TuneFlow插件开发

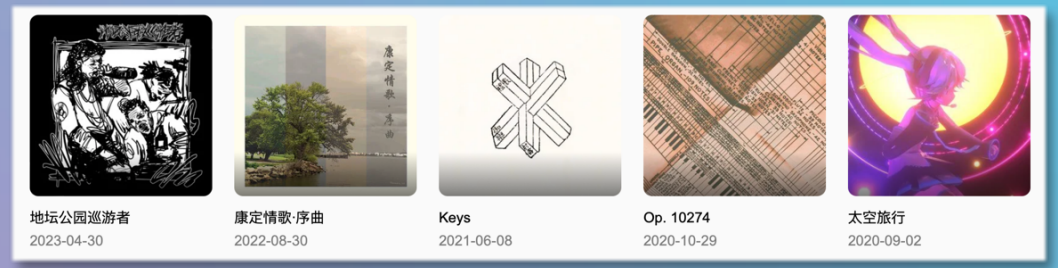
CSMT 2023

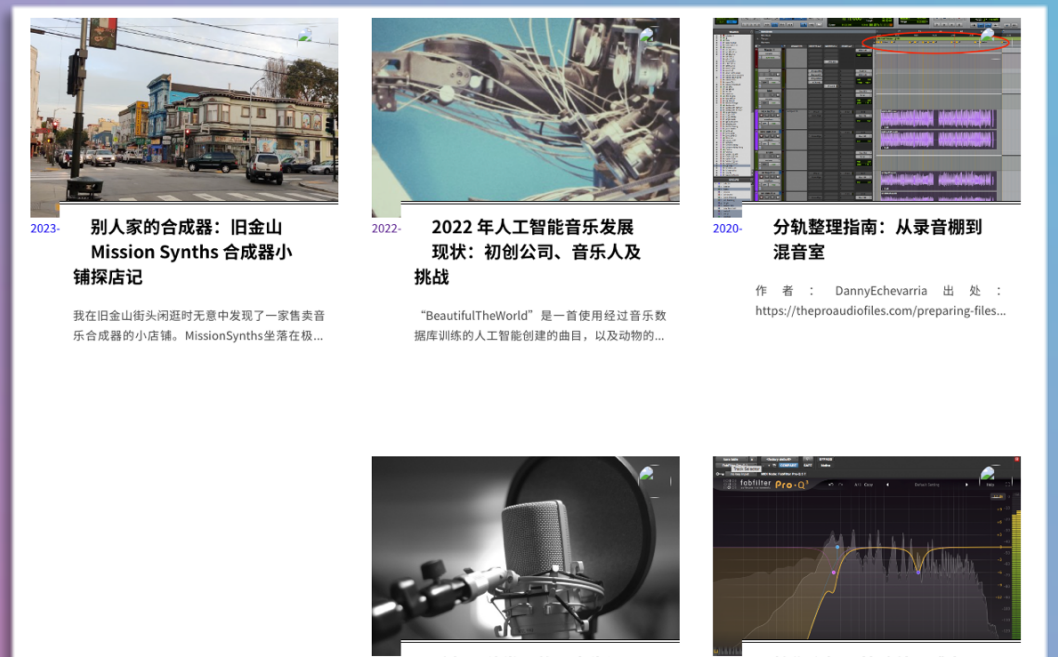
李克镰

李克镰

作曲家，音频工程师。本科及硕士研究生分别就读于美国乔治梅森大学和乔治亚理工学院音乐技术专业，发表论文数篇。研究领域包括深度学习、自动混音、音乐交互等。

个人主页：www.likelian.net



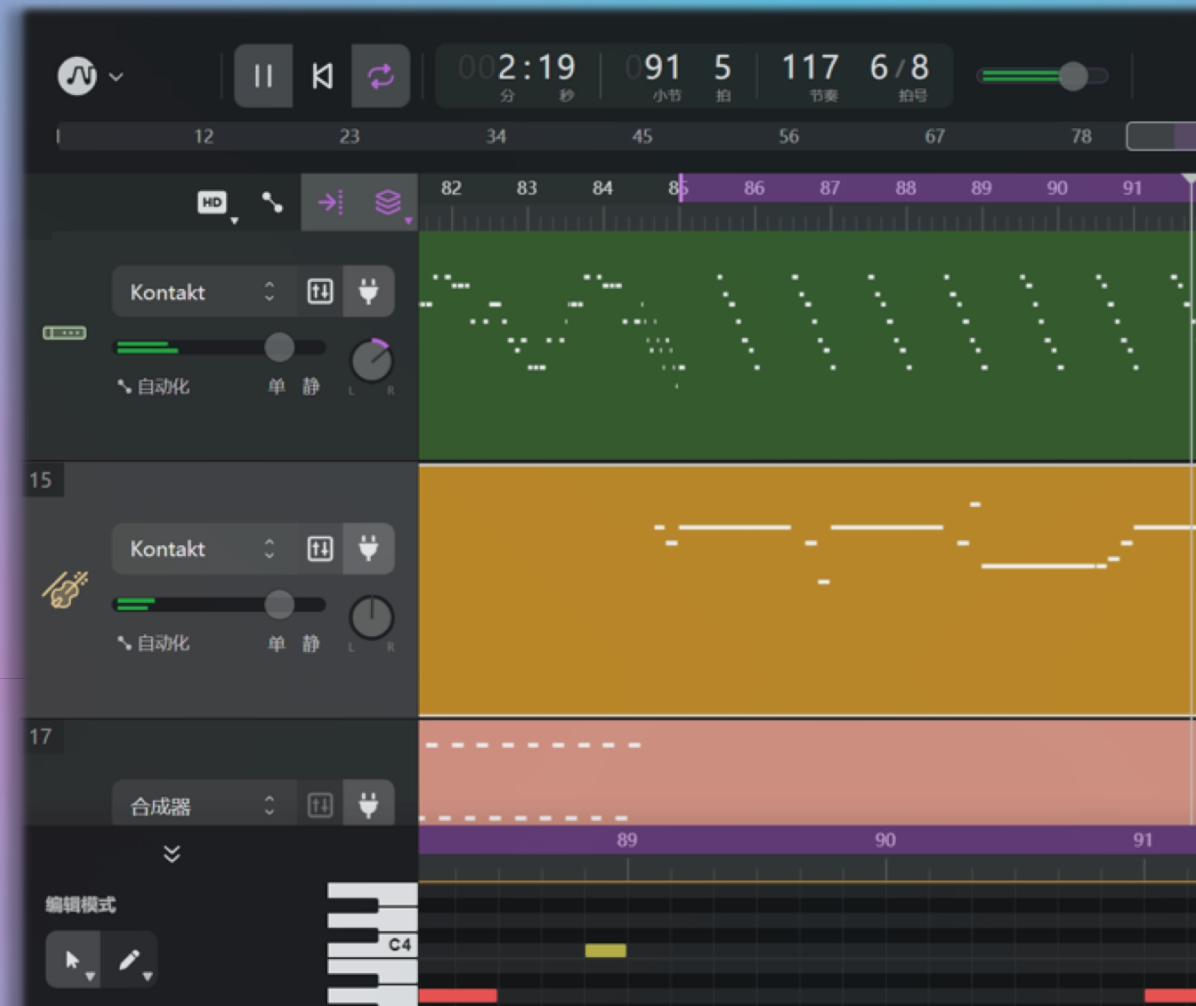


TuneFlow

作为新一代AI助力的音乐制作工具，TuneFlow配备了一整套强大的人工智能算法，覆盖完整音乐制作流程。

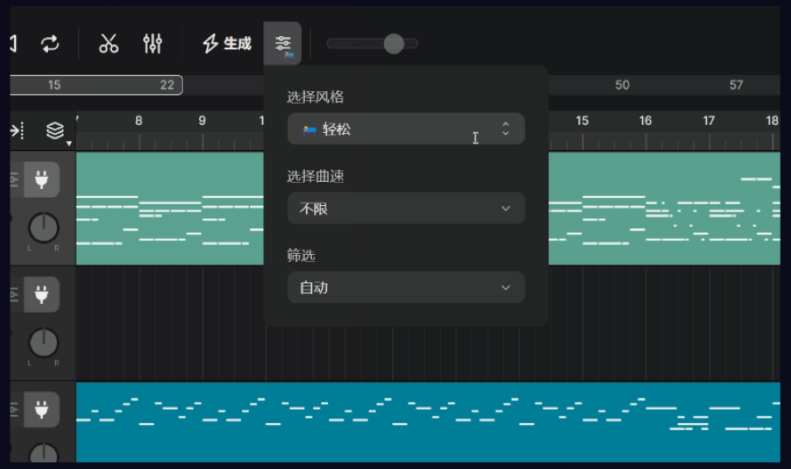
只需打开浏览器，用户就可以通过TuneFlow网页版完成高效的音乐写作。

官方网站：www.TuneFlow.com



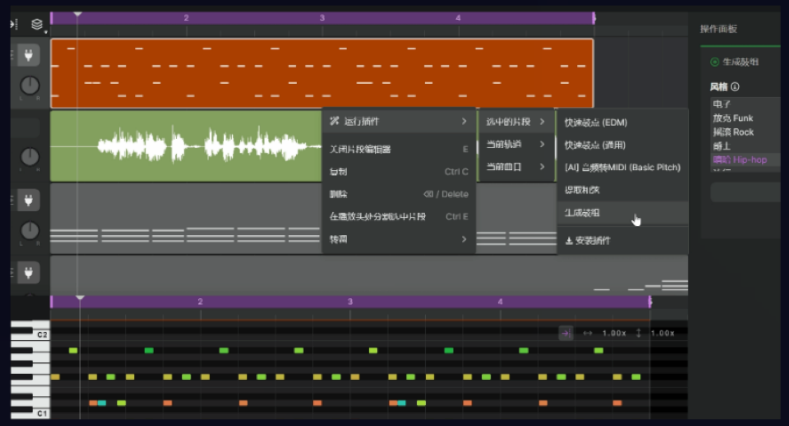
智能作曲

用一首完整的曲子开启你的音乐创作。选择你喜欢的风格和节奏，一眨眼的时间，你就有了完整的主旋律和更多的伴奏轨道。



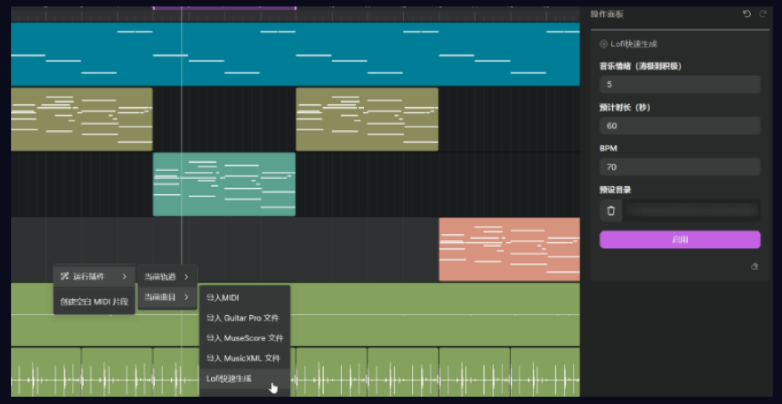
智能鼓手

选择你喜欢的鼓点风格，让AI鼓手为你完成创作。只需几秒，你的创意鼓点就写作完成.



