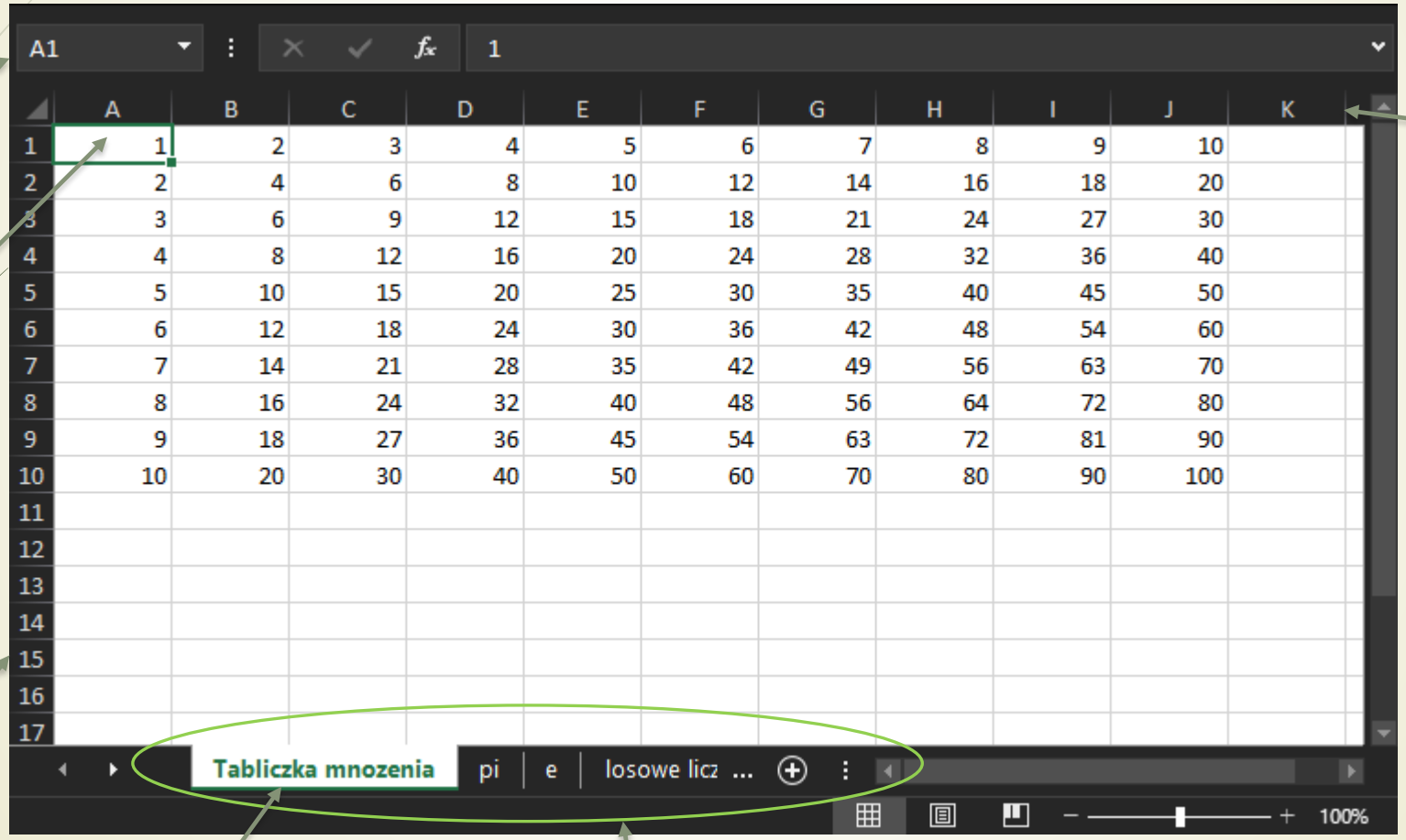
openpyxl

Tomasz Kumor

Struktura pliku Excel (.xlsx)



Adres aktywnej komórki

Aktywna komórka

Wiersze

Kolumny

Aktywny arkusz

Lista arkuszy

Openpyxl

Openpyxl jest to biblioteka języka Python obsługująca pliki programu Excel.

Podstawowe operacje w openpyxl

Tworzenie pliku

Żeby utworzyć nowy plik musimy najpierw zainicjować zeszyt, który będziemy zapisywać (czyli plik Excel).

