

Welcome to Sandwich shop 710069439's documentation!

Indices and tables

- [Index](#)
- [Module Index](#)
- [Search Page](#)

Download and setup instructions

Installing and running sandwich_shop app:

- Download zip file
- Extract all files into an accesible file location
- On your command line interface run:

```
python -m pip install -e C:/Users/YOUR_DOWNLOAD_LOCATION/sandwich_shop
```

- To launch the app run:

```
python C:/Users/YOUR_DOWNLOAD_LOCATION/sandwich_shop/code/main.py
```

Running the app from within the package:

- Open the sandwich_shop folder in an IDE such as Visual Studio Code from the root directory 'sandwich_shop'
- Run the terminal and check that the file path ends with 'sandwich_shop'
- Install the package by running the command `python -m pip install -e .`
- To launch the app run:

```
python ../sandwich_shop/code/main.py
```

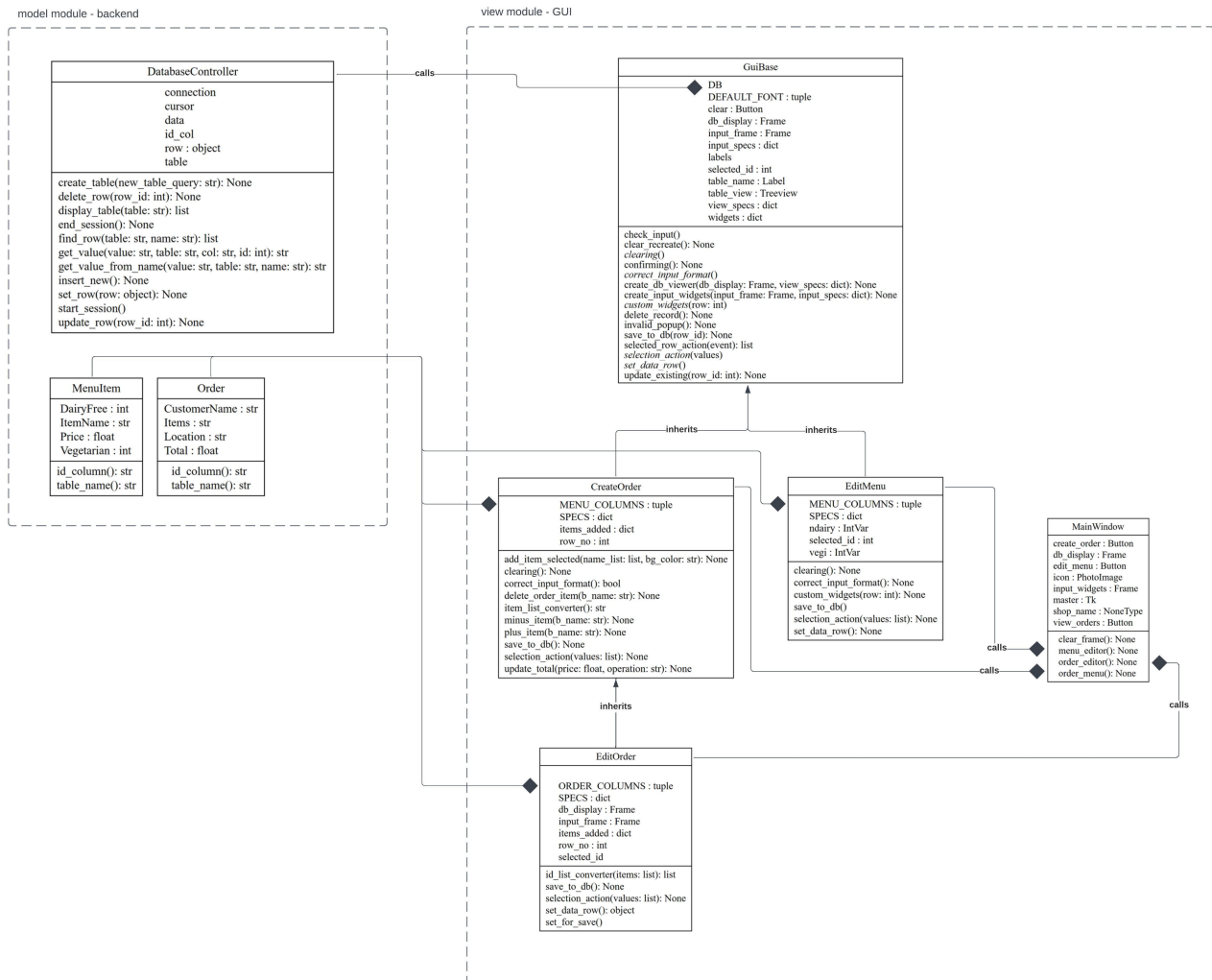
- OR go to sub-directory 'code', go into the module main.py, run the file
- OR in a python file, add the following lines and run the file:

```

from sandwich_shop.code import main
main.run()

