



Python

輸入輸出與變數

賴璉錡

lclai.t11@o365.fcu.edu.tw

安裝 **Python** 執行環境與編輯器

Anaconda

<https://www.anaconda.com/download/success>

Download Now

For installation assistance, refer to [Troubleshooting](#).

Download Anaconda Distribution or [Miniconda](#) by choosing the proper installer for your machine. Learn the difference from our [Documentation](#).



Anaconda Installers

 Download



Windows

Python 3.12

📄 64-Bit Graphical Installer (912.3M)



Mac

Python 3.12

📄 64-Bit (Apple silicon) Graphical Installer (704.7M)

📄 64-Bit (Apple silicon) Command Line Installer (707.3M)

📄 64-Bit (Intel chip) Graphical Installer (734.7M)

📄 64-Bit (Intel chip) Command Line Installer (731.2M)



Linux

Python 3.12

📄 64-Bit (x86) Installer (1007.9M)

📄 64-Bit (AWS Graviton2 / ARM64) Installer (800.6M)

📄 64-bit (Linux on IBM Z & LinuxONE) Installer (425.8M)

Anaconda Navigator & PyCharm

The screenshot displays the Anaconda Navigator desktop application. The interface includes a top menu bar with 'File' and 'Help', a left sidebar with navigation options like 'Home', 'Environments', 'Learning', and 'Community', and a main workspace area. The workspace shows a grid of application tiles, each representing a different tool available within the Navigator ecosystem. The 'PyCharm Professional' tile is highlighted with a red border. Other visible tiles include DataSpell, CMD.exe Prompt, JupyterLab, Jupyter Notebook, Powershell Prompt, Qt Console, Spyder, VS Code, Datalore, Deepnote, IBM Watson Studio Cloud, Oracle Cloud Infrastructure, anaconda-toolbox, anaconda_prompt, console_shortcut_miniconda, Glueviz, and Orange 3. Each tile provides a brief description of the tool's capabilities and a 'Launch' button.

