 初始化键值对和一些特判
                        if key == 'func_line_dict':
                            total_dict['func_line_dict'] = {
                                k['file_name']: value
                            }
                        elif key == 'file_name':
                            total_dict['file_name'] = [value]
                        elif key == 'is_py':
                            total_dict['is_py'] = True
                        elif key == 'max_indent':
                            if value > max_intent:
                                max_intent = value
                            total_dict['max_indent'] = max_intent
                        else:
                            total_dict[key] = value
                    else:
                        # 汇总和一些特判，如max_indent：取最大值而不是相加
                        if key == 'func_line_dict':
                            total_dict['func_line_dict'][k['file_name']] = value
                        elif key == 'file_name':
                            total_dict['file_name'].append(value)
                        elif key == 'is_py':
                            pass
                        elif key == 'max_indent':
                            if value > max_intent:
                                max_intent = value
                            total_dict['max_indent'] = max_intent
                        else:
                            total_dict[key] += value
            # 写入文件
            write_to_file('result.json', total_dict)

        else:
            with open(i, 'r', encoding='utf-8') as f:
                if f.name.strip().endswith('.py'):
                    res = command_lines_checker(f, True)
                    write_to_file('result.json', res)
                else:
                    res = command_lines_checker(f, False)
                    write_to_file('result.json', res)
```

```
def command_lines_checker(file: TextIOWrapper, isPy: bool = False):
    total_lines = 0
    comment_lines = 0
    blank_lines = 0
    func_line_dict = {}
    max_indent = 0
    if_count = 0
    for_count = 0
    while_count = 0
    try_count = 0
    var_count = 0

    data_list = file.readlines()
    total_lines = len(data_list)

    if isPy:
        # 中间变量
        _current_func = None
        _current_func_line = 0
        for i in data_list:
            # 注释
            if i.strip().startswith('#'):
                comment_lines += 1
            # 空行
            elif i.strip() == '':
                blank_lines += 1
            # 函数
            elif i.strip().startswith('def'):
                _func_name = i.strip()[3:len(i)-1]
                if _current_func != _func_name:
                    # 新的函数
                    if _current_func is not None:
                        func_line_dict[_current_func] = _current_func_line
                    _current_func = _func_name
                    _current_func_line = 0
            else:
                # 检查缩进数
                indent = 0
                for j in i:
                    if j == ' ':
                        indent += 1
                    else:
                        break
```

```python
                if indent > max_indent:
                    max_indent = indent

            # 检查if
            if if_re.match(i):
                if_count += 1
            # 检查for
            if for_re.match(i):
                for_count += 1
            # 检查while
            if while_re.match(i):
                while_count += 1
            # 检查try
            if try_re.match(i):
                try_count += 1
            # 检查变量定义
            if var_re.match(i):
                var_count += 1

            if _current_func is not None:
                _current_func_line += 1

        if _current_func is not None:
            func_line_dict[_current_func] = _current_func_line
    res = {}
    res['total_lines'] = total_lines
    res['comment_lines'] = comment_lines
    res['blank_lines'] = blank_lines
    res['code_lines'] = total_lines - comment_lines - blank_lines
    res['func_line_dict'] = func_line_dict
    res['max_indent'] = max_indent
    res['if_count'] = if_count
    res['for_count'] = for_count
    res['while_count'] = while_count
    res['try_count'] = try_count
    res['var_count'] = var_count
    res['is_py'] = isPy
    res['file_name'] = file.name
    return res
```

```python
def get_all_file_names(path):
    # 得到文件夹下的所有文件路径
    file_paths = []
    for root, dirs, files in os.walk(path):
        # os.walk()返回目录路径，路径下所有子目录，路径下所有非目录子文件
        for file in files:
            file_paths.append(os.path.join(root, file))
    return file_paths
```