```

UML Class Diagram



Python Module Index

[c](#) | [m](#) | [v](#)

C

`code`

m

 `model`

`model.data`

`model.db_controller`

`model.utils`

V

 `view`

`view.create_order`

`view.edit_menu`

`view.edit_order`

`view.gui_base`

`view.main_menu`

code

code

- [code package](#)
 - [Subpackages](#)
 - [code.model package](#)
 - [Submodules](#)
 - [code.model.data module](#)
 - [code.model.db_controller module](#)
 - [code.model.utils module](#)
 - [Module contents](#)
 - [code.view package](#)
 - [Submodules](#)
 - [code.view.create_order module](#)
 - [code.view.edit_menu module](#)
 - [code.view.edit_order module](#)
 - [code.view.gui_base module](#)
 - [code.view.main_menu module](#)
 - [Module contents](#)
 - [Submodules](#)
 - [code.main module](#)
 - [Module contents](#)

code package

Subpackages

- [code.model package](#)

- [Submodules](#)

- [code.model.data module](#)

- `MenuItem`

- `MenuItem.DairyFree`

- `MenuItem.ItemName`

- `MenuItem.Price`

- `MenuItem.Vegetarian`

- `MenuItem.id_column()`

- `MenuItem.table_name()`

- `Order`

- `Order.CustomerName`

- `Order.Items`

- `Order.Location`

- `Order.Total`

- `Order.id_column()`

- `Order.table_name()`

- `validate_types()`

- [code.model.db_controller module](#)

- `DatabaseController`

- `DatabaseController.create_table()`

- `DatabaseController.delete_row()`

- `DatabaseController.display_table()`

- `DatabaseController.end_session()`

- `DatabaseController.find_row()`

- `DatabaseController.get_value()`

- `DatabaseController.get_value_from_name()`

- `DatabaseController.insert_new()`

- `DatabaseController.set_row()`
- `DatabaseController.start_session()`
- `DatabaseController.update_row()`

- `code.model.utils` module

- `location_converter()`
- `what3words_converter()`

- Module contents

- `code.view` package

- Submodules

- `code.view.create_order` module

- `CreateOrder`
 - `CreateOrder.MENU_COLUMNS`
 - `CreateOrder.SPECS`
 - `CreateOrder.add_item_selected()`
 - `CreateOrder.clearing()`
 - `CreateOrder.correct_input_format()`
 - `CreateOrder.delete_order_item()`
 - `CreateOrder.item_list_converter()`
 - `CreateOrder.minus_item()`
 - `CreateOrder.plus_item()`
 - `CreateOrder.save_to_db()`
 - `CreateOrder.selection_action()`
 - `CreateOrder.update_total()`

- `code.view.edit_menu` module

- `EditMenu`
 - `EditMenu.MENU_COLUMNS`
 - `EditMenu.SPECS`
 - `EditMenu.clearing()`
 - `EditMenu.correct_input_format()`
 - `EditMenu.custom_widgets()`
 - `EditMenu.save_to_db()`
 - `EditMenu.selection_action()`
 - `EditMenu.set_data_row()`

- `code.view.edit_order` module

- `EditOrder`
 - `EditOrder.ORDER_COLUMNS`

- `EditOrder.SPECS`
- `EditOrder.id_list_converter()`
- `EditOrder.save_to_db()`
- `EditOrder.selection_action()`
- `EditOrder.set_data_row()`
- `EditOrder.set_for_save()`

