**Pygame 官方文档 - pygame.midi**

pygame.midi

用于与midi输入和输出交互的pygame模块。  
[pygame.midi.Input](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Input) - 输入用于从midi设备获取midi输入。  
[pygame.midi.MidiException](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#MidiException) - pygame.midi函数和类可以引发的异常  
[pygame.midi.Output](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output) - 输出用于将midi发送到输出设备  
[pygame.midi.get\_count](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#get_count%28%29) - 获取设备数量。  
[pygame.midi.get\_default\_input\_id](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#get_default_input_id%28%29) - 获取默认输入设备编号  
[pygame.midi.get\_default\_output\_id](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#get_default_output_id%28%29) - 获取默认输出设备编号  
[pygame.midi.get\_device\_info](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#get_device_info%28%29) - 返回有关midi设备的信息  
[pygame.midi.init](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#init%28%29) - 初始化midi模块  
[pygame.midi.midis2events](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#midis2events%28%29) - 将midi事件转换为pygame事件  
[pygame.midi.quit](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#quit%28%29) - 取消初始化midi模块  
[pygame.midi.time](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#time%28%29) - 返回PortMidi计时器的当前时间（以毫秒为单位）

midi模块可以向midi设备发送输出，并从midi设备获取输入。 它还可以列出系统上的midi设备。  
包括真正的midi设备和虚拟设备。  
它使用portmidi库。 移植到portmidi支持的平台（目前是Windows，OSX和Linux）。  
这现在使用pyportmidi，但可能在将来的某个时候使用它自己的绑定。 pygame包含pyportmidi绑定。  
pygame 1.9.0中的新功能。

pygame.midi.Input [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

输入用于从midi设备获取midi输入。  
Input(device\_id) -> None  
Input(device\_id, buffer\_size) -> None  
[pygame.midi.Input.close](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Input.close%28%29) - 关闭midi流，刷新所有挂起的缓冲区  
[pygame.midi.Input.poll](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Input.poll%28%29) - 如果有数据则返回true，否则返回false  
[pygame.midi.Input.read](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Input.read%28%29) - 从缓冲区中读取num\_events midi事件  
buffer\_size - 要缓冲等待的输入事件数

be read using Input.read()

用 Input.read() 读取

pygame.midi.Input.close() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Input)

关闭midi流，刷新所有挂起的缓冲区  
close() -> None  
当应用程序退出时，PortMidi会尝试关闭打开的流 - 这在Windows下尤其困难。  
[搜索pygame.midi.Input.close的示例](https://github.com/search?q=pygame.midi.Input.close&type=Code)

pygame.midi.Input.poll() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Input)

如果有数据则返回true，否则返回false  
poll() -> Bool  
出错时引发MidiException  
[搜索pygame.midi.Input.poll的示例](https://github.com/search?q=pygame.midi.Input.poll&type=Code)

pygame.midi.Input.read() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Input)

从缓冲区中读取num\_events midi事件  
read(num\_events) -> midi\_event\_list  
从输入缓冲区读取并返回midi事件。  
[[[status,data1,data2,data3],timestamp],

[[status,data1,data2,data3],timestamp],...]

[搜索pygame.midi.Input.read的示例](https://github.com/search?q=pygame.midi.Input.read&type=Code)

pygame.midi.MidiException [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

pygame.midi函数和类可以引发的异常  
MidiException(errno) -> None  
[搜索pygame.midi.MidiException的示例](https://github.com/search?q=pygame.midi.MidiException&type=Code)

pygame.midi.Output [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

输出用于将midi发送到输出设备  
Output(device\_id) -> None  
Output(device\_id, latency = 0) -> None  
Output(device\_id, buffer\_size = 4096) -> None  
Output(device\_id, latency, buffer\_size) -> None