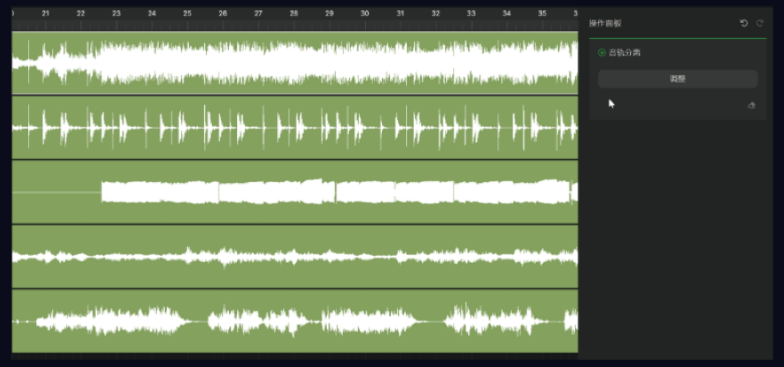
一键 Lo-Fi

想要一些轻松的 Lo-Fi 音乐？有了这个AI插件，你可以一键生成一首完整的 Lo-Fi 曲目，并且精确地精确控制情绪和时长。



智能音频分轨

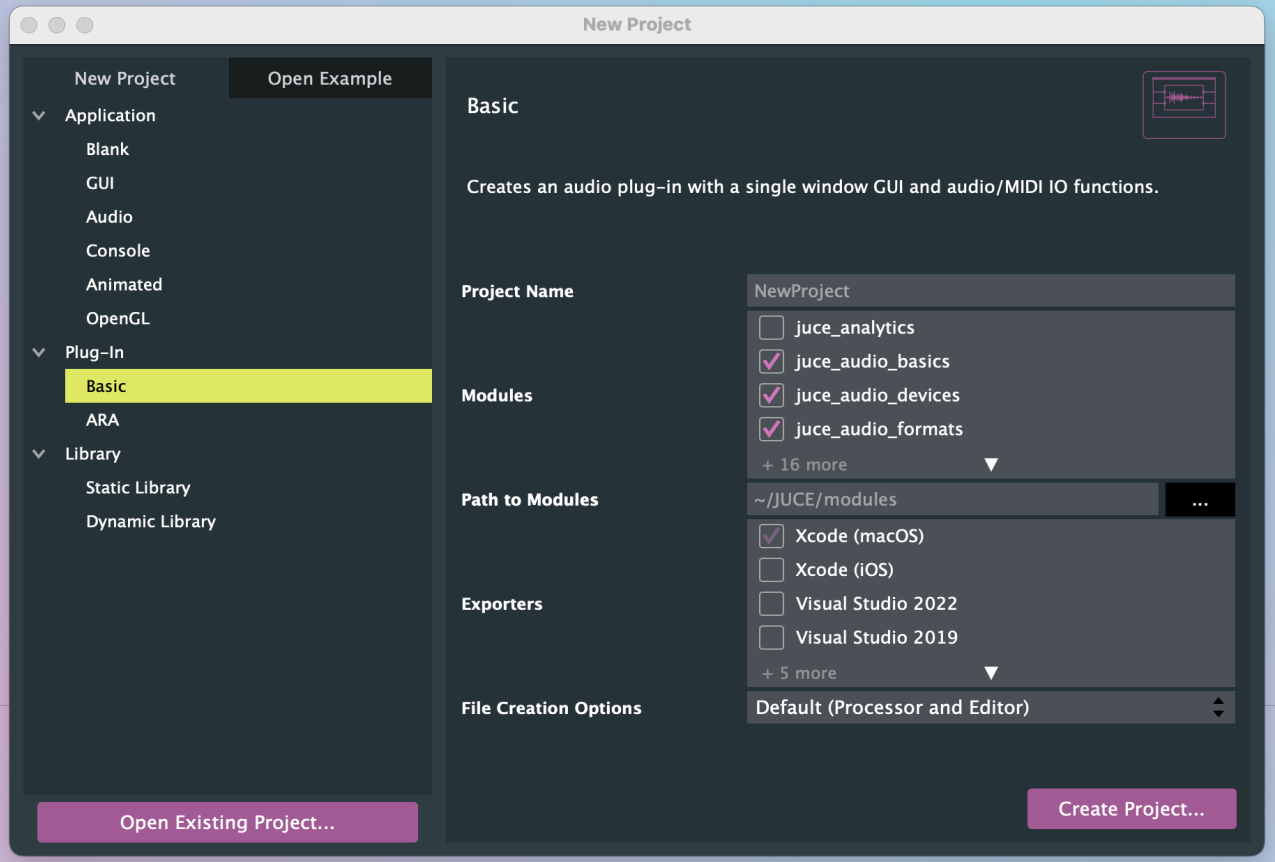
将原始音频分离成人声、鼓、贝斯和其他音轨。通过 TuneFlow 行业领先的AI算法模型，你可以把每个轨道分的干干净净。