- `code.view.gui_base` module

- `GuiBase`
 - `GuiBase.DEFAULT_FONT`
 - `GuiBase.check_input()`
 - `GuiBase.clear_recreate()`
 - `GuiBase.clearing()`
 - `GuiBase.confirming()`
 - `GuiBase.correct_input_format()`
 - `GuiBase.create_db_viewer()`
 - `GuiBase.create_input_widgets()`
 - `GuiBase.custom_widgets()`
 - `GuiBase.delete_record()`
 - `GuiBase.invalid_popup()`
 - `GuiBase.save_to_db()`
 - `GuiBase.selected_row_action()`
 - `GuiBase.selection_action()`
 - `GuiBase.set_data_row()`
 - `GuiBase.update_existing()`

- `code.view.main_menu` module

- `MainWindow`
 - `MainWindow.clear_frame()`
 - `MainWindow.menu_editor()`
 - `MainWindow.order_editor()`
 - `MainWindow.order_menu()`

- `Module contents`

Submodules

`code.main` module

code.model package

Submodules

code.model.data module

Defines dataclasses in line with the database setup

Sources:

<https://docs.python.org/3/library/dataclasses.html>

<https://www.youtube.com/watch?v=CvQ7e6yUtnw>

```
class model.data.MenuItem(ItemName: str, Price: float, Vegetarian: int = 0, DairyFree: int = 0)
```

Bases: `object`

Creates dataclass instance containing all data for a row in the Menu table in database

- Parameters:**
- **ItemName** (*str*) – name of sandwich
 - **Price** (*float*) – price in £
 - **Vegetarian** (*int*) – binary value indicator, 1 means yes. Defaults to 0
 - **DairyFree** (*int*) – binary value indicator, 1 means yes. Defaults to 0

Returns: MenuItem(ItemName="", Price=0., Vegetarian=0, DairyFree=0)

DairyFree: *int* = 0

ItemName: *str*

Price: *float*

Vegetarian: *int* = 0

static id_column() → *str*

Gets name of ID column in database

Returns: ID column name

Return type: *str*

static table_name()→ str

Gets name of associate table in the database

Returns: table name

Return type: str

class model.data.Order(*CustomerName: str, Location: str, Items: str, Total: float*)

Bases: `object`

Creates dataclass instance containing all data for a row in Orders table in the database

Parameters:

- **CustomerName** (*str*) –
- **Location** (*str*) – what3words address
- **Items** (*str*) – string of list of item IDs as found in menu table
- **Total** (*float*) – total price paid at time of order

Returns: Order(CustomerName=", Location=", Items=[], Total=0.)

CustomerName: *str*

Items: *str*

Location: *str*

Total: *float*

static id_column()→ str

Gets name of ID column in database

Returns: ID column name

Return type: str

static table_name()→ str

Gets name of associate table in the database

Returns: table name

Return type: str

model.data.validate_types(*obj: object*)→ None

Gets type hints defined in dataclasses to confirm input

Parameters: **obj** (*dataclass*) – MenuItem or Order

Raises: **TypeError** – if incorrect data type input, else None

code.model.db_controller module

All database manipulation functions

Sources:

<https://docs.python.org/3/library/sqlite3.html>

https://www.tutorialspoint.com/sqlite/sqlite_python.htm

<https://www.geeksforgeeks.org/python-sqlite/>

class `model.db_controller.DatabaseController`(*row: object | None = None*)

Bases: `object`

create_table(*new_table_query: str*) → None

Create new sqlite table

Parameters: **new_table_query** (*str*) – SQLite query for creating new table

delete_row(*row_id: int*) → None

Delete row from table where Name column matches

Parameters: **row_id** (*int*) – ID number of row to be updated

display_table(*table: str | None = None*) → list

Returns all rows in a table based on input or table associated with set data row

end_session() → None

Commit changes and close connection

find_row(*table: str, name: str*) → list

Finds row of data in a given table based on Name column

Parameters:

- **table** (*str*) – name of table to query from
- **name** (*str*) – name of item to find in the Name column of the table

Returns: list of tuples containing data in each row

Return type: list

get_value(*value: str, table: str, col: str, id: int*) → str

Finds data in row in given table by ID of the item

Parameters:

- **value** (*str*) – column name to get value from
- **table** (*str*) – table name
- **col** (*str*) – name of ID column to use for search
- **id** (*str*) – ID number of the row

