# interrogatio Documentation

***Release 1.0.0***

**Francesco Faraone**

**Dec 07, 2022**

# Contents:

A python library to prompt users for inputs in a terminal application.

CHAPTER 1

# What is interrogatio

interrogatio is a python 3.8+ library based on the python-prompt-toolkit and inspired by PyInquirer that help CLI developers to ask users for inputs.

Questions can be rendered onto the terminal prompt or as curses-like dialogs.

## 1.1 Getting started

### 1.1.1 Requirements

`interrogatio` depends on the python-prompt-toolkit library and its dependencies.

### 1.1.2 Installation

#### Using pip

```
$ pip install interrogatio
```

#### Extra dependencies

If you want to use the shell command with yml files you can install the yml dependency:

```
$ pip install interrogatio[yml]
```

### 1.1.3 Basic usage

`interrogatio` needs a list of questions to prompt the user for answers.

Each question is a python dictionary with at least the following keys:

- **name**: it has to be unique within the list of questions. It represents the variable name;
- **type**: the type of question;
- **message**: the text of the prompt.

Optionally you should specify:

- a **default**: a default value;
- a **validators**: a list of children of Validator class
- a **question_mark**: you can customize the question mark symbol.
- a **values**: a list of tuples (value, label) to provide a list of choices for the `selectone` or `selectmany` question types.

Each type of question is managed by a `interrogatio.handlers.QHandler`.

See the *Question handlers* section for more information.

### 1.1.4 Run interrogatio

#### Prompt mode

```
from interrogatio import interrogatio

questions = [
    {
        'name': 'name',
        'type': 'input',
        'message': "What's your name ?",
        'description': 'Please enter your full name. This field is required.',
        'validators': [{'name': 'required'}],
    },
    {
        'name': 'birth_date',
        'type': 'date',
        'message': "What's your birth date ?",
        'description': 'Enter your birth date.',
    },
    {
        'name': 'nationality',
        'type': 'selectone',
        'message': "What's your nationality ?",
        'description': 'Please choose one from the list.',
        'validators': [{'name': 'required'}],
        'values': [
            ('IT', 'Italian'),
            ('ES', 'Spanish'),
            ('US', 'American'),
            ('UK', 'English'),
        ],
    },
    {
        'name': 'languages',
        'type': 'selectmany',
```

(continues on next page)

(continued from previous page)

```
        'message': "What are your favorite programming languages ?",
        'description': 'Please choose your favorites from the list.',
        'values': [
            ('py', 'Python'),
            ('rb', 'Ruby'),
            ('js', 'Javascript'),
            ('go', 'Golang'),
            ('rs', 'Rust'),
            ('c', 'C'),
            ('cpp', 'C++'),
            ('java', 'Java'),
        ],
    },
]


answers = interrogatio(questions)
```

### Dialog mode

```
from interrogatio import dialogus

questions = [
    {
        'name': 'name',
        'type': 'input',
        'message': "What's your name ?",
        'description': 'Please enter your full name. This field is required.',
        'validators': [{'name': 'required'}],
    },
    {
        'name': 'birth_date',
        'type': 'date',
        'message': "What's your birth date ?",
        'description': 'Enter your birth date.',
    },
    {
        'name': 'nationality',
        'type': 'selectone',
        'message': "What's your nationality ?",
        'description': 'Please choose one from the list.',
        'validators': [{'name': 'required'}],
        'values': [
            ('IT', 'Italian'),
            ('ES', 'Spanish'),
            ('US', 'American'),
            ('UK', 'English'),
        ],
    },
    {
        'name': 'languages',
        'type': 'selectmany',
        'message': "What are your favorite programming languages ?",
```

(continues on next page)

(continued from previous page)

```
        'description': 'Please choose your favorites from the list.',
        'values': [
            ('py', 'Python'),
            ('rb', 'Ruby'),
            ('js', 'Javascript'),
            ('go', 'Golang'),
            ('rs', 'Rust'),
            ('c', 'C'),
            ('cpp', 'C++'),
            ('java', 'Java'),
        ],
    },
]

intro = """<blue>Welcome to <b><i>interrogatio 2.0</i></b>!

This is the second major release of interrogatio with nice improvements.</blue>

<b>What's new</b>
<b>----------</b>

* Curses-like dialog experience had been completely rewritten.
* New questions handlers for dates, date ranges and masked inputs.
* Validators are now based on the <u>validators</u> library.
"""


answers = dialogus(questions, 'interrogatio showcase', intro=intro, summary=True)
```

You can customize the dialog title and the confirm and cancel buttons text.

