	b.note v3.0.0 **Developer Manual (2024-06-25)**

b.note v3.0.0 Developer Manual (2024-06-25)

Description

This manual describes how to develop a new application in b.note.

The internal applications of b.note are written in python 3 on a RaspberryPi 3 - A+ model.

The user interface of these applications has been written so new applications can be developed quickly.
It gives access to :

- Menu bar,
- Dialogue boxes,
- A document zone leading to keyboard events and displays in braille,
- A speech synthesis.

Necessary equipment

- A personal computer that can support "Pycharm Professional Edition",
- A b.note device,
- A WIFI connection.

Note :
The use of PycharmPE allows the setup of breakpoints, the visualization of variables and the step-by-step on a program running on b.note (remote debugging). This is the least convenient methodology of development.
However, it is possible to modify the source of the application directly on b.note without any development tools.

Access to the sources of b.note

The sources of b.note are the exclusive property of Eurobraille, a developer or a company willing to develop a new application will have to ask Eurobraille that will provide all sources.

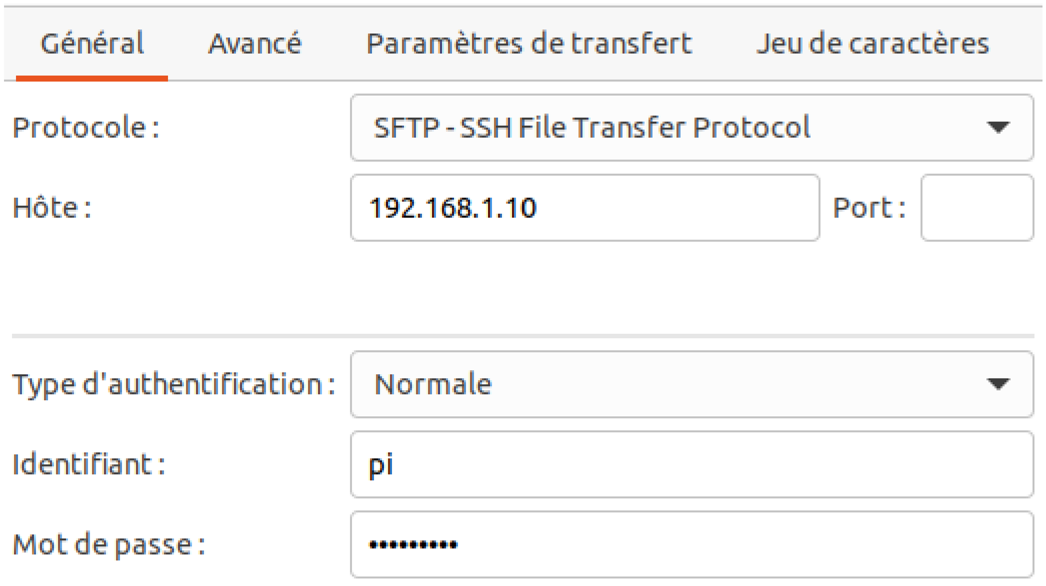
We would like to thank you for your interest in our product and if you develop an application that can be used by the owners of a b.note device, we would be glad to incorporate this application in the official version of b.note.

Connection to the b.note

In the preferences section of b.note, it is possible to set up a wifi connection, you will need to know the SSID and the wifi password of the place where you are and refer to the user manual of b.note.
b.note is configured to request an IP address from the DHCP server of the local network.
This address is visible in the preferences section of b.note (wifi section).

Once these steps are completed, b.note is accessible in ssh console or with filezilla for example.
login : pi
password : euroberry

With filezilla if the IP address of b.note is 192.168.1.10



Protocole :	SFTP - SSH File Transfer Protocol
Hôte :	192.168.1.10
Port :	
Type d'authentification :	Normale
Identifiant :	pi
Mot de passe :	••••••••••

In ssh

```
$ssh pi@192.168.1.10
pi@192.168.1.10's password:
```

```
Linux raspberrypi 5.10.63-v7+ #1488 SMP Thu Nov 18 16:14:44 GMT 2021 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jan 12 09:54:12 2022 from 192.168.1.60
pi@raspberrypi:~ $
```

Enabling the development option

In the settings user interface section switch user interface to yes.
or
A development option allows you to switch b.note into development mode. To do this, change /home/pi/.b.note/settings.txt file by replacing the line :

```
  "system": {
        "braille_type": "dot-8",
        "games": false,
        "developer": false
        },
```

en :

```
 "system": {
        "braille_type": "dot-8",
        "games": false,
        "developer": true
        },
```

After changing this parameter and restarting b.note :

- For example, an application named “Skeleton” can be found in the application menu bar. This application shows how an application can manage its menus, its dialogue boxes, its braille displays and its keyboard events. The first tests may be carried out by modifying this application.

The development folder

To create a development environment independent of the bnote application installed on the device, a develop/ folder is already created on the SD card. It contains a .tar.gz file with the source code of the bnote application.

```
pi@raspberrypi:~/develop $tar -xvf bnote-3.0.0.tar.gz
pi@raspberrypi:~/develop $mv bnote-3.0.0 bnote
pi@raspberrypi:~/develop $cd bnote
pi@raspberrypi:~/develop/bnote $ sh ./setup.sh
```

Development on b.note without PycharmPE

It is possible to modify the source files (.py) of b.note either locally in SSH with nano for example or by copying the source folder in a system, then make the changes on that system and finally by copying the modified folder on b.note.

Then you will need to restart the application:

Solution 1 : By rebooting the pi, if you are connected in SSH, do as follows:

```
pi@raspberrypi:~ $reboot
```

Solution 2 : By stopping the b.note service and manually launching the application :

Stop the application launch service :

```
pi@raspberrypi:~ $sudo systemctl stop b.note.service
```

Manually launch the application :

```
pi@raspberrypi:~ $cd b.note
pi@raspberrypi:~/b.note $python b.note_start.py
```

This second method has the advantage of having the traces of the application in SSH visible on the console.