传统音频插件

传统音频插件的设计思路基于实时信号处理，因此可实现的功能有诸多限制。

开发环境依赖C++，难度较大。



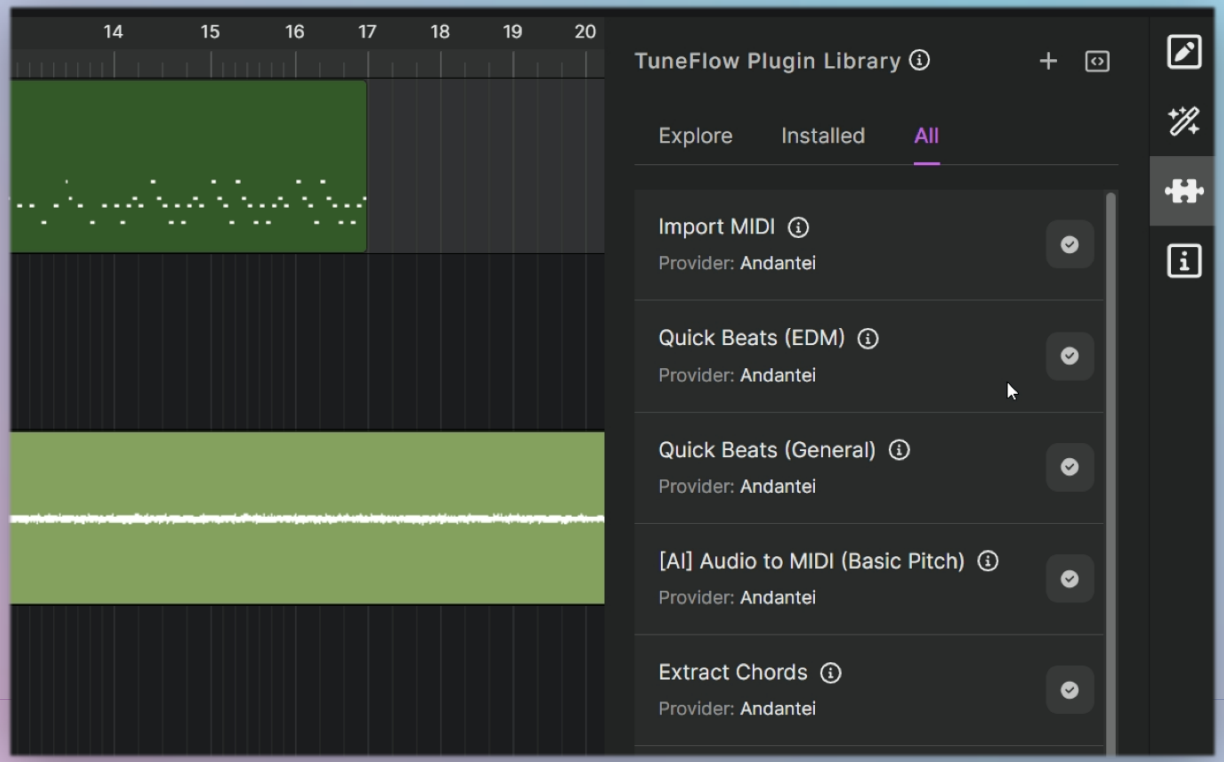
TuneFlow插件

TuneFlow插件的优势：

可对DAW全局进行读取和修改，突破传统音频插件限制。

使用Python开发，难度较低，并且方便接入其他Python库。