Returns: value from the specified column of the found row

Return type: str

get_value_from_name(*value: str, table: str, name: str*)→ str

Finds row of data in given table by item name

Parameters:

- **value** (*str*) – column name to get value from
- **table** (*str*) – table name
- **name** (*str*) – name of item

Returns: value from the specified column of the found row

Return type: str|int

insert_new()→ None

Parses data dictionary into new row in the relevant table

set_row(*row: object*)→ None

Takes instance of a dataclass object, parses into dictionary, gets associated table name

Parameters: row (*object*) – dataclass object MenuItem or Order

start_session()

update_row(*row_id: int*)→ None

Updates the row with given id number to data in set data row

Parameters: row_id (*int*) – ID number of row to be updated

code.model.utils module

Module for external what3words api call functions

<https://developer.what3words.com/public-api/docs>

model.utils.location_converter(*lat_long: str*)→ str

Calls api to convert latitude, longitude to what3words

Parameters: lat_long (*str*) – two float numbers for coordinates

Returns: what3words

Return type: str

model.utils.what3words_converter(*words: str*)→ str

Calls api to convert what3words to coordinates

Parameters: **words** (*str*) – what3words string

Returns: latitude, logitude converted to string

Return type: *str*

Module contents

code.view package

Submodules

code.view.create_order module

All input widgets and functionality for creating order on the UI

```
class view.create_order.CreateOrder(db_display: Frame, input_frame: Frame)
```

Bases: `GuiBase`

```
MENU_COLUMNS= ('Item Name', 'Price(£)', 'Vegetarian', 'Dairy Free')
```

SPECS

```
= {'input_widgets': {'buttons': ['Confirm order'], 'input': {'CustomerName': {'label': 'Customer name:',  
'type': 'entry'}, 'Items': {'label': 'Items in order:', 'type': 'frame'}, 'Location': {'label': 'Location (lat,long):',  
'type': 'entry'}, 'Total': {'label': 'Total £ ', 'type': 'label'}}}, 'view': {'table_name': 'Select items from the  
menu', 'table_view': {'columns': ('Item Name', 'Price(£)', 'Vegetarian', 'Dairy Free'), 'table': 'menu'}}
```

```
add_item_selected(name_list: list, bg_color: str = 'white')→ None
```

Convert data from selected row into items listed in input widgets

- Parameters:**
- **name_list** (*list*) – data points containing item name, quantity, price
 - **bg_color** (*str, optional*) – background color of component. Defaults to “white”.

```
clearing()→ None
```

Set all input widgets to empty values

```
correct_input_format()→ bool
```

Checks format of input values

- Returns:** True if all conditions are correct, else False
- Return type:** bool

```
delete_order_item(b_name: str)→ None
```

Destroys row of labels for a given item name

Parameters: **b_name** (*str*) – Item name

item_list_converter()→ *str*

Converts list of names of sandwiches added to order to list of associated ids

Returns: string of item ids list

Return type: *str*

minus_item(b_name: str)→ *None*

Adds the minus button to decrease quantity of item added

Parameters: **b_name** (*str*) – Item name in selected row from table

plus_item(b_name: str)→ *None*

Adds the plus button to increase quantity of item added

Parameters: **b_name** (*str*) – Item name in selected row from table

save_to_db()→ *None*

Convert inputs to dataclass object and insert into database

selection_action(values: list)→ *None*

Maps values from selected row to items added widgets

Parameters: **values** (*list*) – row of selected menu item data

update_total(price: float, operation: str = '+')→ *None*

Adds or subtracts from total based on plus or minus button click

- Parameters:**
- **price** (*float*) – unit price of a menu item
 - **operation** (*str, optional*) –
 - or - operation selection.
 - **"+".** (*Defaults to*) –

code.view.edit_menu module

User interface and functionality for editing Menu saved in the database

class **view.edit_menu.EditMenu**(*db_display: Frame, input_frame: Frame*)

Bases: GuiBase

MENU_COLUMNS= ('Item Name', 'Price(£)', 'Vegetarian', 'Dairy Free')