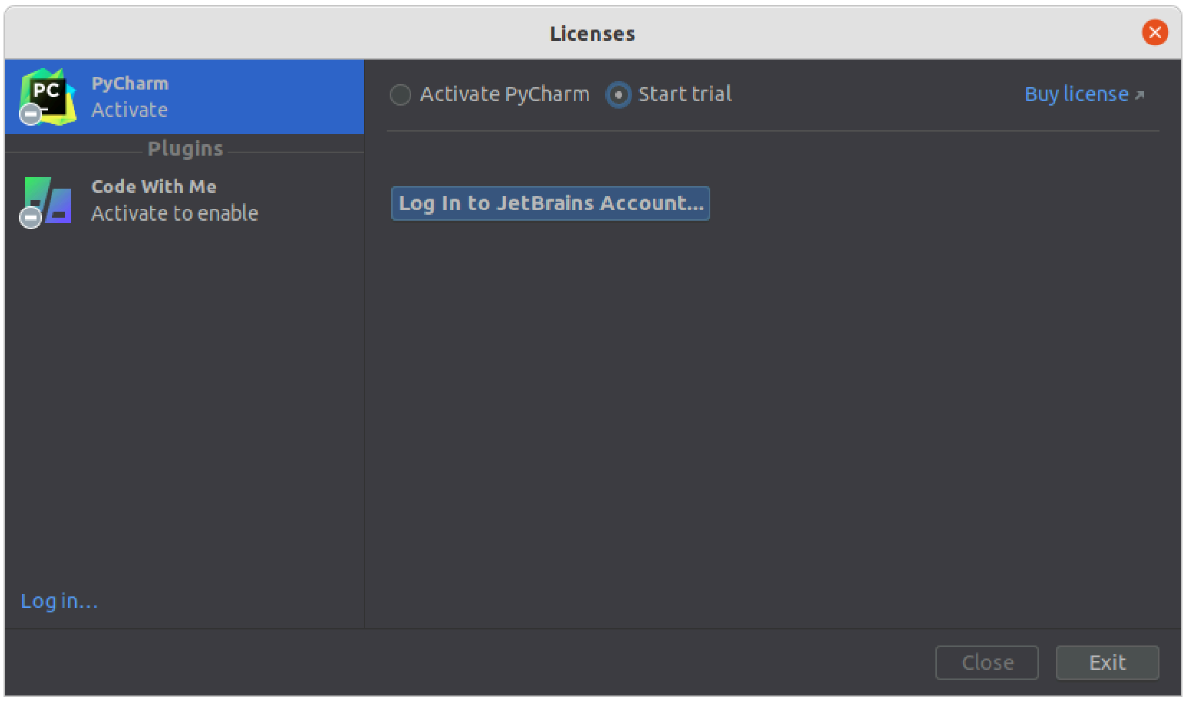
Development on PC with PyCharmPE (Professional Edition)

Unlike the CE version (Community Edition), PyCharmPE will allow “remote debugging”. We tested it under Ubuntu and Windows, and it exists also for Mac.
This version of PyCharm is licensed (There is a trial version limited to 30 days), the editor JetBrain offers various subscription options including a monthly option.

Installation

- Download a version on https://www.jetbrains.com/pycharm/
- Under linux : Extract the tar.gz file and follow the instructions
- Under Windows : Launch the installer (.exe)

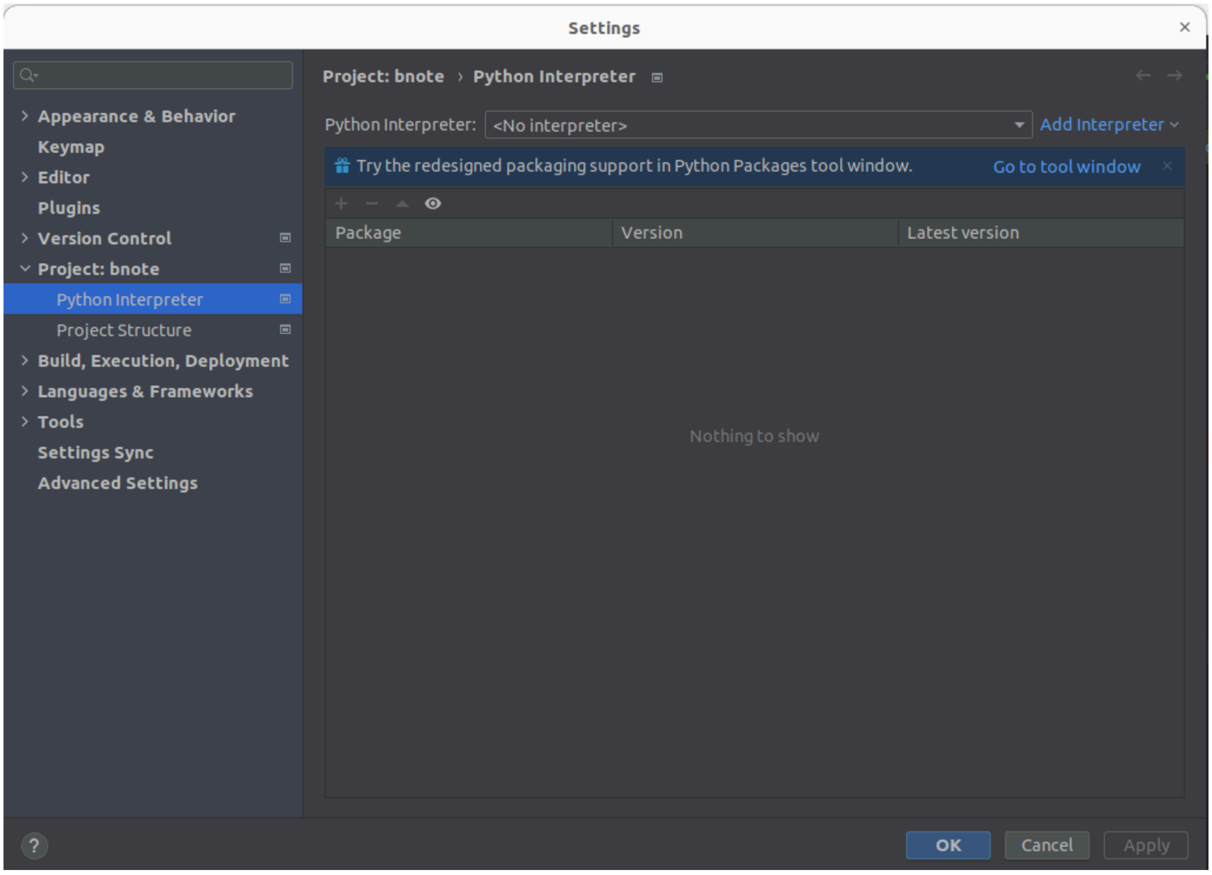
Activation of the test license



Click Log in to JetBrains Account…

Set up of ‘Python Interpreter’