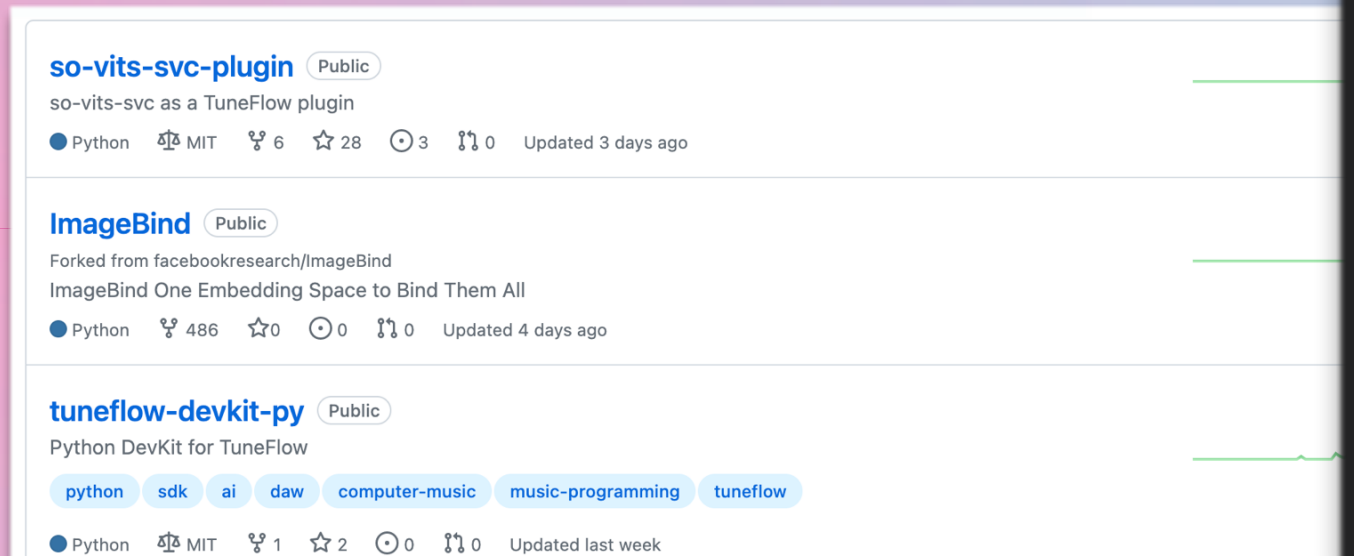
部署后在云端运行，不占用本地运算资源。

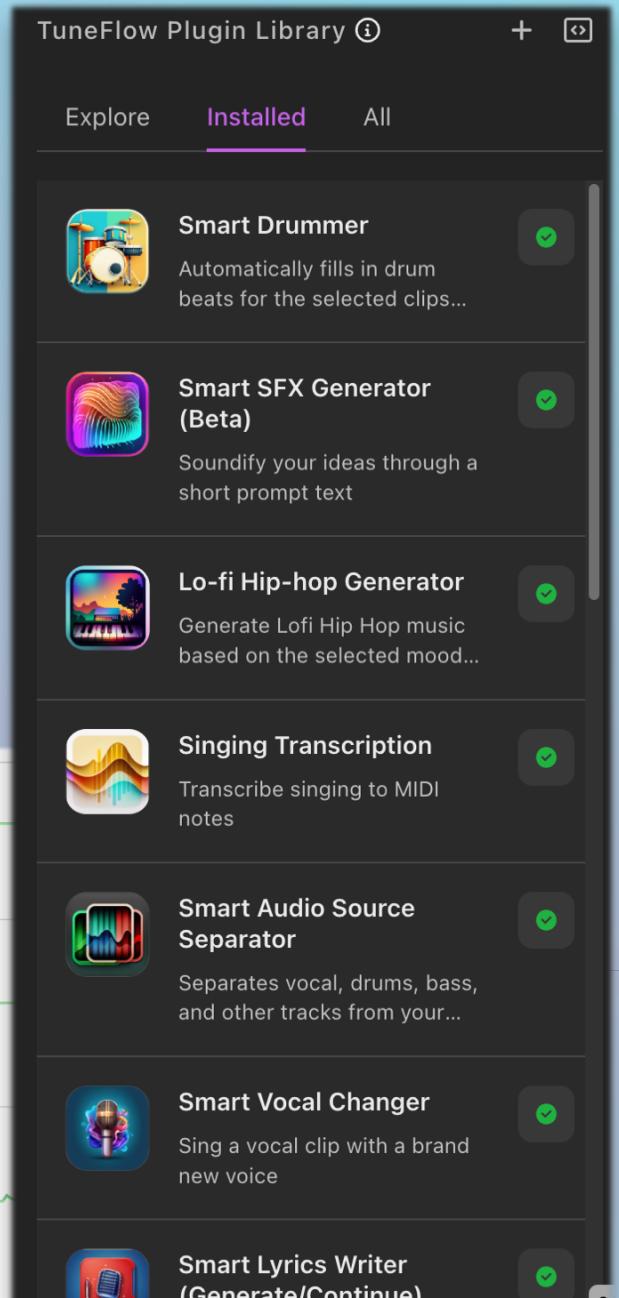


TuneFlow插件展示

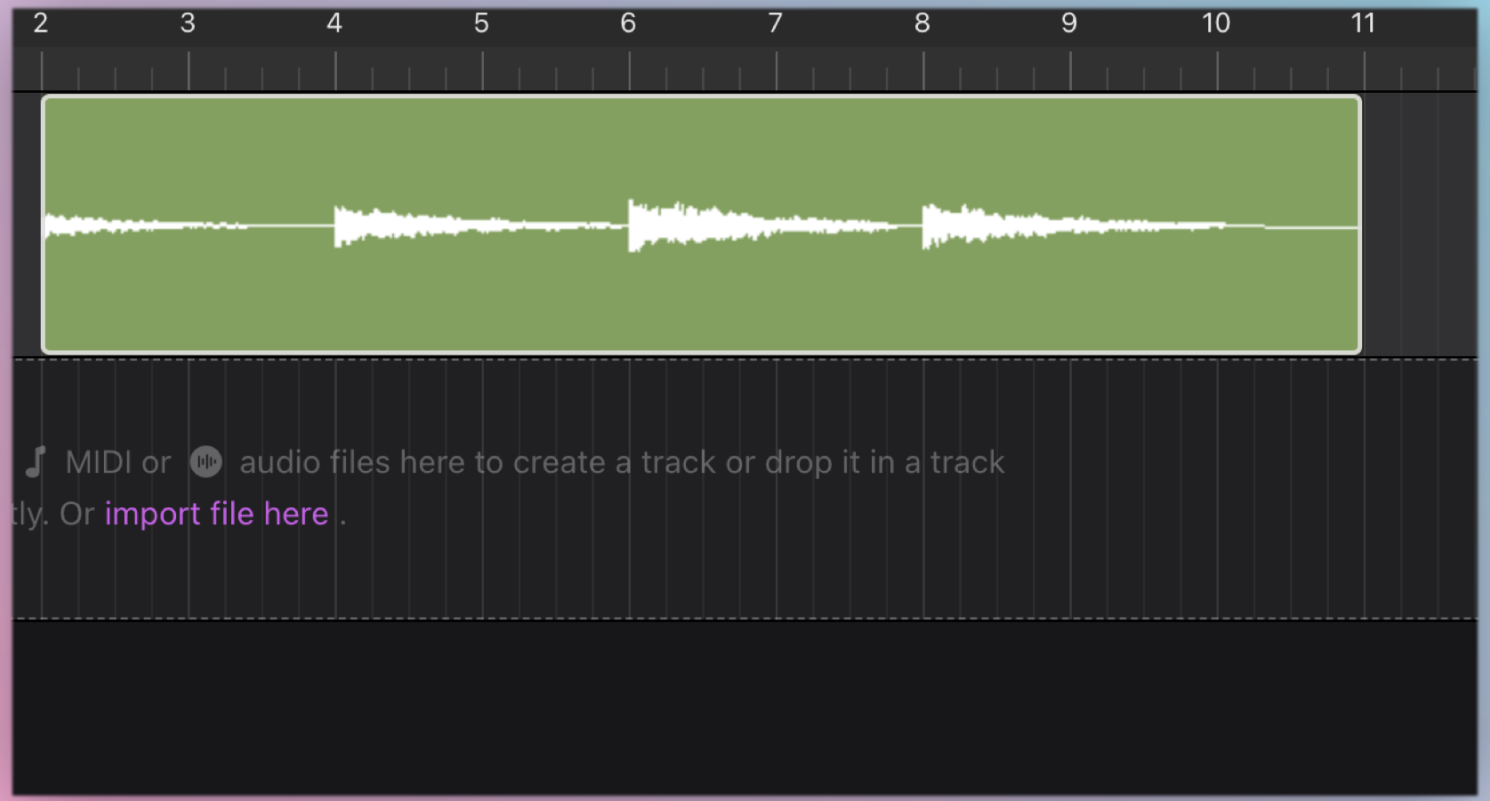
TuneFlow已有插件可以实现鼓组生成，音效生成、音源分离、歌声转换、歌词写作等功能。