Nowy pusty zeszyt tworzymy poprzez funkcje:

Workbook()

```
wb = Workbook()
wb.save("zapisywany_plik.xlsx")
```

Wczytujemy plik poprzez funkcje:

load_workbook(„nazwa_pliku.xlsx")

```
wb = load_workbook("wczytywany_plik.xlsx")
wb.save("zapisywany_plik.xlsx")
```

Zapisujemy plik metodą save().

Modyfikacje arkuszy

Tworze instancje nowego pustego zeszytu.

Następnie wybieram aktywny arkusz, któremu zmieniam nazwę na „Tabliczka mnozenia”.

W kolejnych 2 liniach tworzę nowy arkusz o nazwie „pi” i przypisuje go do zmiennej, a następnie wartość komórki „A1” ustawiam na „3.14”.

Na końcu tworzę kolejny arkusz o nazwie „e”, również przypisuję go do zmiennej oraz ustawiam wartość komórki o adresie „A1” na „2.718”.

```
wb = Workbook()

ws = wb.active
ws.title = "Tabliczka mnozenia"

ws = wb.create_sheet("pi")
ws["A1"] = 3.14

ws = wb.create_sheet("e")
ws.cell(row=1, column=1).value = 2.718
```

Iteracja w arkuszu

W iteracji po arkuszu pomocne są własności:

min_row – nr pierwszego (zapisanego) wiersza

min_column – nr pierwszej (zapisanej) kolumny

max_row – nr ostatniego (zapisanego) wiersza

max_column – nr ostatniej (zapisanej) kolumny

Podaną funkcję przedstawię w późniejszym slajdzie.

```python
def get_data(ws, search_query="^.*$"):
    """
    zwraca dane z arkusza
    :param ws: arkusz
    :param search_query: regexp
    :return: arr:
            jesli search_query ustawione -> zwraca pasujace wyniki,
            w przeciwnym wypadku -> zwraca wszystko
    """
    result_arr = []
    for i in range(ws.min_row, ws.max_row + 1):
        for j in range(ws.min_column, ws.max_column + 1):
            temp_value = ws.cell(row=i, column=j).value
            # sprawdzam czy komorka jest pusta
            if temp_value is None:
                continue
            # wyrazenie regularne (wyrazenie, sprawdzana wartosc)
            # jesli sie zgadza -> dodaje do zwracanej petli
            if re.search(search_query, str(temp_value)):
                result_arr.append({
                    "row": i,
                    "column": j,
                    "value": temp_value
                })
    return result_arr
```

Pomocnicze funkcje użyte w projekcie

Wyrażenia regularne

Wyrażenia regularne wykorzystywane w projekcie

```
confirm_query = "^[Tt1Yy]|[Tt][Aa][Kk]$"
reg_file_name_json = "^.+\\.json$"
reg_file_name_xlsx = "^.+\\.xlsx$"
```

Funkcja check_if_file_exist

Funkcja sprawdza czy istnieje plik w podanej lokalizacji o tej nazwie.

Do sprawdzenia czy plik istnieje wykorzystuje wbudowany moduł pathlib.

```python
def check_if_file_exist(file_name, quiet=False):
    """
    funkcja sprawdzajqca czy podany plik istnieje
    :param file_name: string
    :param quiet: boolean; jeśli True -> nic nie wyświetla
    :return: jeśli istnieje -> True, jeśli nie istnieje -> False
    """
    file = pathlib.Path(file_name)
    if file.exists():
        if quiet:
            return True

        confirm = input("Plik istnieje, czy chcesz go nadpisac? (t - tak): ")
        if not re.search(confirm_query, confirm):
            return True
    return False
```

Funkcja create_reg_exp_query

Funkcja, która tworzy wyrażenie regularne pobrane od użytkownika

```
def create_reg_exp_query():
    """
    funkcja pobierajaca wyrazenie regularne od uzytkownika

    :return: reg exp
    """

    print("""Opcje:
        1 - podstawowe wyszukiwanie (mozna uzyc wyrazenia regularnego)
        2 - wyszukiwanie zaawansowane z pomoca""")

    while True:
        choice = input("Wybor: ")
        if re.search("^[12]$", choice):
            break
        print("Podales bledna wartosc!")

    choice = int(choice)

    if choice == 1:
        return input("Podaj szukane haslo: ")

    # zaawansowane wyrazenie query
    query = ""

    if re.search(confirm_query, input("Czy mam szukac slow zaczynajacych sie od wyrazenia? (t - tak) ")):
        query = "^" + query

    query += input("Podaj zestaw znakow (np. [a-e]), lub szukane slowo (np. abc): ")

    while True:
        n = input("Ile mam szukac wystapien (np. 4), \
jezeli ma zawierac conajmniej x znakow dodaj po tej liczbie przecinek (np. 4,): ")
        if re.search("^[0-9]+[,]*$", n):
            break
        print("Podales nieprawidlowe dane!")
    query += "{" + n + "}"

    if re.search(confirm_query, input("Czy mam szukac slow konczoncych sie wyrazeniem? (t - tak) ")):
        query += "$"

    return query
```