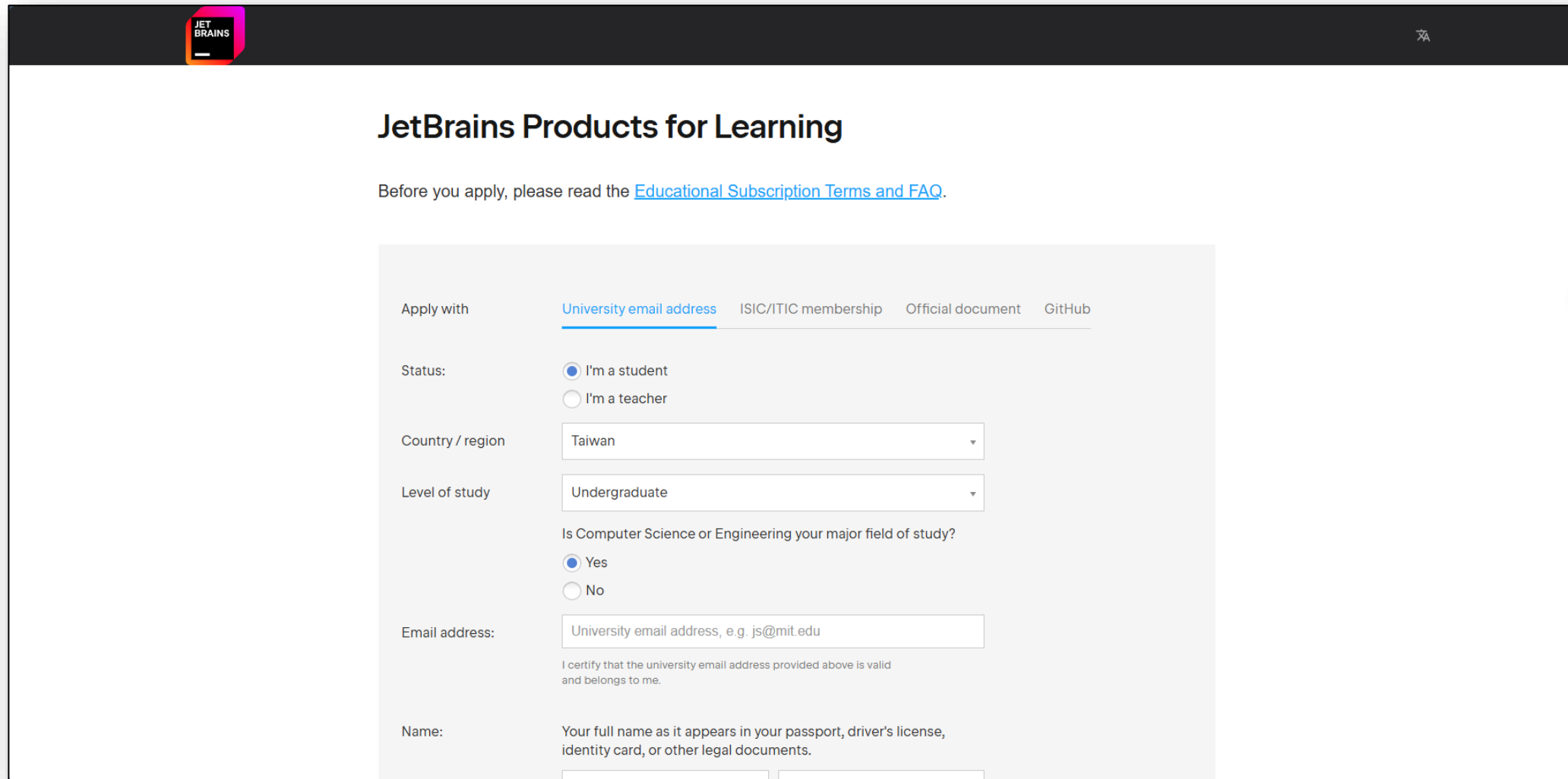
ANACONDA.NAVIGATOR Upgrade Now Connect

All applications on base (root) Channels

Application	Version	Description	Action
DataSpell		DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.	Install
CMD.exe Prompt	0.1.1	Run a cmd.exe terminal with your current environment from Navigator activated	Launch
JupyterLab	3.4.4	An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.	Launch
Jupyter Notebook	6.4.12	Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.	Launch
Powershell Prompt	0.0.1	Run a Powershell terminal with your current environment from Navigator activated	Launch
PyCharm Professional	2022.2.2	A Full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.	Launch
Qt Console	5.2.2	PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.	Launch
Spyder	5.2.2	Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features	Launch
VS Code	1.93.0	Streamlined code editor with support for development operations like debugging, task running and version control.	Launch
Datalore		Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.	Launch
Deepnote		Deepnote is a new kind of data notebook build for collaboration - Jupyter compatible, in the cloud and sharing is easy as sending a link.	Launch
IBM Watson Studio Cloud		IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.	Launch
Oracle Cloud Infrastructure		Oracle Data Science Service	
anaconda-toolbox	4.0.15		
anaconda_prompt	1.0.0		
console_shortcut_miniconda	0.1.1		
Glueviz	1.3.1		
Orange 3	3.36.2		