代码示例：
https://github.com/tuneflow
https://github.com/tuneflow/tuneflow-py-demos





插件示例：音频反转

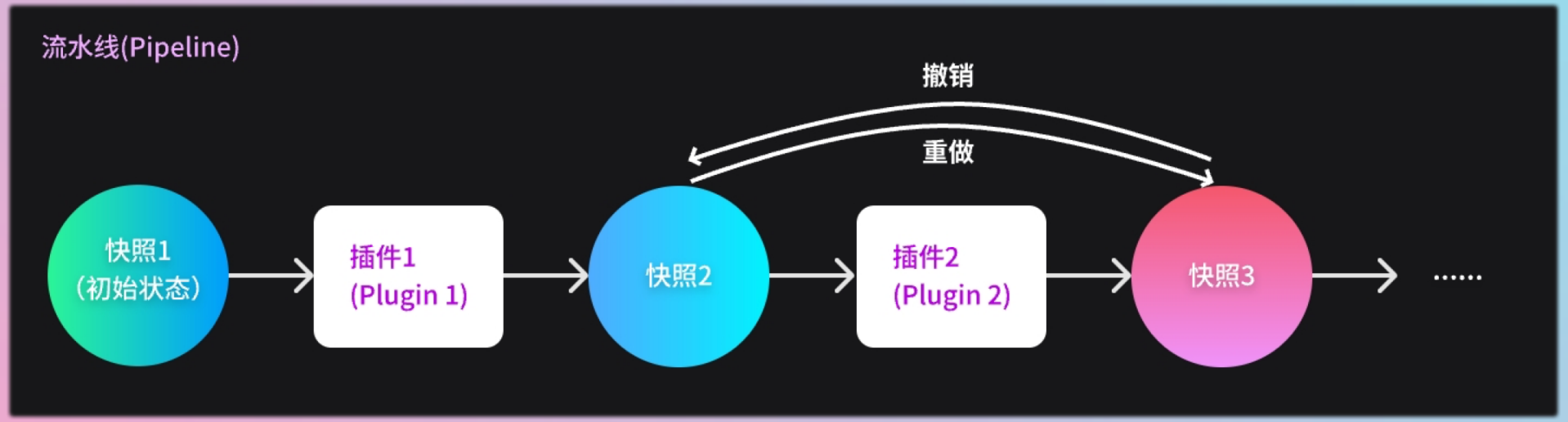


https://github.com/likelian/tuneflow-plugin-demo

插件示例：音频反转

https://github.com/likelian/tuneflow-plugin-demo

TuneFlow插件



TuneFlow插件能对DAW工程进行控制，可以实现创建轨道、添加音频和Midi片段、调节音量、效果器参数自动化等等功能。

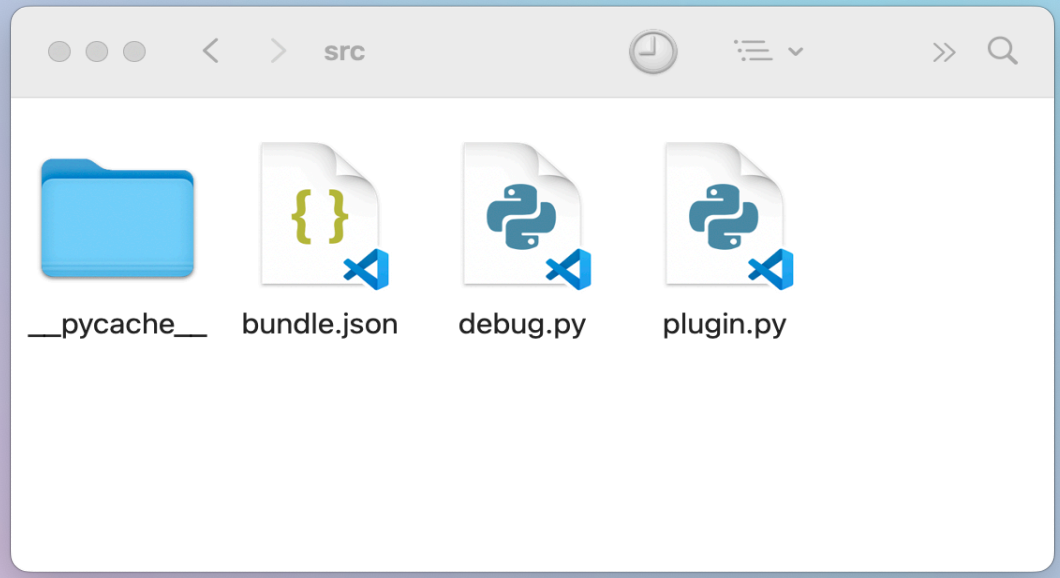
https://help.tuneflow.com/zh/developer/getting-started-python.html