Funkcja print_table

Funkcja, która wyświetla w konsoli listę słownikową (z parametru table).

Do wyświetlenia wykorzystuje zewnętrzną bibliotekę: pretty_table.

```
def print_table(table: List[dict]):
    """
    funkcja ktora wyswietla w konsoli dictionary

    :param table: lista dictonary
    """
    # jeśli tablica table jest pusta, to informujemy o tym uzytkownika
    if len(table) == 0:
        print("Brak wynikow!")
        return None

    # uzupelniamy tytul kluczami z pierwszego elementu
    titles = table[0].keys()

    pretty_table = PrettyTable()

    pretty_table.field_names = [title for title in titles]

    for cell in table:
        pretty_table.add_row([cell[title] for title in titles])

    print(pretty_table)
    print("Liczba znalezionych elementow: {}".format(len(table)))
```

Funkcja add_sheet_to_workbook

Funkcja dodaje arkusz z wartościami zapisanymi w liście słownikowej (table).

Jeśli jest wywołany z opcją overwrite=True, to nadpisuje ostatni arkusz (pomocne gdy nadpisujemy domyślnie utworzony arkusz).

```python
def add_sheet_to_workbook(table: List[dict], wb=Workbook(), title="Untitled", overwrite=False):
    """
    dodaj arkusz na podstawie listy z dictionary, parametry: (row, column, value)

    :param table: list[dict] - dane do zapisania do arkusza
    :param wb: workbook - plik gdzie ma byc zapisany wynik
    :param title: string - nazwa arkusza do zapisania wyniku
    :param overwrite: boolean - jesli prawda -> to nadpisze arkusz, jesli falsz -> doda nowy arkusz
    (parametr pomocny gdy jest utworzony nowy plik i chcemy nadpisac domyslny arkusz)
    :return: nowy worksheet
    """
    if overwrite:
        ws = wb.worksheets[-1]
        ws.title = title
    else:
        ws = wb.create_sheet(title)
    # kopiowanie wartosci komorek
    for cell in table:
        ws.cell(row=cell['row'], column=cell['column']).value = cell['value']
    return ws
```

Funkcja get_data

Podana funkcja zwraca pasujące wyniki do podanego wyrażenia regularnego (w parametrze search_query, który przyjmuje wartość domyślną).

```
def get_data(ws, search_query="^.*$"):
    """
    zwraca dane z arkusza
    :param ws: arkusz
    :param search_query: regexp
    :return: arr:
            jesli search_query ustawione -> zwraca pasujace wyniki,
            w przeciwnym wypadku -> zwraca wszystko
    """
    result_arr = []
    for i in range(ws.min_row, ws.max_row + 1):
        for j in range(ws.min_column, ws.max_column + 1):
            temp_value = ws.cell(row=i, column=j).value
            # sprawdzam czy komorka jest pusta
            if temp_value is None:
                continue
            # wyrazenie regularne (wyrazenie, sprawdzana wartosc)
            # jesli sie zgadza -> dodaje do zwracanej petli
            if re.search(search_query, str(temp_value)):
                result_arr.append({
                    "row": i,
                    "column": j,
                    "value": temp_value
                })
    return result_arr
```

Funkcja get_sheet

Funkcja zwraca arkusz pobrany od użytkownika.