Click file>settings the open Project:b.note>Pythoninterpreter



Settings

Project: bnote › Python Interpreter

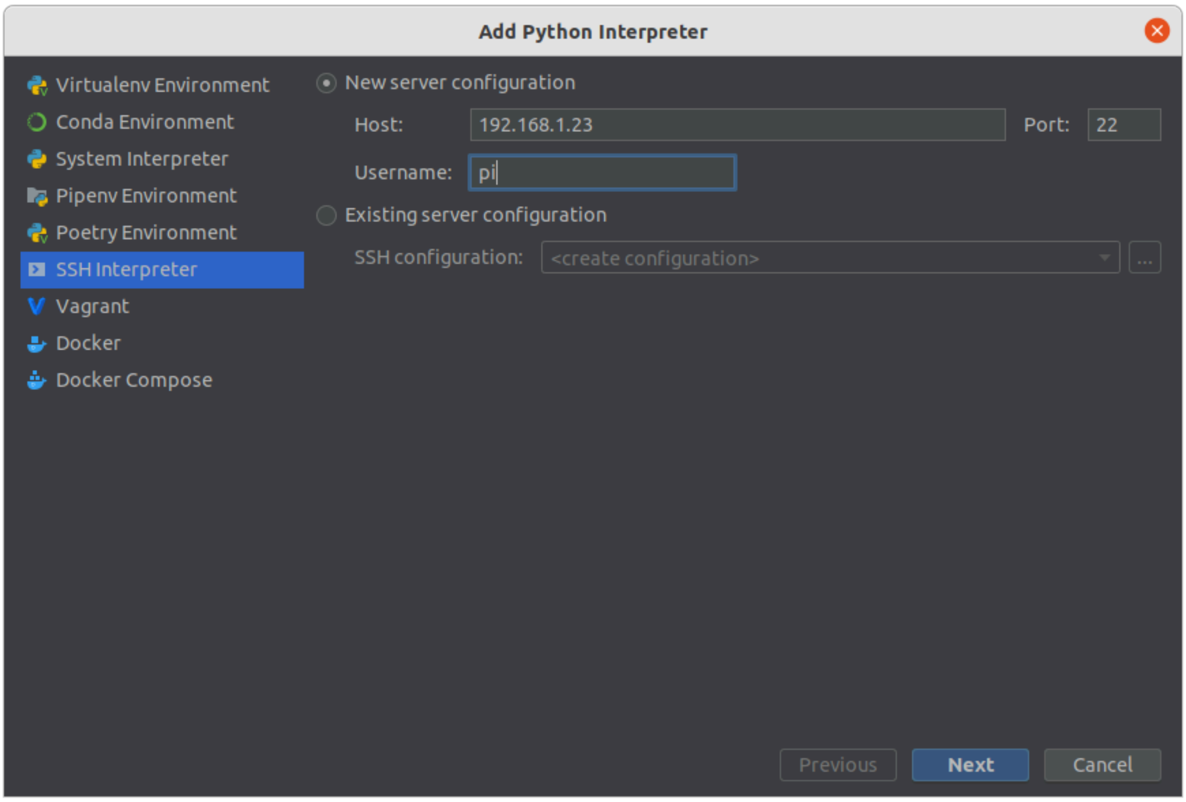
Python Interpreter: <No interpreter> Add Interpreter

Try the redesigned packaging support in Python Packages tool window. Go to tool window

Package	Version	Latest version

Nothing to show

Then click on add interpreter>on SSH
type IP adress then username « pi »