插件包

一个TuneFlow插件包含3个文件：

- bundle.json
- plugin.py
- debug.py

bundle.json包含必要信息描述
plugin.py是插件运行的主体
debug.py是调试运行的入口



https://help.tuneflow.com/zh/developer/getting-started-python.html

bundle.json

Bundle.json描述记录了一些必要的插件信息。

其中providerId和pluginId需要与plugin.py中的信息保持完全一致。

triggers是指用户调用插件的入口方式。

- “context-track-content”代表用户可以从轨道上右键菜单中选择运行插件。
- ” selected-clips”代表用户从素材片段右键菜单中选择运行插件。

```
{
  "plugins": [
    ......,
    {
      "providerId": "my-provider-id",
      "providerDisplayName": "我的开发者名
称",   "pluginId": "my-plugin-id",
      "pluginDisplayName": "我的插件名称",
      "version": "1.0.0",
      "minRequiredDesktopVersion": "1.8.3",
      "options": {
        "allowReset": false
      },
      "triggers": [
        {
          "type": "context-track-content",
          "config": {
            "allowedTrackTypes": ["audio"]
          }
        }
      ],
      "categories": ["generate"]
    },
    ......
  ]
}
```

https://help.tuneflow.com/zh/developer/getting-started-python.html

bundle.json示例
音频反转

Bundle.json描述记录了一些必要的插件信息。

其中providerId和pluginId需要与plugin.py中的信息保持完全一致。

triggers是指用户调用插件的入口方式。

- “context-track-content”代表用户可以从轨道上右键菜单中选择运行插件。
- ” selected-clips”代表用户从素材片段右键菜单中选择运行插件。

```
{
    "plugins": [
      {
        "providerId": "likelian",
        "providerDisplayName": {
          "en": "likelian"
        },
        "pluginId": "Reverse",
        "pluginDisplayName": {
            "en": "Reverse"
        },
        "pluginDescription": {
          "zh": " ",
          "en": " "
        },
        "triggers": [
            {
              "type": "selected-clips",
              "config": {
                "allowedClipTypes": ["audio"],
                "maxNumClips": 1
              }
            }
        ],
        "categories": ["generate"],
        "version": "0.0.1",
        "options": {
          "allowReset": false
        }
      }
    ]
}
```

https://help.tuneflow.com/zh/developer/getting-started-python.html

plugin.py

plugin.py是插件运行的主体，包括4个必备的类别方法(class method)：

- provider_id( )
- Plugin_id( )
- params( )
- run( )

```python
from tuneflow_py import TuneflowPlugin, Song, ParamDescriptor
from typing import Any, Dict


class YourPlugin(TuneflowPlugin):
    @staticmethod
    def provider_id():
        return "your_provider_id"

    @staticmethod
    def plugin_id():
        return "your_plugin_id"

    @staticmethod
    def params(song: Song) -> dict[str, ParamDescriptor]:
        #get the project data that you need
        return {}

    @staticmethod
    def run(song: Song, params: dict[str, Any]):
        #do something here...
        print("Hello World!")
```

https://help.tuneflow.com/zh/developer/getting-started-python.html

plugin.py

provider_id( )和Plugin_id( )返回与bundle.json中相同的字符。

```python
from tuneflow_py import TuneflowPlugin, Song, ParamDescriptor
from typing import Any, Dict


class YourPlugin(TuneflowPlugin):
    @staticmethod
    def provider_id():
        return "your_provider_id"

    @staticmethod
    def plugin_id():
        return "your_plugin_id"

    @staticmethod
    def params(song: Song) -> dict[str, ParamDescriptor]:
        #get the project data that you need
        return {}

    @staticmethod
    def run(song: Song, params: dict[str, Any]):
        #do something here...
        print("Hello World!")
```

https://help.tuneflow.com/zh/developer/getting-started-python.html

params( )

当用户启动插件时将首先调用params( )。

params( )用于返回所需的DAW参数和用户输入参数。同时，根据用户输入参数TuneFlow会生成所需的UI。

```python
from tuneflow_py import TuneflowPlugin, Song, ParamDescriptor
from typing import Any, Dict


class YourPlugin(TuneflowPlugin):
    @staticmethod
    def provider_id():
        return "your_provider_id"

    @staticmethod
    def plugin_id():
        return "your_plugin_id"

    @staticmethod
    def params(song: Song) -> dict[str, ParamDescriptor]:
        #get the project data that you need
        return {}

    @staticmethod
    def run(song: Song, params: dict[str, Any]):
        #do something here...
        print("Hello World!")
```

https://help.tuneflow.com/zh/developer/getting-started-python.html

params( )

参数输入来源于两个渠道：

一个是用户输入，这部分参数我们需要为它们提供 widget 值，以便提供 UI 让用户进行交互；

另一种则是直接来源于 DAW，这部分的参数在插件运行时由 DAW 直接注入到参数结果中。

```python
from tuneflow_py import TuneflowPlugin, Song, ParamDescriptor
from typing import Any, Dict


class YourPlugin(TuneflowPlugin):
    @staticmethod
    def provider_id():
        return "your_provider_id"

    @staticmethod
    def plugin_id():
        return "your_plugin_id"

    @staticmethod
    def params(song: Song) -> dict[str, ParamDescriptor]:
        #get the project data that you need
        return {}

    @staticmethod
    def run(song: Song, params: dict[str, Any]):
        #do something here...
        print("Hello World!")
```

https://help.tuneflow.com/zh/developer/python-devguide.html
https://github.com/likelian/tuneflow-plugin-demo/blob/main/Reverse/src/plugin.py

params( )

params( )需要返回一个参数名(str)到参数配置信息(ParamDescriptor)的字典。

字典中每一个参数名都会从用户或DAW中获得实际的取值，并提供给 run( ) 中的 params 参数。

```python
from tuneflow_py import TuneflowPlugin, Song, ParamDescriptor
from typing import Any, Dict


class YourPlugin(TuneflowPlugin):
    @staticmethod
    def provider_id():
        return "your_provider_id"

    @staticmethod
    def plugin_id():
        return "your_plugin_id"

    @staticmethod
    def params(song: Song) -> dict[str, ParamDescriptor]:
        #get the project data that you need
        return {}

    @staticmethod
    def run(song: Song, params: dict[str, Any]):
        #do something here...
        print("Hello World!")
```

https://help.tuneflow.com/zh/developer/getting-started-python.html

params( )

run( ) 中的输入参数 params 与 params( ) 方法返回的字典中的 key 对应。要读取一个具体的参数值，只需访问 params[keyName] 即可。

```python
class YourPlugin(TuneflowPlugin):
    ...

    @staticmethod
    def params(song: Song) -> Dict[str, ParamDescriptor]:
        return {
            "myParam": ...,
        }

    @staticmethod
    def run(song: Song, params: Dict[str, Any]):
        myParam = params["myParam"]
        ...
    ...
```


https://help.tuneflow.com/zh/developer/getting-started-python.html

params( )示例

ParamDescriptor 中几个主要值的含义：

- **displayName**： 在TuneFlow 的 UI 中显示的名称，可以是字符串。
- **defaultValue**：该参数初始值。
- **widget**：该参数在 UI 中用于收集用户输入的 UI 控件。
- **injectFrom**：从TuneFlow 注入该参数的方式。