Użytkownik wybiera arkusz wpisując jego nazwę.

```
def get_sheet(wb: Workbook, search_query=""):
    """
    wybiera arkusz
    :param wb: workbook - plik
    :param search_query: string - jesli cos jest przeszukiwane to pokaze query
    :return: arkusz wybrany przez uzytkownika
    """
    sheets = [wb.worksheets[i].title for i in range(len(wb.worksheets))]
    while True:
        if search_query == "":
            temp_sheet = input("Podaj arkusz (dostepne: {}): ".format(sheets))
        else:
            temp_sheet = input(
                "Podaj arkusz (dostepne: {}) w ktorym ma byc przeszukiwane wyrazenie ({}): ".format(sheets,
                                                                                                    search_query))
        if temp_sheet in sheets:
            ws = wb.worksheets[sheets.index(temp_sheet)]
            return ws
        print("Niepoprawna nazwa arkusza!")
```

Funkcja save_to_json

Funkcja zapisuje listę, lub słownik do pliku w formacie .json.

Do zapisu pliku wykorzystuję funkcję json.dump(), który przyjmuje jako parametry zapisywaną listę (, lub słownik) oraz plik uprzednio otwarty.

```
def save_to_json(table):
    file_name = input("Podaj nazwe pliku (lokalizacje) gdzie zapisac: ")

    if not re.search(reg_file_name_json, file_name):
        file_name = "{}.json".format(file_name)

    if check_if_file_exist(file_name):
        return "Nie zapisano pliku"

    # saving file
    try:
        with open("{}".format(file_name), "w") as json_file:
            json.dump(table, json_file)
            print("Liste zapisano w pliku {}".format(file_name))
            return "Plik zapisano"
    except IOError as e:
        print("Nie mozna zapisac pliku: ", e)
        if re.search(confirm_query, input("Czy chcesz sprobowac zapisac raz jeszcze? (t - tak) ")):
            return save_to_json(table)
```

Projekt

Menu wyboru

Pętla do…while, która kończy się gdy użytkownik wpisze 0.

Pętla pobiera od użytkownika numer opcji, z zakresu od 0 do długości listy.

Następnie sprawdza czy użytkownik wpisał 0, jeśli tak, to kończy program, jeśli nie, to wywołuje funkcje przypisaną do elementu listy.

```python
from components.menu import menu_option

while True:
    # print menu
    n = 1
    for option in menu_option:
        print("{} : {}".format(n, option["name"]))
        n += 1
    del n
    print("{} : {}".format(0, "koniec programu"))

    # user choice
    while True:
        choice = int(input("Wybierz opcje: "))
        if 0 <= choice <= len(menu_option):
            break
    if choice == 0:
        break

    # call function
    menu_option[choice - 1]["function"]()
```

Struktura pliku menu

W tym pliku głównie znajdują się opcje (menu wyboru), które w poprzednim slajdzie wykorzystywaliśmy.

Lista ma wewnątrz elementy słownikowe, zawierające opis funkcji, oraz funkcje która ma się wywołać.

```python
from components.copyFile import copy_file
from components.search import search
from components.jsonInputOutput import save_xlsx_to_json, save_json_to_xlsx
from components.basicFunctions import new_static_file, create_empty_file

menu_option = [
    {"name": "Nowy plik z szablonu", "function": new_static_file},
    {"name": "Nowy pusty plik xlsx", "function": create_empty_file},
    {"name": "Kopiuj plik", "function": copy_file},
    {"name": "Przeszukaj plik", "function": search},
    {"name": "Eksportuj dane z pliku xlsx do pliku JSON", "function": save_xlsx_to_json},
    {"name": "Importuj dane z pliku JSON do pliku xlsx", "function": save_json_to_xlsx}
]
```

Funkcja new_static_file

Funkcja tworzy nowy plik z szablonu.

Na początku pobiera od użytkownika nazwę pliku do zapisu (gdy użytkownik nie podał rozszerzenia, to dodajemy za niego).

Następnie sprawdzamy czy plik istnieje funkcją check_if_file_exist.

Następnie ustawiam tytuły, a także zmieniam wartości poszczególnych arkuszy.