Add Python Interpreter

New server configuration

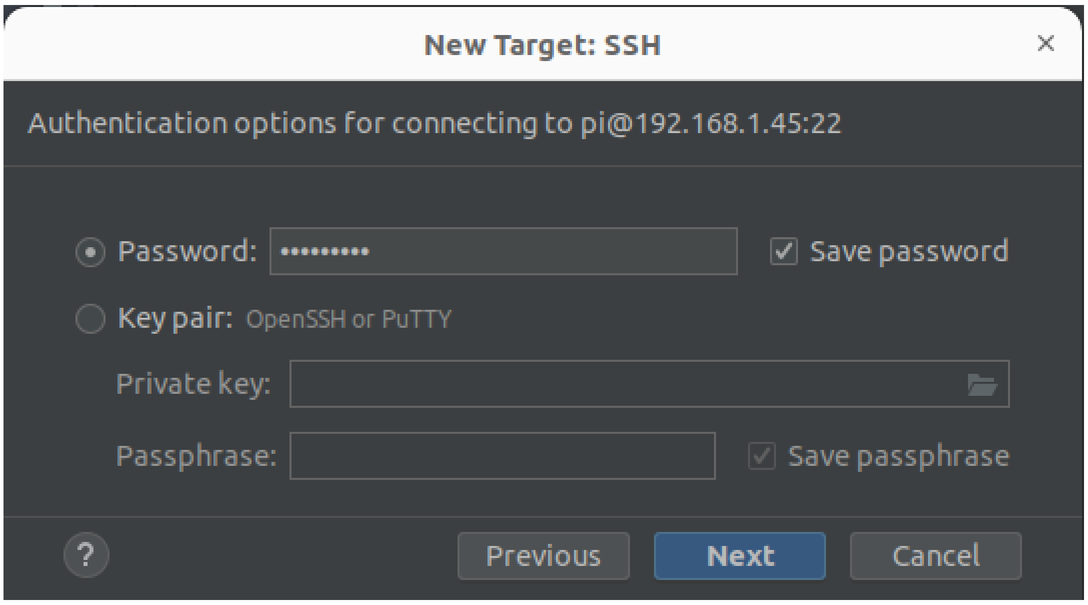
Host: 192.168.1.23 Port: 22

Username: pi

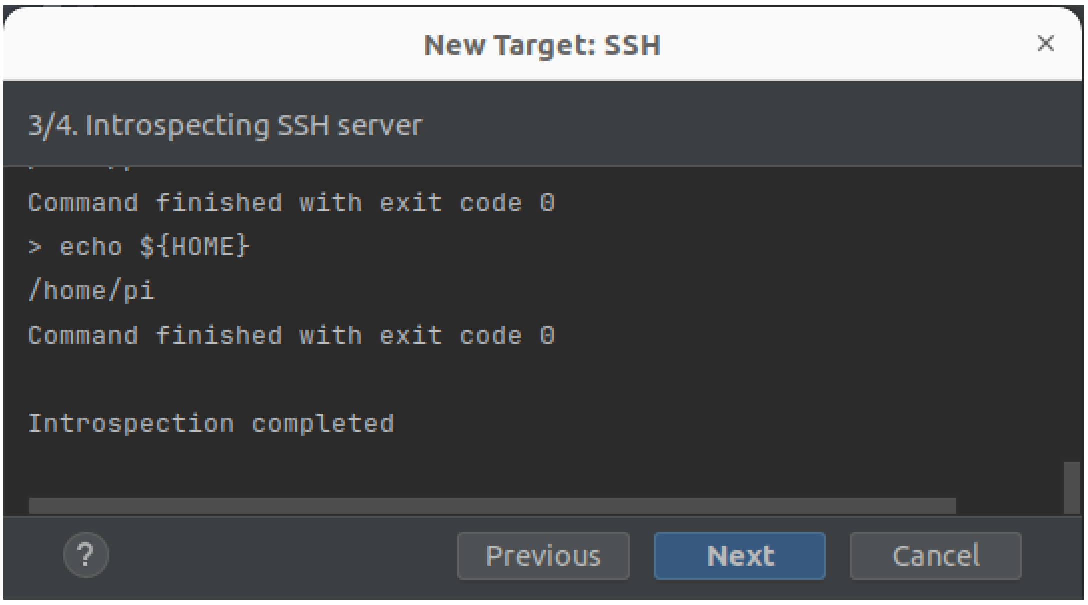
Existing server configuration

SSH configuration: <create configuration>

click on next and type the password « euroberry »



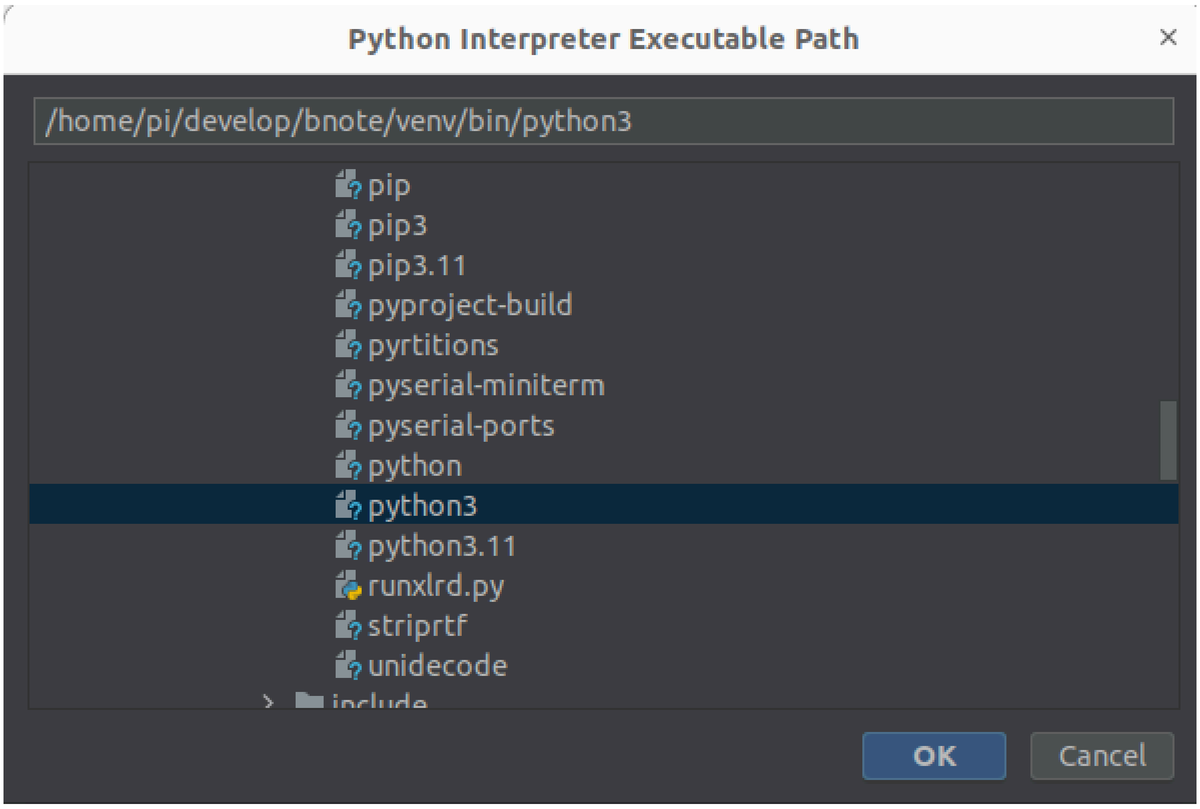
click next pycharm perform a conection test



```
Command finished with exit code 0
> echo ${HOME}
/home/pi
Command finished with exit code 0

Introspection completed
```

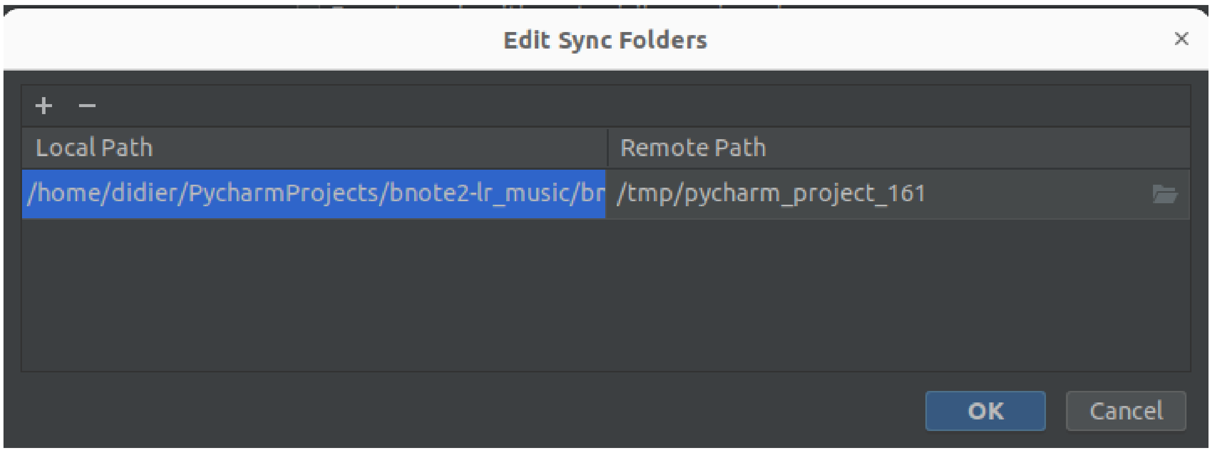
click next then existing and det theVirtualENv Path to
home/pi/develop/bnote/venv/bin/python3