[pygame.midi.Output.abort](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.abort%28%29) - 立即终止传出消息  
[pygame.midi.Output.close](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.close%28%29) - 关闭midi流，刷新任何挂起的缓冲区。  
[pygame.midi.Output.note\_off](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.note_off%28%29) - 关闭midi音符。 音符必须开启。  
[pygame.midi.Output.note\_on](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.note_on%28%29) - 打开midi音符。 音符必须关闭。  
[pygame.midi.Output.set\_instrument](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.set_instrument%28%29) - 选择一个值介于0到127之间的乐器  
[pygame.midi.Output.pitch\_bend](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.pitch_bend%28%29) - 修改频道的音高。  
[pygame.midi.Output.write](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.write%28%29) - 将midi数据列表写入输出  
[pygame.midi.Output.write\_short](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.write_short%28%29) - write\_short（status <，data1> <，data2>）  
[pygame.midi.Output.write\_sys\_ex](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output.write_sys_ex%28%29) - 写入带时间戳的系统专用midi消息。

buffer\_size指定要缓冲的输出事件的数量，等待输出。 （在某些情况下 - 见下文 - PortMidi根本不缓冲输出，只是将数据传递给较低级别的API，在这种情况下，忽略buffersize。）

延迟是应用于时间戳的延迟（以毫秒为单位），以确定何时应实际发生输出。 （如果延迟为<< 0，则假设为0。）

如果延迟为零，则忽略时间戳并立即传递所有输出。 如果延迟大于零，则输出会延迟到消息时间戳加上延迟。 （注意：时间是相对于time\_proc指示的时间源测量的。时间戳是绝对的，而不是相对延迟或偏移。）在某些情况下，PortMidi可以通过将时间戳传递给设备驱动程序或硬件来获得比应用程序更好的计时。 延迟还可以通过将midi延迟与音频缓冲延迟相匹配来帮助您将midi数据与音频数据同步。

pygame.midi.Output.abort() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

立即终止传出消息  
abort() -> None  
调用者应立即关闭输出端口; 该调用可能导致部分midi消息的传输。 Midi输入没有中止，因为用户可以随时忽略缓冲区中的消息并关闭输入设备。  
[搜索pygame.midi.Output.abort的示例](https://github.com/search?q=pygame.midi.Output.abort&type=Code)

pygame.midi.Output.close() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

关闭midi流，刷新任何挂起的缓冲区。  
close() -> None  
当应用程序退出时，PortMidi会尝试关闭打开的流 - 这在Windows下尤其困难。  
[搜索pygame.midi.Output.close的示例](https://github.com/search?q=pygame.midi.Output.close&type=Code)

pygame.midi.Output.note\_off() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

关闭midi音符。 音符必须开启。  
note\_off(note, velocity=None, channel = 0) -> None  
在输出流中关闭音符。 必须已经打开该音符才能使其正常工作。  
[搜索pygame.midi.Output.note\_off的示例](https://github.com/search?q=pygame.midi.Output.note_off&type=Code)

pygame.midi.Output.note\_on() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

打开midi音符。 音符必须关闭。  
note\_on(note, velocity=None, channel = 0) -> None  
在输出流中打开音符。 必须已关闭该音符才能使其正常工作。  
[搜索pygame.midi.Output.note\_on的示例](https://github.com/search?q=pygame.midi.Output.note_on&type=Code)

pygame.midi.Output.set\_instrument() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

选择一个乐器，其值介于0到127之间  
set\_instrument(instrument\_id, channel = 0) -> None  
[搜索pygame.midi.Output.set\_instrument的示例](https://github.com/search?q=pygame.midi.Output.set_instrument&type=Code)

pygame.midi.Output.pitch\_bend() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

修改频道的音高。  
set\_instrument(value = 0, channel = 0) -> None  
调整通道的音高。 该值是从-8192到+8191的带有符号的整数。 例如，0表示“无变化”，+ 4096通常是半音更高，-8192是1整音更低（尽管在一些合成器中也可以改变对应于弯音范围的音域）。  
如果没有给出值，则弯音返回到“无变化”。 pygame 1.9.4中的新功能。  
[搜索pygame.midi.Output.pitch\_bend的示例](https://github.com/search?q=pygame.midi.Output.pitch_bend&type=Code)

pygame.midi.Output.write() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

将midi数据列表写入输出  
write(data) -> None  
以列表的形式写入一系列MIDI信息：

write([[[status <,data1><,data2><,data3>],timestamp],

[[status <,data1><,data2><,data3>],timestamp],...])

字段是可选示例：在时间20000选择程序改变1并且在500ms之后以速度100发送音符65。

write([[[0xc0,0,0],20000],[[0x90,60,100],20500]])

注意：

1. timestamps will be ignored if latency = 0.

2. To get a note to play immediately, send MIDI info with

timestamp read from function Time.