SPECS

```
= {'input_widgets': {'buttons': ['Add/Update', 'Delete record'], 'input': {'DairyFree': {'label': 'DairyFree:', 'type': 'checkbox'}, 'ItemName': {'label': 'Item Name:', 'type': 'entry'}, 'Price': {'label': 'Price £:', 'type': 'entry'}, 'Vegetarian': {'label': 'Vegetarian:', 'type': 'checkbox'}}}, 'view': {'table_name': 'Select menu item to edit', 'table_view': {'columns': ('Item Name', 'Price(£)', 'Vegetarian', 'Dairy Free'), 'table': 'menu'}}
```

clearing()→ None

Sets input widgets to empty values

correct_input_format()→ None

Checks correct format of input values

Returns: True if all formats are correct, else False

Return type: bool

custom_widgets(row: int)→ None

Adds custom widgets to those defined in base class GuiBase

Parameters: row (int) – row number for placement in UI grid

save_to_db()

Updates data or saves to database, updates database viewer

Parameters: row_id (int) – id number of row in the database. Defaults to 0.

selection_action(values: list)→ None

Parses selected row in table view to populate in input widgets

Parameters: values (list) – values from the selected row

set_data_row()→ None

Turns input data into dataclass if datatypes are as expected

Returns: dataclass from input values

Return type: MenuItem

code.view.edit_order module

Widgets and functionality for editing recorded orders

```
class view.edit_order.EditOrder(db_display: Frame, input_frame: Frame)
```

Bases: CreateOrder

```
ORDER_COLUMNS= ('Customer Name', 'Location', 'Items', 'Total(£)')
```

SPECS

```
= {'input_widgets': {'buttons': ['Update', 'Delete record'], 'input': {'CustomerName': {'label': 'Customer name:', 'type': 'entry'}, 'Items': {'label': 'Items in order:', 'type': 'frame'}, 'Location': {'label': 'Location (lat,long):', 'type': 'entry'}, 'Total': {'label': 'Total £ (when ordered):', 'type': 'label'}}}, 'view': {'table_name': 'Select order to edit', 'table_view': {'columns': ('Customer Name', 'Location', 'Items', 'Total(£)'), 'table': 'orders'}}
```

id_list_converter(items: list) → list

Converts list of item ids to names for display

Parameters: items (list) – ids of items in the order

Returns: names of ordered items

Return type: list

save_to_db() → None

Updates data or saves to database, updates database viewer

selection_action(values: list) → None

Maps values to widgets, re-creates ordered items list

Parameters: values (list) – row of selected data

set_data_row() → object

Maps data in input widgets to Order dataclass

Returns: Order dataclass

Return type: object

set_for_save()

code.view.gui_base module

Base class for view, edit, create user interfaces and functionalities

```
class view.gui_base.GuiBase(db_display: Frame, input_frame: Frame, specs: dict)
```

Bases: `object`

DEFAULT_FONT= ('OpenSans', 12)

check_input()

Checks correct input format as implemented by each child class

Returns: True if all input values are in expected format and datatype, else False

Return type: bool|None

clear_recreate()→ None

Destroys database viewer and recreates it so that any changes are reflected

abstract clearing()

Clears all input widgets

confirming()→ None

Saves to database, creates pop-up message confirming completion for correctly formatted input, clears all input widgets.

abstract correct_input_format()

Child class specific input checker

create_db_viewer(db_display: Frame, view_specs: dict)→ None

Creates Treeview table for viewing database tables

- Parameters:**
- **db_display** (*Frame*) –
 - **view_specs** (*dict*) – SAMPLE_SPECS[“view”] view related configurations

create_input_widgets(input_frame: Frame, input_specs: dict)→ None

Creates all input widgets

- Parameters:**
- **input_frame** (*Frame*) –
 - **input_specs** (*dict*) – SAMPLE_SPECS[“input_widgets”] widget related configurations

custom_widgets(row: int)

Additional widgets for child classes to specify

- Parameters:**
- **row** (*int*) – row number to place custom widgets on the UI

delete_record()→ None

Deletes record from the database

invalid_popup()→ None

Creates pop-up message box indicating invalid input

save_to_db(row_id=0)→ None

Updates data or saves to database, updates database viewer

- Parameters:**
- **row_id** (*int*) – id number of row in the database. Defaults to 0.