Python Interpreter Executable Path

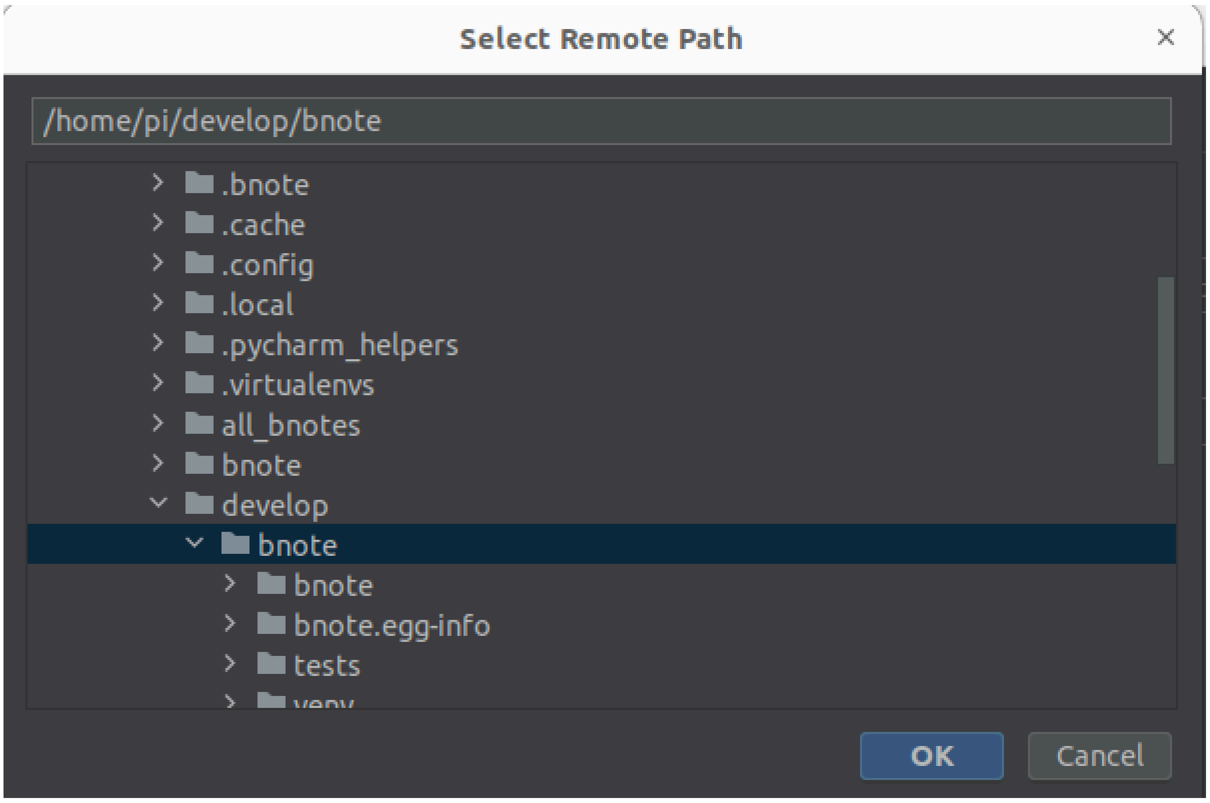
/home/pi/develop/bnote/venv/bin/python3

pip
pip3
pip3.11
pyproject-build
pyrtitions
pyserial-miniterm
pyserial-ports
python
python3
python3.11
runxlrd.py
striprtf
unidecode
include

click on the Sync folders to set up the path where the sources wil be synchronized with Pycharm

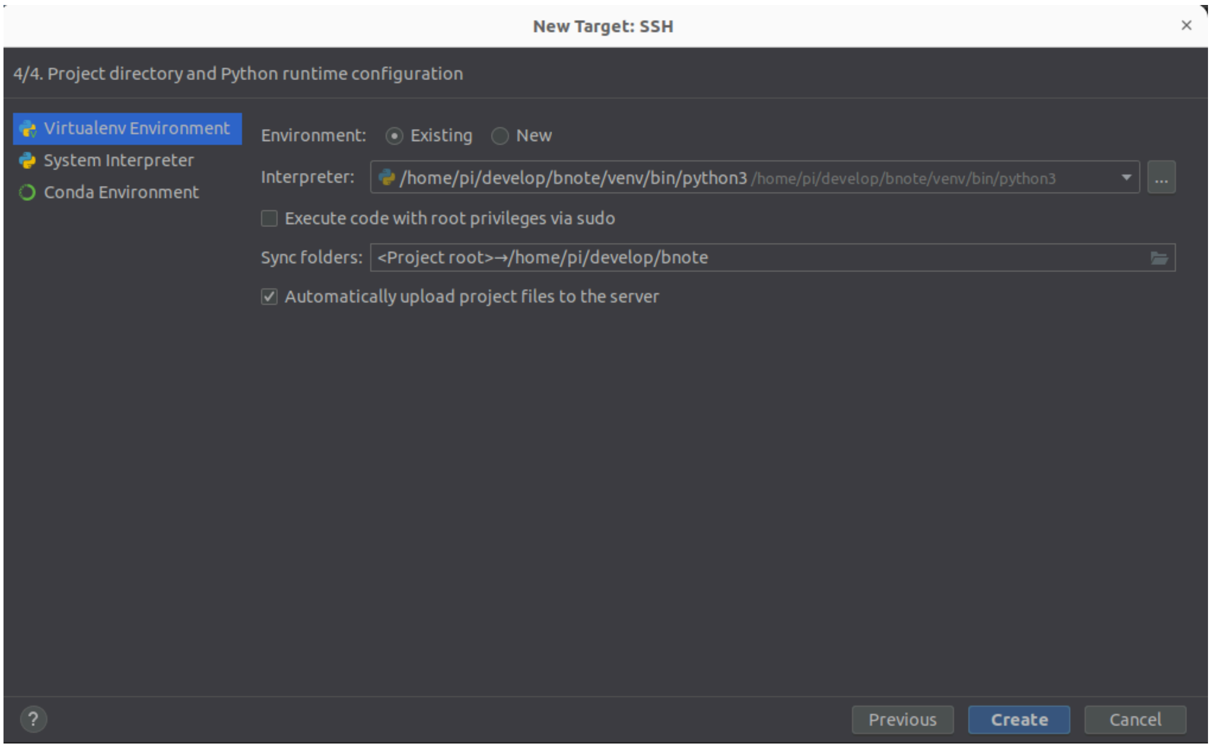

Edit Sync Folders

Local Path	Remote Path
/home/didier/PycharmProjects/bnote2-lr_music/br	/tmp/pycharm_project_161