Na końcu zapisuje plik.

```
def new_static_file():
    wb = Workbook()
    file_name = input("Podaj nazwe pliku: ")
    if not re.search(reg_file_name_xlsx, file_name):
        file_name = "{}.xlsx".format(file_name)

    if check_if_file_exist(file_name):
        return "Nie utworzono pliku"

    ws = wb.active
    ws.title = "Tabliczka mnozenia"

    for row in range(1, 11):
        for column in range(1, 11):
            ws.cell(row=row, column=column).value = row * column

    ws = wb.create_sheet("pi")
    ws["A1"] = 3.14

    ws = wb.create_sheet("e")
    ws["A1"] = 2.718

    ws = wb.create_sheet("losowe liczby")

    for i in range(1, 100):
        ws["A{}".format(i)] = i
        ws["B{}".format(i)] = round(random.random() * 100000) / 1000

    wb.save(file_name)
```

Funkcja create_empty_file

Funkcja tworzy pusty plik i tak jak wcześniej sprawdza czy plik istnieje oraz sprawdza czy podano rozszerzenie.

```
def create_empty_file():
    file_name = input("Podaj nazwe pliku: ")
    if not re.search(reg_file_name_xlsx, file_name):
        file_name = "{}.xlsx".format(file_name)

    if check_if_file_exist(file_name):
        return "Nie utworzono pliku"

    wb = Workbook()
    wb.save(file_name)
```

Funkcja copy_file część 1

Tak jak w poprzednich funkcjach:

Pobieram nazwy plików.

Inicjuje zmienne do późniejszej edycji.

Sprawdzam podane rozszerzenia plików, jak nie podane, to poprawiam.

Sprawdzam czy podany plik istnieje.

```
def copy_file():
    # pobranie od uzytkownika nazw plikow
    file_name_src = input("Wpisz nazwe pliku (lokalizacje) do skopiowania: ")
    file_name_dst = input("Wpisz nazwe pliku (lokalizacje) gdzie skopiowac: ")

    # zainicjowanie zmiennych
    wb1 = Workbook()
    wb2 = Workbook()

    # poprawienie rozszerzen plikow
    if not re.search(reg_file_name_xlsx, file_name_src):
        file_name_src = "{}.xlsx".format(file_name_src)

    if not re.search(reg_file_name_xlsx, file_name_dst):
        file_name_dst = "{}.xlsx".format(file_name_dst)

    # sprawdzenie czy plik do zapisania istnieje
    if check_if_file_exist(file_name_dst):
        return "Nie utworzono pliku"
```

Funkcja copy_file część 2

Funkcja load_workbook może wywołać błędy, kiedy nie ma podanej lokalizacji pliku, gdy jest błędne rozszerzenie oraz gdy nie mamy uprawnień do otwarcia pliku (kiedy otwarty jest w programie Excel).

Następnie iterując po arkuszach i komórkach w nich zawartych, kopiuje z pierwszego pliku do drugiego wartości.

Następnie przepisuje tytuły arkuszy oraz zapisuje nowo utworzony plik.

Na końcu zamykam pliki (gdybym nie zainicjował wstępnie zmiennych, a błąd wyskoczyłby przed zainicjowaniem, to te metody również wywołały by błąd).

```
try:
    wb1 = load_workbook(file_name_src)  # zaladowanie arkusza
    ws2 = wb2.active  # ustawienie aktywnego arkusza
    ws2.title = wb1.worksheets[0].title  # przepisanie tytulu pierwszego arkusza

    for n in range(len(wb1.worksheets)):
        ws1 = wb1.worksheets[n]  # ustawienie aktywnego arkusza (do skopiowania) na kolejny
        # kopiowanie wartosci komorek
        for i in range(ws1.min_row, ws1.max_row + 1):
            for j in range(ws1.min_column, ws1.max_column + 1):
                cell = ws1.cell(row=i, column=j)

                ws2.cell(row=i, column=j).value = cell.value
        # ustawianie tytulow pozostalych arkuszy
        if n + 1 < len(wb1.worksheets):
            ws2 = wb2.create_sheet(wb1.worksheets[n + 1].title)

    wb2.save(str(file_name_dst))
    print("PLIK SKOPIOWANO")
except InvalidFileException:
    print("Nieobslugiwany format pliku!")
except PermissionError as e:
    print("Brak uprawnien do pliku!", e.filename)
except FileNotFoundError as e:
    print("Nie znaleziono lokalizacji!", e.filename)
finally:
    wb1.close()
    wb2.close()
```

Funkcja search

Tak jak poprzednio, inicjujemy, pobieramy potrzebne rzeczy od użytkownika (nazwe pliku, wyrażenie regularne).

Następnie wczytujemy plik z opcją read_only (poprawia nieco wydajność) oraz wczytujemy arkusz.

Później pobieramy z arkusza listę z wartościami pasującymi do wyrażenia oraz wypisujemy to w konsoli.

Jeśli znaleziono pasujące wyniki, to pytamy użytkownika o to czy zapisać wynik do pliku.

