C++调用python方法及环境配置

工具: windows, VS code

C++和python使用混合编程, 有四种方式

1. C++调用python
2. 直接调用python文件并执行
3. 使用Cpython, 可以将python代码直接编程c代码
4. 使用pybind11 库

C++在VS中调用python的配置

1.1 安装python

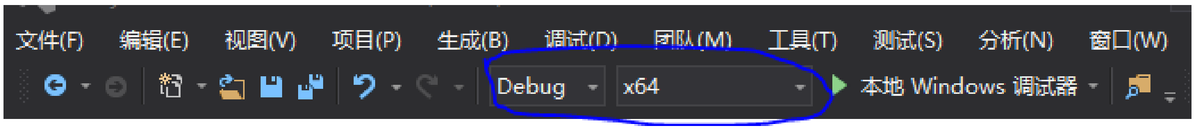
下载python, 并设置环境变量.

测试:

```
pip install numpy
pip install matplotlib
```

1.2 在vs中设置调用python

若下载python版本位64, 则在VS中将Debug修改位x64, 若为32位, 则将Debug修改位x86



然后再VS的`项目属性>配置属性>C/C++>>添加包含目录`, 将python的根目录下的include文件夹添加进行来

然后再VS的`项目属性>>配置属性>>链接器>>附件库目录`中, 将python的根目录下的libs文件夹添加进来

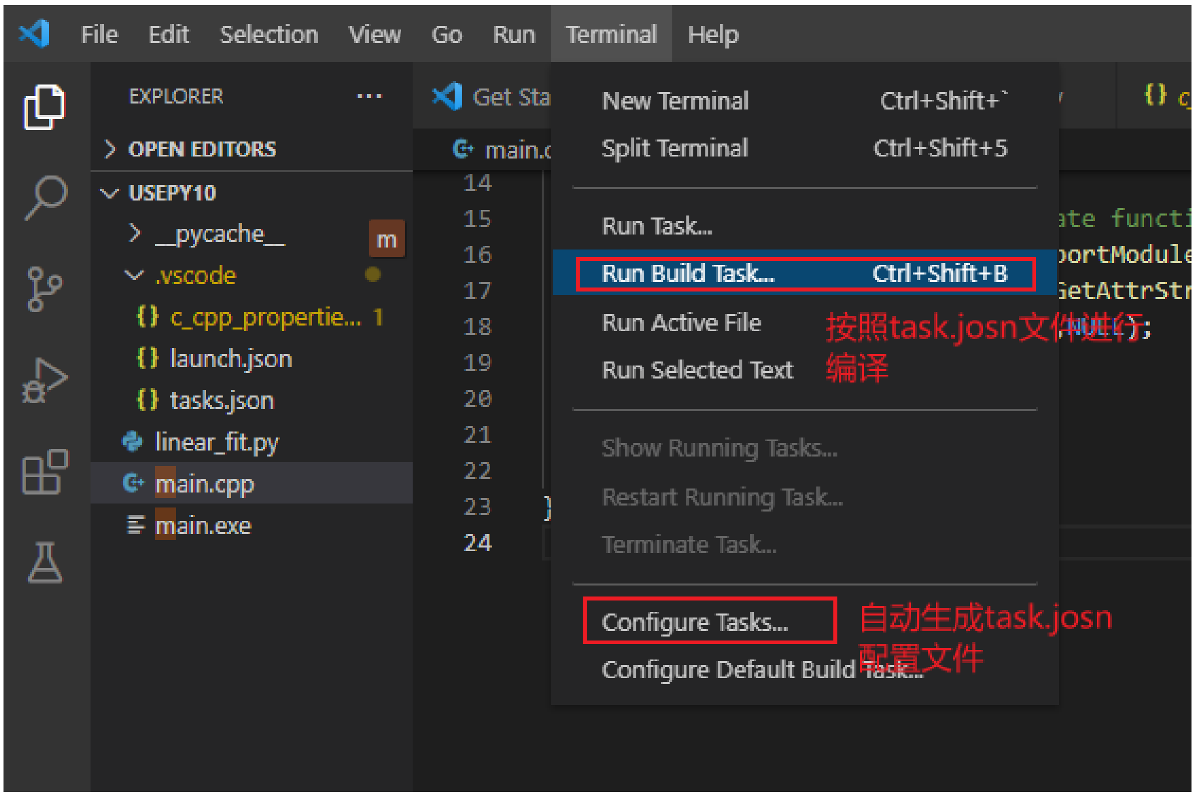
1.3 测试编译运行

在VS中新建一个文件.cpp

```
#include<Python.h>
int main()
{
    Py_Initialize();

    PyRun_SimpleString("print ('hello')");

    PyRun_SimpleString("import numpy as np");

    Py_Finalize();

    system("pause");
    return 0;
}
```

若不报错, 同时打印hello, 则表明完成.