/home/pi/develop/bnote

```
> .bnote
> .cache
> .config
> .local
> .pycharm_helpers
> .virtualenvs
> all_bnotes
> bnote
v develop
  v bnote
    > bnote
    > bnote.egg-info
    > tests
    > venv
```



4/4. Project directory and Python runtime configuration

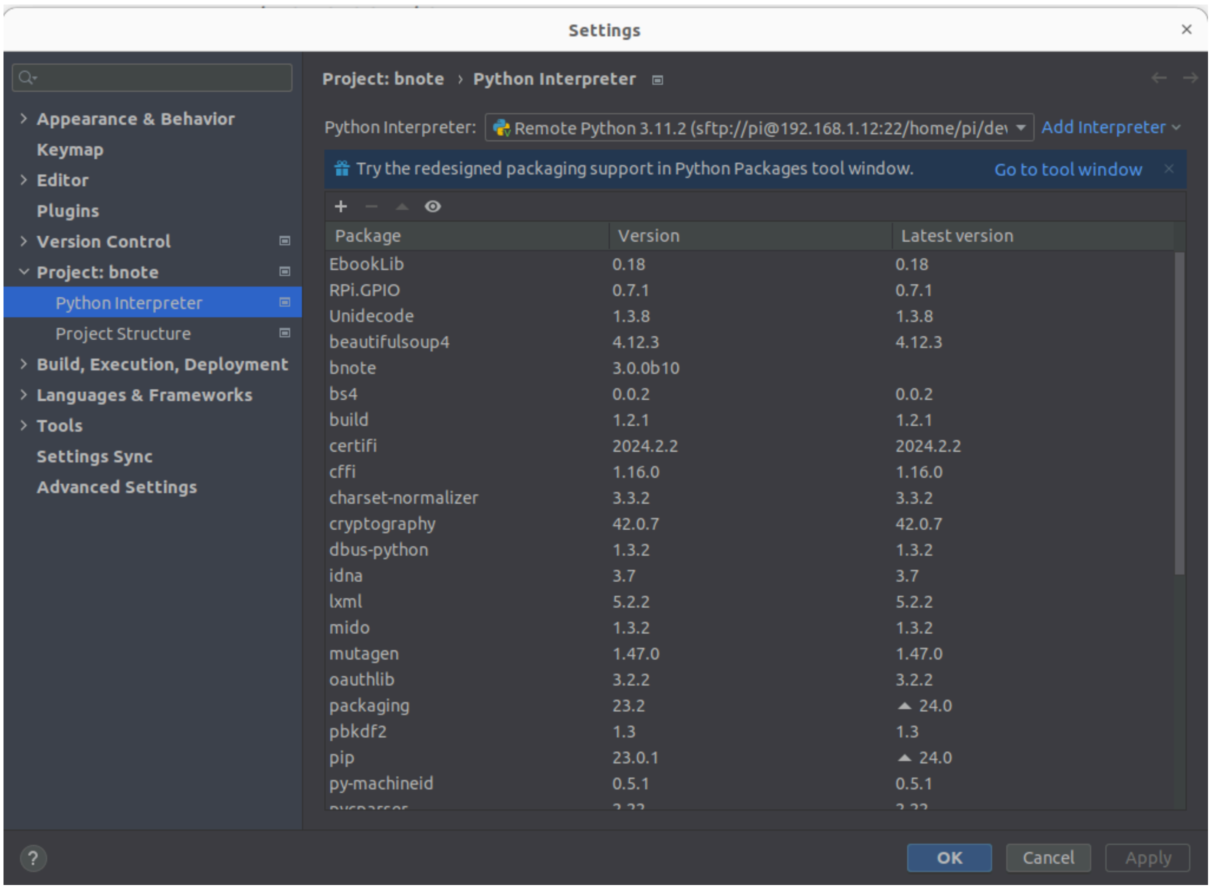
Environment: [✓ Existing] New

Interpreter: /home/pi/develop/bnote/venv/bin/python3 /home/pi/develop/bnote/venv/bin/python3