3. understanding optional data fields:

write([[[0xc0,0,0],20000]]) is equivalent to

write([[[0xc0],20000]])

1. 如果 latency 为0，时间戳就忽略。

2. 为了得到立即播放的音符，用从函数时间读取的时间戳发送MIDI信息

3. 理解可选的数据字段：

write([[[0xc0,0,0],20000]]) is equivalent to

write([[[0xc0],20000]])

最多可以在数据列表中发送1024个元素，否则为

引发IndexError异常。

[搜索pygame.midi.Output.write的示例](https://github.com/search?q=pygame.midi.Output.write&type=Code)

pygame.midi.Output.write\_short() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

write\_short（status <，data1> <，data2>）  
write\_short(status) -> None  
write\_short(status, data1 = 0, data2 = 0) -> None  
输出3字节或更少的MIDI信息。 数据字段是可选的状态字节可以是：

0xc0 = program change

0x90 = note on

etc.

data bytes are optional and assumed 0 if omitted

数据字节是可选的，如果被忽略就假设为0

write\_short(0x90,65,100)

[搜索pygame.midi.Output.write\_short的示例](https://github.com/search?q=pygame.midi.Output.write_short&type=Code)

pygame.midi.Output.write\_sys\_ex() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#Output)

写入带时间戳的系统专用midi消息。  
write\_sys\_ex(when, msg) -> None  
msg - can be a *list* or a *string* when - a timestamp in milliseconds example:

（假设 o 是一个输出MIDI流）

o.write\_sys\_ex(0,'\xF0\x7D\x10\x11\x12\x13\xF7')

和下面等价

o.write\_sys\_ex(pygame.midi.time(),

[0xF0,0x7D,0x10,0x11,0x12,0x13,0xF7])

[搜索pygame.midi.Output.write\_sys\_ex的示例](https://github.com/search?q=pygame.midi.Output.write_sys_ex&type=Code)

pygame.midi.get\_count() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

获取设备数量。  
get\_count() -> num\_devices  
设备ID范围从0到get\_count()-1  
[搜索pygame.midi.get\_count的示例](https://github.com/search?q=pygame.midi.get_count&type=Code)

pygame.midi.get\_default\_input\_id() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

获取默认输入设备编号  
get\_default\_input\_id() -> default\_id  
如果没有设备，则返回默认设备ID或-1。 结果可以传递给Input()/ Output()类。  
在PC上，用户可以通过设置环境变量来指定默认设备。 例如，使用设备＃1。

set PM\_RECOMMENDED\_INPUT\_DEVICE=1

用户应首先使用提供的应用程序“testin”或“testout”确定可用的设备ID。

一般来说，注册表是这种信息的一个更好的地方，并且对于可以来去的USB设备，使用整数对于设备识别来说不是非常可靠。 在Windows下，如果在环境中找到PM\_RECOMMENDED\_OUTPUT\_DEVICE（或PM\_RECOMMENDED\_INPUT\_DEVICE）\* NOT \*，则通过在注册表中查找以下字符串来获取默认设备：

HKEY\_LOCAL\_MACHINE/SOFTWARE/PortMidi/Recommended\_Input\_Device

和字符串的HKEY\_LOCAL\_MACHINE / SOFTWARE / PortMidi / Recommended\_Output\_Device。 返回具有与字符串完全匹配的子字符串的第一个设备的编号。 例如，如果注册表中的字符串是“USB”，并且设备1被命名为“In USB MidiSport 1x1”，那么这将是默认输入，因为它包含字符串“USB”。

除了名称之外，get\_device\_info()还返回“interf”，这是接口名称。 （“接口”是PortMidi用于访问设备的底层软件系统或API。例如MMSystem，DirectX（未实现），ALSA，OSS（未实现）等。）目前，唯一的Win32接口是“MMSystem” “，唯一的Linux接口是”ALSA“，唯一的Max OS X接口是”CoreMIDI“。 要在注册表中指定接口和设备名称，请使用逗号和空格分隔两者，例如：

MMSystem, In USB MidiSport 1x1

在这种情况下，逗号前面的字符串必须是“interf”字符串的子字符串，并且空格后面的字符串必须是“name”名称字符串的子字符串才能匹配设备。  
注意：在当前版本中，默认值只是第一个设备（具有最低PmDeviceID的输入或输出设备）。  
[搜索pygame.midi.get\_default\_input\_id的示例](https://github.com/search?q=pygame.midi.get_default_input_id&type=Code)

pygame.midi.get\_default\_output\_id() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