selected_row_action(event)→ list

Gets values of the selected row in Treeview, processes custom to each child class

Parameters: **event** (*click*) – listens for click on any row

Returns: values of data in the selected row

Return type: list

abstract selection_action(values)

Parses and operates on data from selected row in table view

Parameters: **values** (*list|tuple*) – values from the selected row

set_data_row()

Gets data from input widgets into dataclass

update_existing(row_id: int)→ None

Updates existing record in the database by id

Parameters: **row_id** (*int*) – id number of row in the database

code.view.main_menu module

Creates app window and the initial view, initiates relevant interfaces

class view.main_menu.MainWindow(*master: Tk*)

Bases: object

clear_frame()→ None

Delete all elements in editor interface frames

menu_editor()→ None

Clears frames, recreates interface for editing the menu

order_editor()→ None

Clears frames, recreates interface for editing orders

order_menu()→ None

Clears frames, recreates interface for creating order

Module contents

Package containing all Tkinter GUI modules

Sources for all modules:

https://www.youtube.com/watch?v=yQSEXcf6s2I&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV&ab_channel=Codemy.com

noqa E501

<https://github.com/flatplanet/Intro-To-TKinter-Youtube-Course>

<https://www.geeksforgeeks.org/python-gui-tkinter/>

<https://realpython.com/python-gui-tkinter/>

Index

[A](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [I](#) | [L](#) | [M](#) | [O](#) | [P](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#)

A

[add_item_selected\(\)](#) ([view.create_order.CreateOrder](#) method)

C

[check_input\(\)](#) ([view.gui_base.GuiBase](#) method)

[clear_frame\(\)](#)

([view.main_menu.MainWindow](#) method)

[clear_recreate\(\)](#) ([view.gui_base.GuiBase](#) method)

[clearing\(\)](#) ([view.create_order.CreateOrder](#) method)

([view.edit_menu.EditMenu](#) method)

([view.gui_base.GuiBase](#) method)

[code](#)

[module](#)

[compile_command\(\)](#) (in [module code](#))

[confirming\(\)](#) ([view.gui_base.GuiBase](#) method)

[correct_input_format\(\)](#)

([view.create_order.CreateOrder](#) method)

([view.edit_menu.EditMenu](#) method)

([view.gui_base.GuiBase](#) method)

[create_db_viewer\(\)](#) ([view.gui_base.GuiBase](#) method)

[create_input_widgets\(\)](#) ([view.gui_base.GuiBase](#) method)

[create_table\(\)](#)

([model.db_controller.DatabaseController](#) method)

[CreateOrder](#) (class in [view.create_order](#))

[custom_widgets\(\)](#) ([view.edit_menu.EditMenu](#) method)

([view.gui_base.GuiBase](#) method)

[CustomerName](#) ([model.data.Order](#) attribute)

D

[DairyFree](#) ([model.data.MenuItem](#) attribute)

[DatabaseController](#) (class in [model.db_controller](#))

[DEFAULT_FONT](#) ([view.gui_base.GuiBase](#) attribute)

[delete_order_item\(\)](#)

([view.create_order.CreateOrder](#) method)

[delete_record\(\)](#) ([view.gui_base.GuiBase](#) method)

[delete_row\(\)](#)

([model.db_controller.DatabaseController](#) method)

[display_table\(\)](#)

([model.db_controller.DatabaseController](#) method)

E

[EditMenu](#) (class in [view.edit_menu](#))

[EditOrder](#) (class in [view.edit_order](#))

[end_session\(\)](#)

([model.db_controller.DatabaseController](#) method)

F

[find_row\(\)](#) ([model.db_controller.DatabaseController](#) method)

G

`get_value()`
(`model.db_controller.DatabaseController`
method)

`get_value_from_name()`
(`model.db_controller.DatabaseController`
method)
`GuiBase` (class in `view.gui_base`)

I

`id_column()` (`model.data.MenuItem` static method)
(`model.data.Order` static method)
`id_list_converter()` (`view.edit_order.EditOrder`
method)
`insert_new()`
(`model.db_controller.DatabaseController` method)
`interact()` (`code.InteractiveConsole` method)
(in module `code`)