☐ Execute code with root privileges via sudo

Sync folders: <Project root>→/home/pi/develop/bnote

☑ Automatically upload project files to the server



Settings

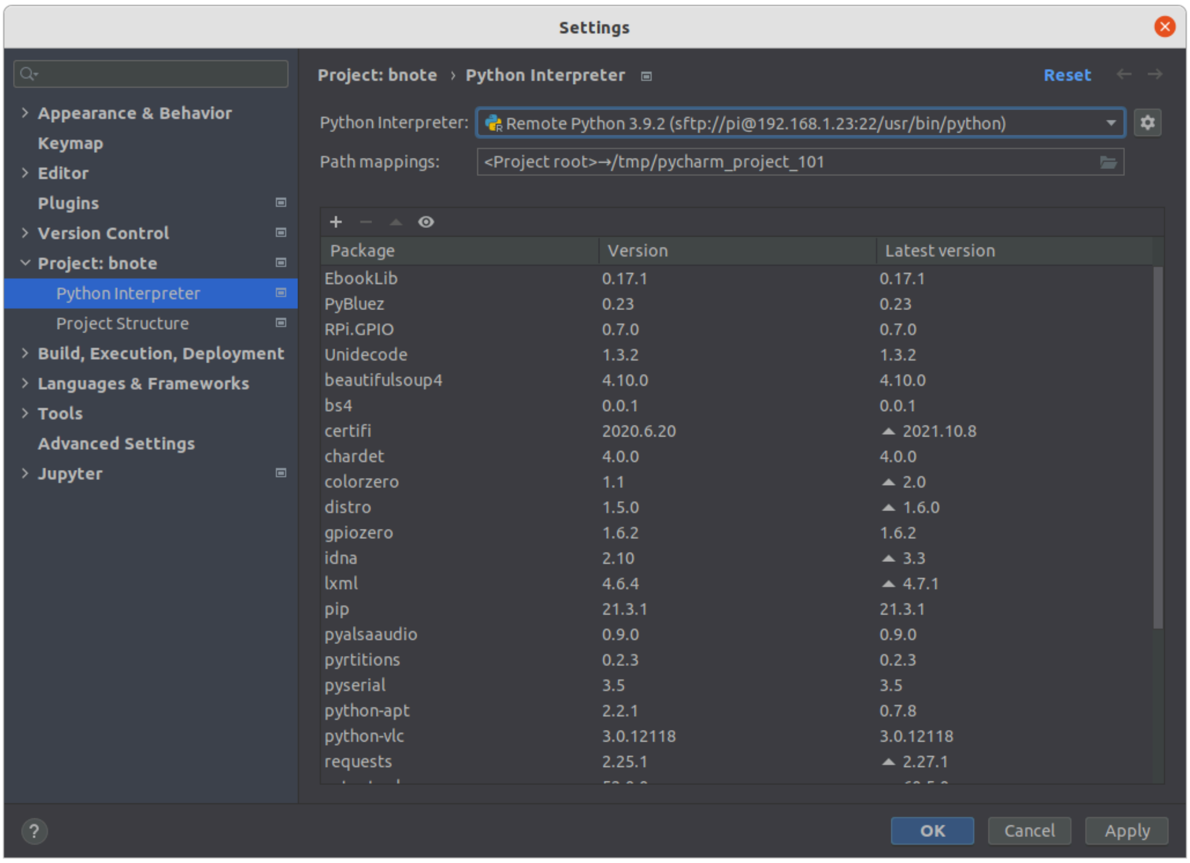
Project: bnote › Python Interpreter

Python Interpreter: Remote Python 3.11.2 (sftp://pi@192.168.1.12:22/home/pi/dev   Add Interpreter

Try the redesigned packaging support in Python Packages tool window.   Go to tool window

Package	Version	Latest version
EbookLib	0.18	0.18
RPi.GPIO	0.7.1	0.7.1
Unidecode	1.3.8	1.3.8
beautifulsoup4	4.12.3	4.12.3
bnote	3.0.0b10	
bs4	0.0.2	0.0.2
build	1.2.1	1.2.1
certifi	2024.2.2	2024.2.2
cffi	1.16.0	1.16.0
charset-normalizer	3.3.2	3.3.2
cryptography	42.0.7	42.0.7
dbus-python	1.3.2	1.3.2
idna	3.7	3.7
lxml	5.2.2	5.2.2
mido	1.3.2	1.3.2
mutagen	1.47.0	1.47.0
oauthlib	3.2.2	3.2.2
packaging	23.2	▲ 24.0
pbkdf2	1.3	1.3
pip	23.0.1	▲ 24.0
py-machineid	0.5.1	0.5.1

Choose OK then Create



Python Interpreter: Remote Python 3.9.2 (sftp://pi@192.168.1.23:22/usr/bin/python)

Path mappings: <Project root>→/tmp/pycharm_project_101

Package	Version	Latest version
EbookLib	0.17.1	0.17.1
PyBluez	0.23	0.23
RPi.GPIO	0.7.0	0.7.0
Unidecode	1.3.2	1.3.2
beautifulsoup4	4.10.0	4.10.0
bs4	0.0.1	0.0.1
certifi	2020.6.20	▲ 2021.10.8
chardet	4.0.0	4.0.0
colorzero	1.1	▲ 2.0
distro	1.5.0	▲ 1.6.0
gpiozero	1.6.2	1.6.2
idna	2.10	▲ 3.3
lxml	4.6.4	▲ 4.7.1
pip	21.3.1	21.3.1
pyalsaaudio	0.9.0	0.9.0
pyrtitions	0.2.3	0.2.3
pyserial	3.5	3.5
python-apt	2.2.1	0.7.8
python-vlc	3.0.12118	3.0.12118
requests	2.25.1	▲ 2.27.1

Once these operations have been completed, you can right-click on the root bnote folder of the project then choose Deployment>Upload to … to synchronize the sources of your bnote development folder with those of pycharm.
You can then launch bnote_start.py from pycharm.

For more information, check this link :
https://www.jetbrains.com/help/pycharm/remote-debugging-with-product.html

Editing bnote packages

Python package management

In ssh on bnote, from your development folder, you can type:

```
pi@raspberrypi:~/develop/bnote $ source venv/bin/activate
(venv)pi@raspberrypi:~/develop/bnote $pip install my_package
```

To retrieve the package and its dependencies you can use the following commands:

```
pi@raspberrypi:~/develop/bnote $mkdir -p /tmp/packages
pi@raspberrypi:~/develop/bnote $pip download --dest /tmp/packages requests
```

You can then download it and include it in the whl/ folder of your update and add its name and version in pyproject.toml of your bnote project.

to remove a python package.

```
pi@raspberrypi:~/develop/bnote $pip uninstall my_package
```

Linux library management

In ssh on bnote, from your development folder, you can type:

```
pi@raspberrypi:~/develop/bnote $sudo apt install my_library
```

to add a library.

```
pi@raspberrypi:~/develop/bnote $sudo apt remove my_library
```

to remove a library.

When you perform the update, you will need to take library changes into account by updating the libraries.txt file.
Each of the commands in this file will be executed when installing the new version of the application. An error during these executions will result in a failure to install the version.
This file will continue to grow with updates.
Creation and execution of a bnote update
In ssh on develop/bnote, from your development folder type:

```
pi@raspberrypi:~/develop/bnote $sh./generate.sh
```

This will generate a bnote-....whl.zip file with the name of the bnote version defined in pyproject.toml.

Running a bnote update

It is the bnote…whl.zip file which will allow the installation of the application. All you have to do is copy it to the target bnote and run it from the bnote file explorer.

The new version will be installed in the all_bnotes folder with its own virtual environment.
It is therefore possible to have several versions of bnote coexist and choose the one that will be launched at startup.

Structure of the application

The source code is in the /home/pi/b.note folder.
The documents folder visible from the application can be found in /home/pi/.b.note

The application

All b.note applications come from a b.noteApp class located in the b.note_app.py file.
The menus, the dialogue boxes and the braille refresh mechanism of the app work thanks to this basic class.