VS code配置文件



c_cpp_properties

```
{
    "configurations": [
        {
            "name": "Win32",
            "includePath": [//需要导入的库文件路径
                "${workspaceFolder}/**",
                "D:\\Program Files\\Python37\\include",//python的库文件路径
            ],
            "defines": [
                "_DEBUG",
                "UNICODE",
                "_UNICODE"
            ],
            "compilerPath": "D:\\msys64\\mingw64\\bin\\gcc.exe",//编译器
            "cStandard": "gnu17",
            "cppStandard": "gnu++17",
            "intelliSenseMode": "windows-gcc-x64"
        }
    ],
    "version": 4
}
```

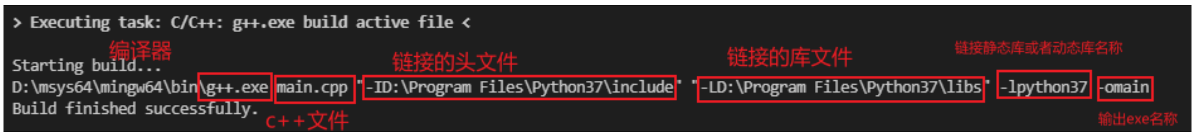
launch

```
{
    "configurations": [
        {
            "name": "Win32",
            "includePath": [
                "${workspaceFolder}/**",
                "D:\\Program Files\\Python37\\include",
            ],
            "defines": [
                "_DEBUG",
                "UNICODE",
                "_UNICODE"
            ],
            "compilerPath": "D:\\msys64\\mingw64\\bin\\gcc.exe",
            "cStandard": "gnu17",
            "cppStandard": "gnu++17",
            "intelliSenseMode": "windows-gcc-x64"
        }
    ],
    "version": 4
}
```

tasks

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: g++.exe build active file",
            "command": "D:\\msys64\\mingw64\\bin\\g++.exe",
            "args": [\*参数配置*\
                "main.cpp",
                "-ID:\\Program Files\\Python37\\include",
                "-LD:\\Program Files\\Python37\\libs",
                "-lpython37",
                "-omain"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": "build",
            "detail": "compiler: D:\\msys64\\mingw64\\bin\\g++.exe"
        }
    ]
}
```

- result

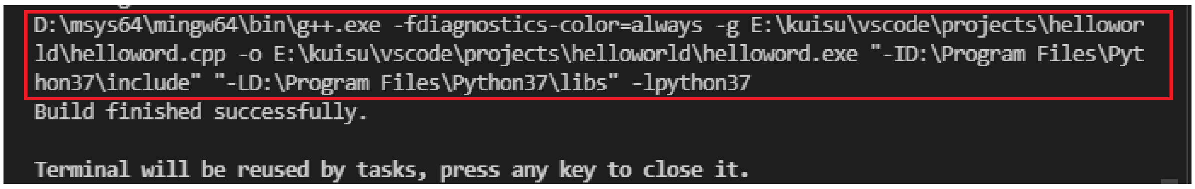


```
> Executing task: C/C++: g++.exe build active file <

Starting build...
D:\msys64\mingw64\bin\g++.exe main.cpp "-ID:\Program Files\Python37\include" "-LD:\Program Files\Python37\libs" -lpython37 -omain
Build finished successfully.
```

tasks.json2

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: cpp.exe build active file",
            "command": "D:\\msys64\\mingw64\\bin\\g++.exe",//must be g++.exe
            "args": [
                "-fdiagnostics-color=always",
                "-g","${file}",//active file current. -g must concat ${file}
                "-o",//mean output, it must concat next
                "${fileDirname}\\${fileBasenameNoExtension}.exe",//output exe of
name
                "-ID:\\Program Files\\Python37\\include",//external headr file
                "-LD:\\Program Files\\Python37\\libs",//external librarys file
                "-lpython37",//link file
            ],
            "options": {
                "cwd": "${fileDirname}"//current path to execute command
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "compiler: D:\\msys64\\mingw64\\bin\\g++.exe"//must be
g++.exe
        }
    ]
}
```

第二种表示方式



```
D:\msys64\mingw64\bin\g++.exe -fdiagnostics-color=always -g E:\kuisu\vscode\projects\hellowor
ld\helloword.cpp -o E:\kuisu\vscode\projects\helloworld\helloword.exe "-ID:\Program Files\Pyt
hon37\include" "-LD:\Program Files\Python37\libs" -lpython37
Build finished successfully.

Terminal will be reused by tasks, press any key to close it.
```

make

```
g++.exe main.cpp -ID:\Program\Python36\include -LD:\Program\Python36\libs -lpython36 -omain

#解析
-I:添加需要的外部头文件
-L:链接需要的库目录
-l:链接库目录下的动态或者静态文件
-o:输出exe的文件名
-c: 若有, 则表明生成.c二进制编译文件.
```

question

1. "error: '::hypot' has not been declared" in cmath while trying to embed python

   answer: try adding `#include <cmath>` before including Python when compiling.

   reason: error is result of `hypot` being renamed to `_hypot` in your pyconfig header file. cmath is expecting to see `hypot` and not `_hypot`

2. error: `_hypot` has not been declared in 'std' -> define hypot _hypot

   reason: due to define `hypot as _hypot` in pyconfig.h head file.

   resolve: 1) find cmath file; 2) try adding `#define _hypot hypot` in cmath head file. 3) save and run again