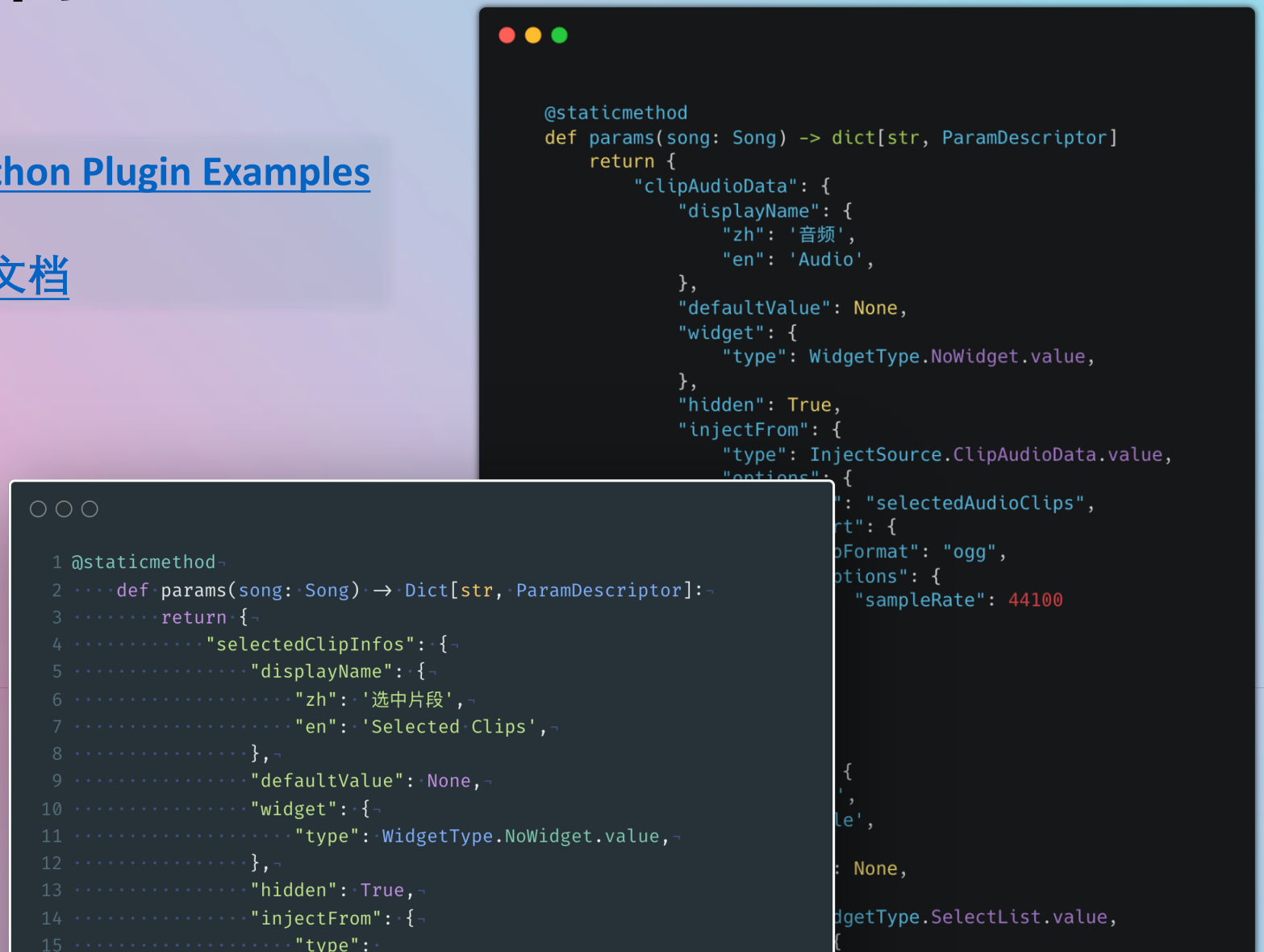
```python
@staticmethod
def params(song: Song) -> Dict[str, ParamDescriptor]:

    paramDict = {
        "selectedClipInfos": {
            "defaultValue": None,
            "widget": {
                "type": WidgetType.NoWidget.value,
            },
            "injectFrom": {
                "type": InjectSource.SelectedClipInfos.value
            }
        },
        "clipAudioData": {
            "defaultValue": None,
            "widget": {
                "type": WidgetType.NoWidget.value,
            },
            "injectFrom": {
                "type": InjectSource.ClipAudioData.value,
                "options": {
                    "clips": "selectedAudioClips",
                    "convert": {
                        "toFormat": "wav"
                    }
                }
            }
        },
    }

    return paramDict
```

https://help.tuneflow.com/zh/developer/python-devguide.html
https://github.com/likelian/tuneflow-plugin-demo/blob/main/Reverse/src/plugin.py

params( )示例

更多示例：[TuneFlow Python Plugin Examples]

详细信息参见 [param.py文档]



run( )

运行插件的目的在于修改工程文件。

开发者只需要在run( )中对 song 进行修改，TuneFlow 便会自动应用改动。

```python
from tuneflow_py import TuneflowPlugin, Song, ParamDescriptor
from typing import Any, Dict


class YourPlugin(TuneflowPlugin):
    @staticmethod
    def provider_id():
        return "your_provider_id"

    @staticmethod
    def plugin_id():
        return "your_plugin_id"

    @staticmethod
    def params(song: Song) -> dict[str, ParamDescriptor]:
        #get the project data that you need
        return {}

    @staticmethod
    def run(song: Song, params: dict[str, Any]):
        #do something here...
        print("Hello World!")
```

https://help.tuneflow.com/zh/developer/getting-started-python.html

Song数据类型

https://help.tuneflow.com/zh/developer/data-models.html

run( )

第一步，从params中获取当前选中音频片段的音频数据。

该数据来自于param( )的返回值。

```python
@staticmethod
def run(song: Song, params: Dict[str, Any]):

    ###gather the audio clip data
    clipAudioData = params["clipAudioData"] #list
    trackId = clipAudioData[0]["clipInfo"]['trackId']
    clipId = clipAudioData[0]["clipInfo"]['clipId']
    audio_format = clipAudioData[0]["audioData"]["format"]
    audio_bytes = clipAudioData[0]["audioData"]["data"]
```


https://github.com/likelian/tuneflow-plugin-demo/blob/main/Reverse/src/plugin.py

run( )