The keyboard events

The keyboard events have been categorized into 4 types that each match to a python function of the application. They come and override the base class, therefore this is important because they call for the functions of the base class as it is done in skeleton.py to make sure the dialogue boxes and the menus work properly.

Control keyboard

Pressing one or more keys from the 2 keypads will trigger this following event :

```
def input_command(self, data, modifier, key_id) -> bool:
    """
    Does what is expected for this command key.
    :param data: ?
    :param modifier: bits field (see Keyboard.BrailleModifier)
    :param key_id: (see Keyboard.KeyId)
    :return: True if command treated, otherwise False
    """
```

Note : An event with key_id = `Keyboard.KeyId.KEY_NONE` is generated when all keys are released. It must be ignored for all applications of b.note.

Braille keyboard

The key combinations of the braille keyboard are divided into 2 categories :

- Combinaisons leading to alphanumeric character (input_character())
- Combinaisons leading to a function (input_bramigraph())

```
def input_character(self, modifier, character, data) -> bool:
    """
    Do what needs to be done for this braille modifier and character.
    :param modifier: bits field (see Keyboard.BrailleModifier)
    :param character: unicode char
    :param data: brut braille comb. for advanced treatment
    :return: True if command treated, otherwise False
    """
```

```
def input_bramigraph(self, modifier, bramigraph) -> bool:
    """
    Do what needs to be done for this modifier and bramigraph.
    :param modifier: bits field (see Keyboard.BrailleModifier)
    :param bramigraph: braille function (see
Keyboard.BrailleFunction)
    :return: True if command treated, otherwise False
    """
```

Routing cursor keyboard

Pressing a routing cursor key called interactive key in b.note will trigger this following function :

```
def input_interactive(self, modifier, position, key_type) -> bool:
    """
    Do what needs to be done for this modifier and cursor routine
event.
    :param modifier: bits field (see Keyboard.BrailleModifier)
    :param position: index of key (based 1)
    :param key_type: see Keyboard.InteractiveKeyType
    :return: True if command treated, otherwise False
    """
```

Other events

Function

If the application uses the multi-task mode, it may need to trigger functions that will be taken into account by the main process, these events will trigger the following function :

```
def input_function(self, *args, **kwargs) -> bool:
    """
    Call when function is not treated by base class of this class.
    :param args[0]: The function id
    :param kwargs:
    :return: True if function treated.
    """
```

In order to trigger a function, you must type :

```
self._put_in_function_queue(FunctionId.FUNCTION_…)
```

Timer

A timer event occurs every second

```
def on_timer(self):
    """
    Event each seconds
    :return: None
    """
```

The refreshment of the braille display

A single function allows to display a line of text on the braille display

```
def set_data_line(self):
    """
    Construct the braille display line from document
    :return: None (self._braille_display.set_data_line is done)
    """
```

Usually, this function allows the line of the document from the application to be sent to the braille display.
The coding of this function consists of building 3 buffers of alphanumeric text (for esyviewer for example)

- A unicode braille buffer describing the fixed points
- A unicode braille buffer describing the blinking points

then name *self._braille_display.set_data_line* with those 3 buffers as parameters.

Standard objects

The menus

The menus are generated by 2 classes of folder ui :

- `UiMenuBar` `The menu bar and a submenu`
- `UiMenuItem` `An item from terminal menu`

```
A UiMenuBar is specified by :
'name': the name of submenu,
'action': (None by default) A function of the application,
'ui_objects': A list describing its content, it will be made of
UiMenubar and UiMenuItem,
'is_root': (False by default) this Boolean will allow to mark the
root element of the menu description,
'focused_object': 0

A UiMenuItem is specified by :
'name': the name of the menu item,
'shortcut_modifier': the keyboard shortcut modifiers associated to
this element,
'shortcut_key': the description of the shortcut key,
'action': Each element of the terminal menu is associated to a
function from the application called convention
def _exec_menu_nom_de_la_fonction(self),
'action_param': (None by default) This parameter enables the
```

```
specification of the settings to be carried out once the action is
launched,
'is_hide': (False by default) Hiding the menu item,

Example for creating a menu :
def __create_menu(self):
   # Instantiate menu (A menu bar with 1 sub menu of 2 menu items and one menu
item).
   return UiMenuBar(
       name=_("skeleton"),
       # Call on ESC bramigraph key
       is_root=True,
       menu_item_list=[
           UiMenuBar(
               name=_("&group"),
               menu_item_list=[
                   UiMenuItem(name=_("&menu_1"), action=self._exec_menu_1),
                   UiMenuItem(name=_("&menu_2"), action=self._exec_menu_2),
               ]),
           UiMenuItem(name=_("&say hello"), action=self._exec_say_hello),
           UiMenuItem(name=_("&about"), action=self._exec_about,
                      shortcut_modifier=Keyboard.BrailleModifier.BRAILLE_FLAG_NONE,
                      shortcut_key = Keyboard.BrailleFunction.BRAMIGRAPH_F1),
       ],
   )
```

The dialogue boxes

Elements

The dialogue boxes are managed by the classes of the following ui folder :

- `UiDialogBox` The dialogue box
- `UiCheckBox` A checkbox
- `UiListBox` A list box
- `UiEditBox` An editable box
- `UiLabel` A text that can not be modified
- `UiButton` A button

Predefined dialogue boxes

Some standard dialogue boxes are already defined in order to globalize boxes of general use and spare the applications code. They correspond to the following classes :

- `UiMessageDialogBox` A dialogue box that includes a name, a message and a list of button,
- `UiInfoDialogBox` An information dialogue box that includes a

```
    message and an OK button,
```

Braille management

The braille is managed by a global class b.noteApp.lou that corresponds with an instance of liblouis in the language used for the device.

The key functions

```
def to_dots_8(self, txt):
    """
    Convert a string into an unicode braille string (8 dots) (x28nn chars).
    : param txt : (str) alphanumeric string of text
    : return : (str) unicode braille string (\u28xx...)
    """
```

This function converts a text into a braille character line. This is the only useful function for an application in 8-dot computer braille mode. It will convert the line to the braille display.

Speech synthesis

b.note has a global entry point to enable speech synthesis.

```
def speak(text, lang_id=None, volume=None, speed=None,
purge_before_speak=True):
```

In order to get an advanced use of the speech synthesis, it will be necessary to interface directly to the class SpeechManager().