获取默认输出设备编号  
get\_default\_output\_id() -> default\_id  
如果没有设备，则返回默认设备ID或-1。 结果可以传递给Input()/ Output()类。  
在PC上，用户可以通过设置环境变量来指定默认设备。 例如，使用设备＃1。

set PM\_RECOMMENDED\_OUTPUT\_DEVICE=1

用户应首先使用提供的应用程序“testin”或“testout”确定可用的设备ID。

一般来说，注册表是这种信息的一个更好的地方，并且对于可以来去的USB设备，使用整数对于设备识别来说不是非常可靠。 在Windows下，如果在环境中找到PM\_RECOMMENDED\_OUTPUT\_DEVICE（或PM\_RECOMMENDED\_INPUT\_DEVICE）\* NOT \*，则通过在注册表中查找以下字符串来获取默认设备：

HKEY\_LOCAL\_MACHINE/SOFTWARE/PortMidi/Recommended\_Input\_Device

和字符串的HKEY\_LOCAL\_MACHINE / SOFTWARE / PortMidi / Recommended\_Output\_Device。 返回具有与字符串完全匹配的子字符串的第一个设备的编号。 例如，如果注册表中的字符串是“USB”，并且设备1被命名为“In USB MidiSport 1x1”，那么这将是默认输入，因为它包含字符串“USB”。

除了名称之外，get\_device\_info（）还返回“interf”，这是接口名称。 （“接口”是PortMidi用于访问设备的底层软件系统或API。例如MMSystem，DirectX（未实现），ALSA，OSS（未实现）等。）目前，唯一的Win32接口是“MMSystem” “，唯一的Linux接口是”ALSA“，唯一的Max OS X接口是”CoreMIDI“。 要在注册表中指定接口和设备名称，请使用逗号和空格分隔两者，例如：

MMSystem, In USB MidiSport 1x1

在这种情况下，逗号前面的字符串必须是“interf”字符串的子字符串，并且空格后面的字符串必须是“name”名称字符串的子字符串才能匹配设备。

注意：在当前版本中，默认值只是第一个设备（具有最低PmDeviceID的输入或输出设备）。  
[搜索pygame.midi.get\_default\_output\_id的示例](https://github.com/search?q=pygame.midi.get_default_output_id&type=Code)

pygame.midi.get\_device\_info() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

返回有关midi设备的信息  
get\_device\_info(an\_id) -> (interf, name, input, output, opened)  
interf - 描述设备接口的文本字符串，例如’ALSA’。 name - 设备名称的文本字符串，例如’Midi Through Port-0’输入-0，如果设备是输入设备，则为1。 输出-0，如果设备是输出设备，则为1。 打开-0，如果设备打开则为1。  
如果id超出范围，则该函数返回None。  
[搜索pygame.midi.get\_device\_info的示例](https://github.com/search?q=pygame.midi.get_device_info&type=Code)

pygame.midi.init() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

初始化midi模块  
init() -> None  
在使用midi模块之前调用初始化函数。  
调用多次是安全的。  
[搜索pygame.midi.init的示例](https://github.com/search?q=pygame.midi.init&type=Code)

pygame.midi.midis2events() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

将midi事件转换为pygame事件  
midis2events(midis, device\_id) -> [Event, …]  
采用一系列midi事件并返回pygame事件列表。  
[搜索pygame.midi.midis2events的示例](https://github.com/search?q=pygame.midi.midis2events&type=Code)

pygame.midi.quit() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

取消初始化midi模块  
quit() -> None  
如果你不调用它，自动调用退出。  
不止一次调用此函数是安全的。  
[搜索pygame.midi.quit的示例](https://github.com/search?q=pygame.midi.quit&type=Code)

pygame.midi.time() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88362225#title)

返回PortMidi计时器的当前时间（以毫秒为单位）  
time() -> time  
模块启动时，时间重置为0。  
[搜索pygame.midi.time的示例](https://github.com/search?q=pygame.midi.time&type=Code)

以上文档，自己翻译，可能有误，可参考：[pygame.midi](https://www.pygame.org/docs/ref/midi.html)