`InteractiveConsole` (class in `code`)
`InteractiveInterpreter` (class in `code`)
`invalid_popup()` (`view.gui_base.GuiBase`
method)
`item_list_converter()`
(`view.create_order.CreateOrder` method)
`ItemName` (`model.data.MenuItem`
attribute)
`Items` (`model.data.Order` attribute)

L

`Location` (`model.data.Order` attribute)

`location_converter()` (in module `model.utils`)

M

`MainWindow` (class in `view.main_menu`)
`MENU_COLUMNS`
(`view.create_order.CreateOrder` attribute)
(`view.edit_menu.EditMenu` attribute)
`menu_editor()` (`view.main_menu.MainWindow`
method)
`MenuItem` (class in `model.data`)
`minus_item()` (`view.create_order.CreateOrder`
method)
`model`
 `module`
`model.data`
 `module`
`model.db_controller`
 `module`
`model.utils`
 `module`

`module`
 `code`
 `model`
 `model.data`
 `model.db_controller`
 `model.utils`
 `view`
 `view.create_order`
 `view.edit_menu`
 `view.edit_order`
 `view.gui_base`
 `view.main_menu`

O

`Order` (class in `model.data`)
`ORDER_COLUMNS`
(`view.edit_order.EditOrder` attribute)

`order_editor()` (`view.main_menu.MainWindow`
method)
`order_menu()` (`view.main_menu.MainWindow`
method)

P

`plus_item()` (`view.create_order.CreateOrder`
method)

`Price` (`model.data.MenuItem` attribute)
`push()` (`code.InteractiveConsole` method)

R

<code>raw_input()</code> (<code>code.InteractiveConsole</code> method)	<code>runcode()</code> (<code>code.InteractiveInterpreter</code> method)
<code>resetbuffer()</code> (<code>code.InteractiveConsole</code> method)	<code>runsource()</code> (<code>code.InteractiveInterpreter</code> method)

S

<code>save_to_db()</code> (<code>view.create_order.CreateOrder</code> method)	<code>set_for_save()</code> (<code>view.edit_order.EditOrder</code> method)
(<code>view.edit_menu.EditMenu</code> method)	<code>set_row()</code> (<code>model.db_controller.DatabaseController</code> method)
(<code>view.edit_order.EditOrder</code> method)	<code>showsyntaxerror()</code> (<code>code.InteractiveInterpreter</code> method)
(<code>view.gui_base.GuiBase</code> method)	<code>showtraceback()</code> (<code>code.InteractiveInterpreter</code> method)
<code>selected_row_action()</code> (<code>view.gui_base.GuiBase</code> method)	<code>SPECS</code> (<code>view.create_order.CreateOrder</code> attribute)
<code>selection_action()</code> (<code>view.create_order.CreateOrder</code> method)	(<code>view.edit_menu.EditMenu</code> attribute)
(<code>view.edit_menu.EditMenu</code> method)	(<code>view.edit_order.EditOrder</code> attribute)
(<code>view.edit_order.EditOrder</code> method)	<code>start_session()</code> (<code>model.db_controller.DatabaseController</code> method)
(<code>view.gui_base.GuiBase</code> method)	
<code>set_data_row()</code> (<code>view.edit_menu.EditMenu</code> method)	
(<code>view.edit_order.EditOrder</code> method)	
(<code>view.gui_base.GuiBase</code> method)	

T

<code>table_name()</code> (<code>model.data.MenuItem</code> static method)	<code>Total</code> (<code>model.data.Order</code> attribute)
(<code>model.data.Order</code> static method)	

U

<code>update_existing()</code> (<code>view.gui_base.GuiBase</code> method)	<code>update_row()</code> (<code>model.db_controller.DatabaseController</code> method)
	<code>update_total()</code> (<code>view.create_order.CreateOrder</code> method)

V

<code>validate_types()</code> (in module <code>model.data</code>)	<code>view.edit_order</code> module
<code>Vegetarian</code> (<code>model.data.MenuItem</code> attribute)	<code>view.gui_base</code> module
<code>view</code> module	<code>view.main_menu</code> module
<code>view.create_order</code> module	
<code>view.edit_menu</code> module	

W

<code>what3words_converter()</code> (in module <code>model.utils</code>)	<code>write()</code> (<code>code.InteractiveInterpreter</code> method)
---	---