第二步，获取轨道信息和片段信息。

```python
@staticmethod
def run(song: Song, params: Dict[str, Any]):
    ...
    ###gather the track and clip info
    track = song.get_track_by_id(trackId)
    track_index = song.get_track_index(track_id=trackId)
    audio_clip = track.get_clip_by_id(clipId)
```


https://github.com/likelian/tuneflow-plugin-demo/blob/main/Reverse/src/plugin.py

run( )

第三步，将二进制的音频数据转化成numpy array。前后顺序颠倒后，再将新的numpy array记录成二进制音频格式。

```python
@staticmethod
def run(song: Song, params: Dict[str, Any]):
    ...
    #convert WAV bytes into numpy array
    input_samplerate, input_audio = read(BytesIO(audio_bytes))

    #reverse
    reversed_audio = input_audio[::-1]

    #convert numpy array into WAV bytes
    empty_bytes = bytes()
    byte_io = BytesIO(empty_bytes)
    write(byte_io, input_samplerate, reversed_audio)
    reversed_audio_bytes = byte_io.read()
```


https://github.com/likelian/tuneflow-plugin-demo/blob/main/Reverse/src/plugin.py

run( )

最后一步，将新的音频数据写入一条新建的轨道当中。

```python
@staticmethod
def run(song: Song, params: Dict[str, Any]):
    ...
    #create an empty track
    new_track = song.create_track(type=TrackType.AUDIO_TRACK,
                                  index=track_index+1)

    #add the reversed the audio clip at the exact postion
    new_track.create_audio_clip(
        clip_start_tick=audio_clip.get_clip_start_tick(),
        clip_end_tick=audio_clip.get_clip_end_tick(),
        audio_clip_data={"audio_data":
                             {"format": "wav",
                              "data": reversed_audio_bytes},
                         "duration": audio_clip.get_duration(),
                         "start_tick": audio_clip.get_clip_start_tick()
                        })
```

https://github.com/likelian/tuneflow-plugin-demo/blob/main/Reverse/src/plugin.py

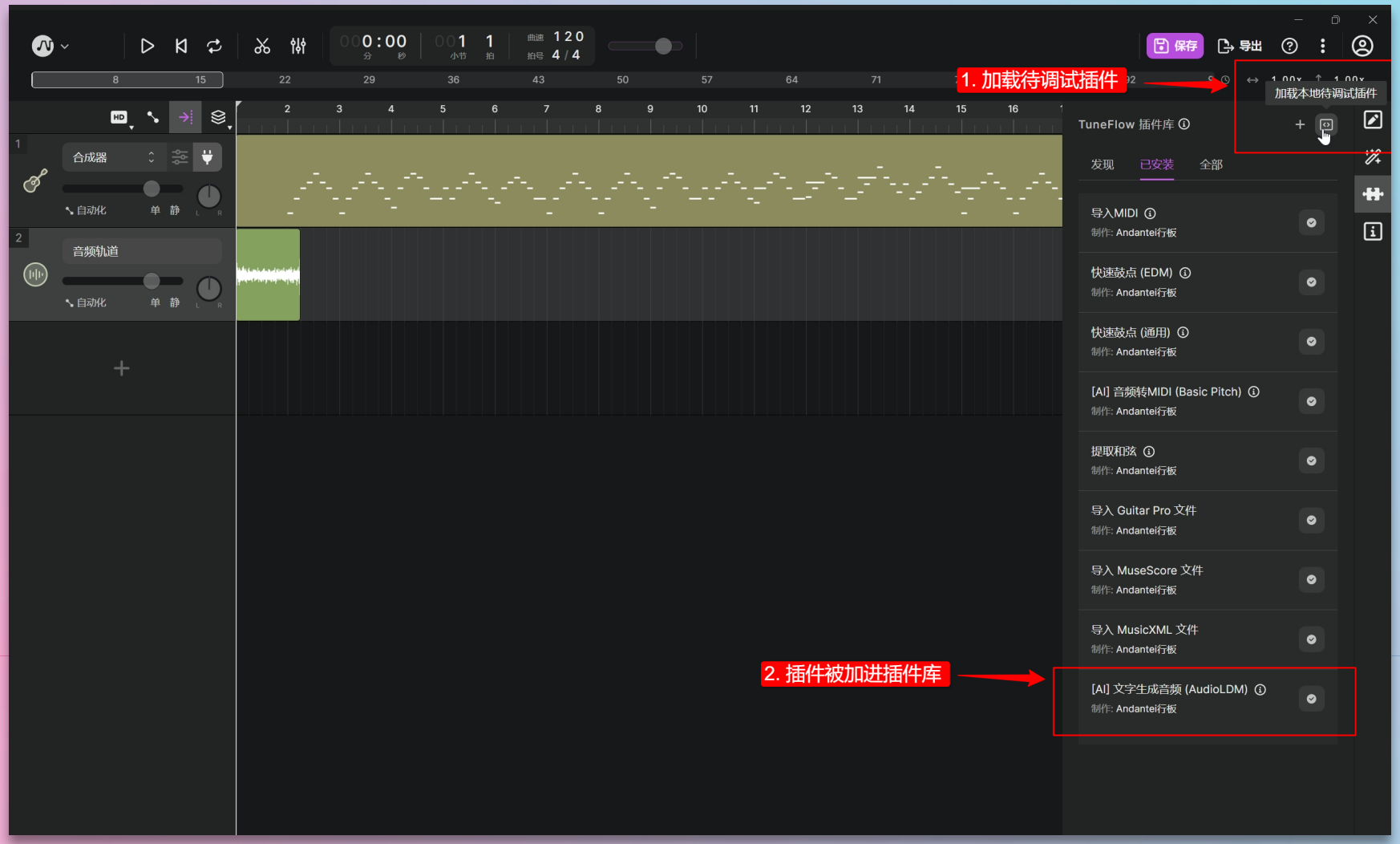
debug.py

```python
from your_plugin_path import YourPlugin
from tuneflow_devkit import Debugger

if __name__ == "__main__":
    Debugger(plugin_class=YourPlugin, bundle_file_path="your bundle.json file
path").start()
```

```
python debug.py
```


https://help.tuneflow.com/zh/developer/devkit-python.html

debug.py



https://help.tuneflow.com/zh/developer/devkit-python.html

TuneFlow

- 官方网站：www.TuneFlow.com
- 开发文档：
  - https://help.tuneflow.com/zh/
  - https://github.com/tuneflow
  - https://github.com/tuneflow/tuneflow-py
  - https://github.com/tuneflow/tuneflow-py-demos

Mike. 李克镰

Sichuan Chengdu

Scan the QR code to add me as friend