### 1.1.5 Validation

You could specify a list of validators for each question:

```
from interrogatio import dialogus
from interrogatio.validators import RequiredValidator, MinLengthValidator

questions = [
    {
        'name': 'username',
        'type': 'input',
        'message': 'Enter your username',
        'validators': [RequiredValidator()]

    },
    {
        'name': 'password',
        'type': 'password',
        'message': 'Enter your password',
        'validators': [MinLengthValidator(8)]
    }  ,
]
answers = dialogus(
    questions,
```

(continues on next page)

(continued from previous page)

```
    'Please enter your credential',
    finish='login',
    cancel='cancel',
)
```

Validators can also be expressed using aliases:

```
from interrogatio import dialogus
from interrogatio.validators import RequiredValidator, MinLengthValidator

questions = [
    {
        'name': 'username',
        'type': 'input',
        'message': 'Enter your username',
        'validators': [
            {
                'name': 'required'
            }
        ]

    },
    {
        'name': 'favorite_pet',
        'type': 'input',
        'message': 'What is your favorite pet',
        'validators': [
            {
                'name': 'min-length',
                'args': {
                    'min_length': 8
                }
            }
        ]
    }
]
answers = dialogus(
    questions,
    'Please enter your credential',
    finish='login',
    cancel='cancel',
)
```

This way you can read questions from a json or yaml file.

## 1.2 Shell command

Interrogatio ships with a shell command that can be usefull for shell scripting. Questions can be provided both as a json or yaml file (Needs pyYAML).

### 1.2.1 Usage

### Prompt mode

usage: interrogatio [-h] –input INPUT [–output OUTPUT] [–input-format {json,yaml}] [–output-format {json,yaml}] [–theme THEME]

Prompt user for questions.

optional arguments: -h, –help show this help message and exit –input INPUT, -i INPUT

Input file with questions

**--output OUTPUT, -o OUTPUT** Output file to write answers to (Default: STDOUT)

**–input-format {json,yaml}** Questions file format (Default: json)

**–output-format {json,yaml}** Answers file format (Default: json)

**--theme THEME, -t THEME** Name of the UI theme to use (Default: default)

### Dialog mode

..code-block:: bash

$ dialogus –help

**usage: dialogus [-h] –input INPUT [–output OUTPUT] [–input-format {json,yaml}] [–output-format {json,yaml}] [–theme THE**
[–summary] [–previous PREVIOUS] [–next NEXT] [–cancel CANCEL] [–finish FINISH]

Show a wizard dialog to prompt user for questions.

optional arguments: -h, –help show this help message and exit –input INPUT, -i INPUT

Input file with questions

**--output OUTPUT, -o OUTPUT** Output file to write answers to (Default: STDOUT)

**–input-format {json,yaml}** Questions file format (Default: json)

**–output-format {json,yaml}** Answers file format (Default: json)

**--theme THEME, -t THEME** Name of the UI theme to use (Default: default)

**--title TITLE** Title of the dialog

**--intro INTRO** Specify the text of the introduction step (Default: no intro)

**--summary** Show a summary with answers as the latest step (Default: no summary)

**--previous PREVIOUS** Customize the text of the “previous” button (Default: Previous)

**--next NEXT** Customize the text of the “next” button (Default: Next)

**--cancel CANCEL** Customize the text of the “cancel” button (Default: Cancel)

**--finish FINISH** Customize the text of the “finish” button (Default: Finish)

## 1.3 Question handlers

interrogatio 1.0.0 has these built-in type of handlers:

- **input**: for strings and numbers
- **password**: for password

- **selectone**: like a radio list, users have to choose one value from a list of choices.
- **selectmany**: like a checkbox list, users can choose multiple values within a list of choices.

### 1.3.1 input

The `input` handler prompts the user for a string or a number. You can provide it with a default value.

### 1.3.2 password

The `password` handler hides the user input with an asterisk symbol.

### 1.3.3 selectone

The `selectone` handler allow the user to choose from a list of values. To choose a value, users can move up and down the list with the arrow keys, select a value using the space key and accept the answer using the enter key.

The list of values to choose from must be provided as a list of tuples (or two element lists) where, like a html radio input, the first element of the tuple is the value and the second one is the label.

### 1.3.4 selectmany

The `selectone` handler allow the user to choose multiple answers from a list of values.

To choose a value, users can move up and down the list with the arrow keys, select a value using the space key and accept the answer using the enter key.

The list of values to choose from must be provided as a list of tuples (or two element lists) where, like a html radio input, the first element of the tuple is the value and the second one is the label.

This input handler return a list with the chosen values.

### 1.3.5 maskedinput

The `maskedinput` handler allow the user to enter an input given a mask.

### 1.3.6 date

The `date` handler allow the user to enter a date.

### 1.3.7 daterange

The `daterange` handler allow the user to enter a range of dates.

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search