Na końcu standardowo przechwytujemy błędy oraz zamykamy plik.

```
def search():
    file_name = input("Wpisz nazwe pliku (lokalizacje) do przeszukania: ")
    if not re.search(reg_file_name_xlsx, file_name):
        file_name = "{}.xlsx".format(file_name)

    wb = Workbook()
    try:
        wb = load_workbook(file_name, read_only=True)
        search_query = create_reg_exp_query()

        # pobieranie nazwy arkusza do przeszukiwania
        ws = get_sheet(wb, search_query)

        # pobranie danych z arkusza pasujace do query
        result_arr = get_data(ws, search_query)

        # wyswietlenie tabeli w konsoli
        print_table(result_arr)

        # zapisywanie wynikow wyszukiwania do pliku
        if len(result_arr) > 0:
            choice = input("Czy zapisac wynik wyszukiwania? (t - tak): ")
            if re.search(confirm_query, choice):
                save_to_json({
                    ws.title: result_arr
                })
    except InvalidFileException:
        print("Nieobslugiwany format pliku!")
    except PermissionError as e:
        print("Brak uprawnien do pliku!", e.filename)
    except FileNotFoundError as e:
        print("Nie znaleziono lokalizacji!", e.filename)
    finally:
        wb.close()
```

Funkcja save_xlsx_to_json

Funkcja pobiera od użytkownika nazwę pliku, wstępnie inicjalizujemy zmienną przechowującą plik, oraz zmienną przechowującą zawartość pliku xlsx (w typie słownikowym).

Funkcja iteruje po arkuszach pobierając dane z nich, następnie wywołuje funkcje save_to_json, która zapisuje wynik do pliku.

```
def save_xlsx_to_json():
    file_name = input(
        "Wpisz nazwe pliku " +
        "(lokalizacje pliku z rozszerzeniem xlsx) " +
        "do eksportu (do pliku json): ")
    result_dic = {}
    wb = Workbook()
    try:
        if not re.search(reg_file_name_xlsx, file_name):
            file_name = "{}.xlsx".format(file_name)
        wb = load_workbook(file_name, read_only=True)

        for ws in wb.worksheets:
            result_dic[ws.title] = get_data(ws)

        save_to_json(result_dic)
    except InvalidFileException:
        print("Nieobslugiwany format pliku!")
    except PermissionError as e:
        print("Brak uprawnien do pliku!", e.filename)
    except FileNotFoundError as e:
        print("Nie znaleziono lokalizacji!", e.filename)
    finally:
        wb.close()
```

Funkcja save_json_to_xlsx

Funkcja pobiera od użytkownika niezbędne rzeczy, jak poprzednio.

Jedyna nowa funkcja, która jest tu wykorzystywana, to json.load(), która wczytuje zawartość pliku w formacie json, który jest również konwertowany do odpowiadającego mu typowi danych w pythonie (np. słownik).

```
def save_json_to_xlsx():
    file_name = input(
        "Wpisz nazwe pliku " +
        "(lokalizacje pliku z rozszerzeniem json) " +
        "do eksportu (do pliku xlsx): ")
    if not re.search(reg_file_name_json, file_name):
        file_name = "{}.json".format(file_name)

    try:
        with open("{}".format(file_name), "r") as json_file:
            tables = json.load(json_file)
            wb = Workbook()

            file_name_out = input("Wpisz nazwe pliku gdzie zapisac plik: ")
            if not re.search(reg_file_name_xlsx, file_name_out):
                file_name_out = "{}.xlsx".format(file_name_out)

            overwrite_first_sheet = False
            if check_if_file_exist(file_name_out, True):
                wb = load_workbook(file_name_out)
            else:
                overwrite_first_sheet = True

            if type(tables) == list:
                add_sheet_to_workbook(tables, wb, overwrite=overwrite_first_sheet)
            elif type(tables) == dict:
                for table_keys in tables.keys():
                    add_sheet_to_workbook(tables[table_keys], wb, table_keys, overwrite_first_sheet)
                    overwrite_first_sheet = False
            del overwrite_first_sheet

            wb.save(file_name_out)
    except InvalidFileException:
        print("Nieobslugiwany format pliku!")
    except PermissionError as e:
        print("Brak uprawnien do pliku!", e.filename)
    except FileNotFoundError as e:
        print("Nie znaleziono lokalizacji!", e.filename)
```

Dziękuję za uwagę