JetBrains Products for Learning

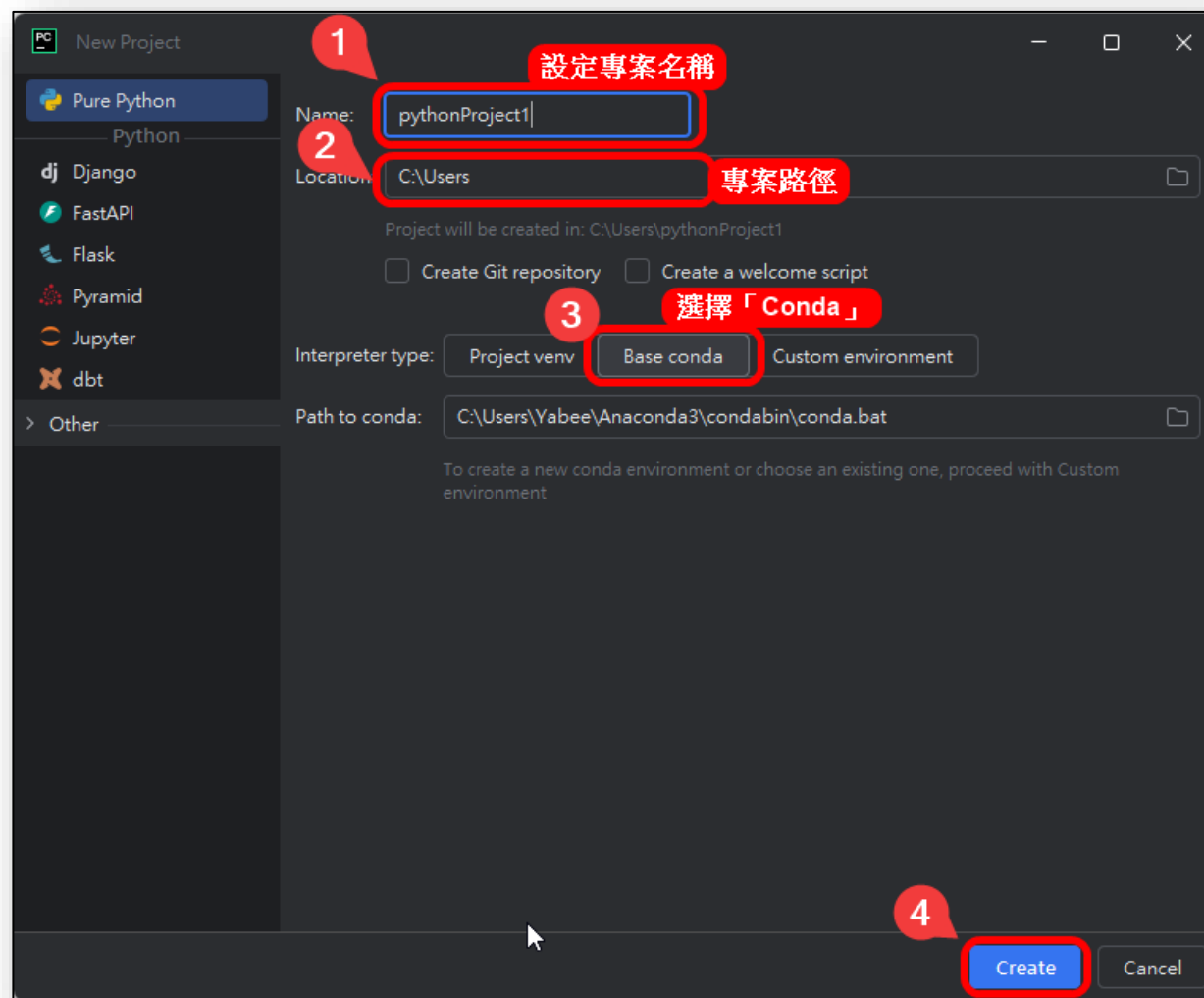
- JetBrains Products for Learning



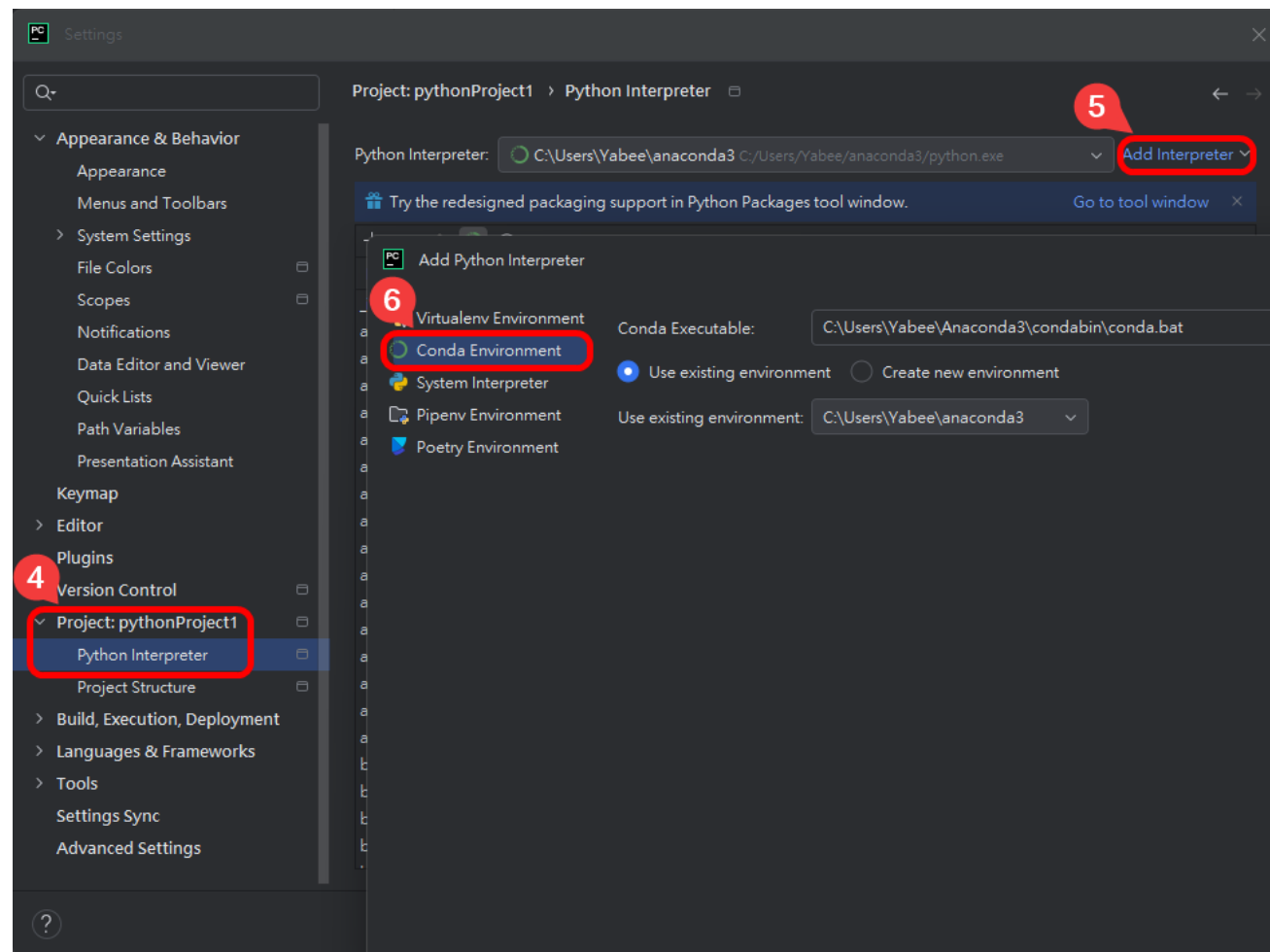
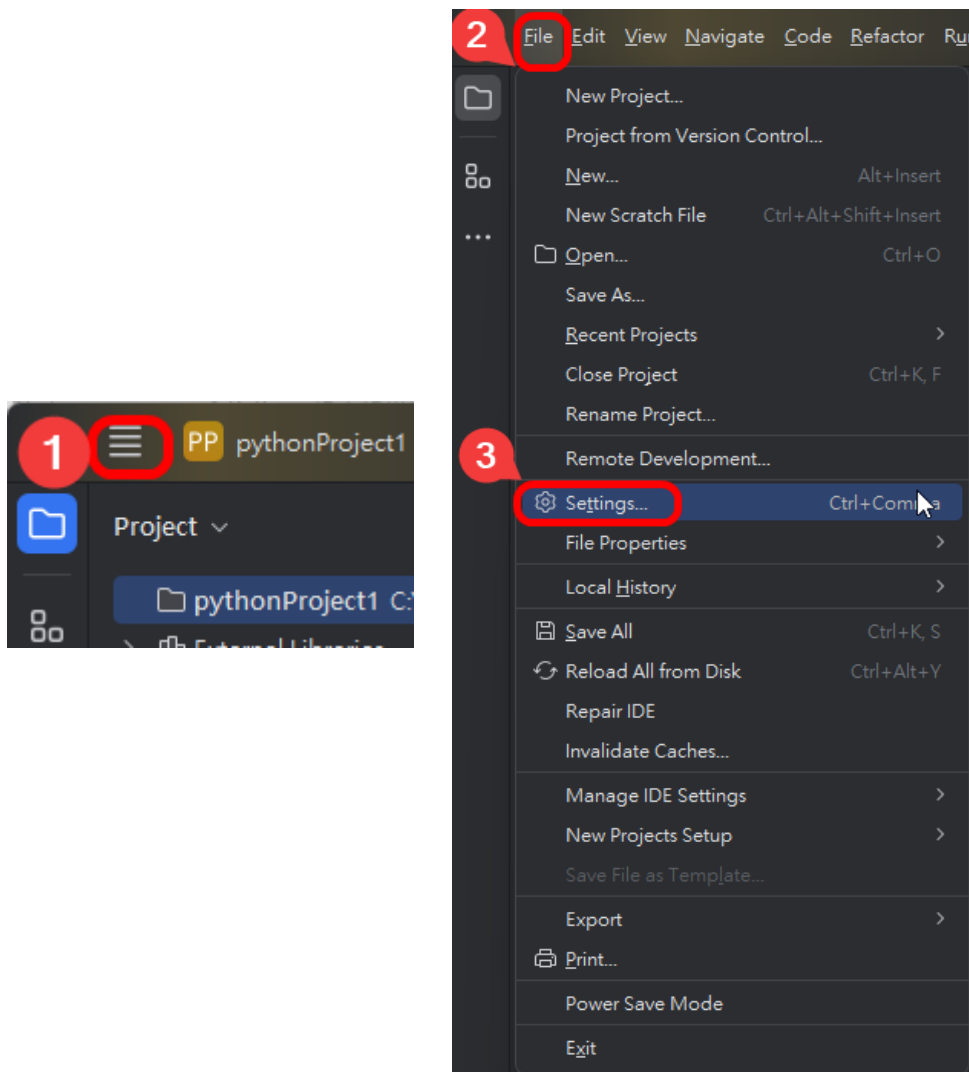
The screenshot shows the JetBrains Products for Learning application form. At the top left is the JetBrains logo, and at the top right is a language icon. The main heading is "JetBrains Products for Learning". Below it, a link to the "Educational Subscription Terms and FAQ" is provided. The form is divided into sections by tabs: "University email address" (selected), "ISIC/ITIC membership", "Official document", and "GitHub". The "University email address" section contains the following fields:

- Apply with:** University email address (selected), ISIC/ITIC membership, Official document, GitHub
- Status:** ☒ I'm a student, ☐ I'm a teacher
- Country / region:** Taiwan (dropdown menu)
- Level of study:** Undergraduate (dropdown menu)
- Is Computer Science or Engineering your major field of study?** ☒ Yes, ☐ No
- Email address:** University email address, e.g. js@mit.edu (text input)
- Confirmation:** I certify that the university email address provided above is valid and belongs to me.
- Name:** Your full name as it appears in your passport, driver's license, identity card, or other legal documents. (text input)

新增 Python 專案



Python 專案環境設定



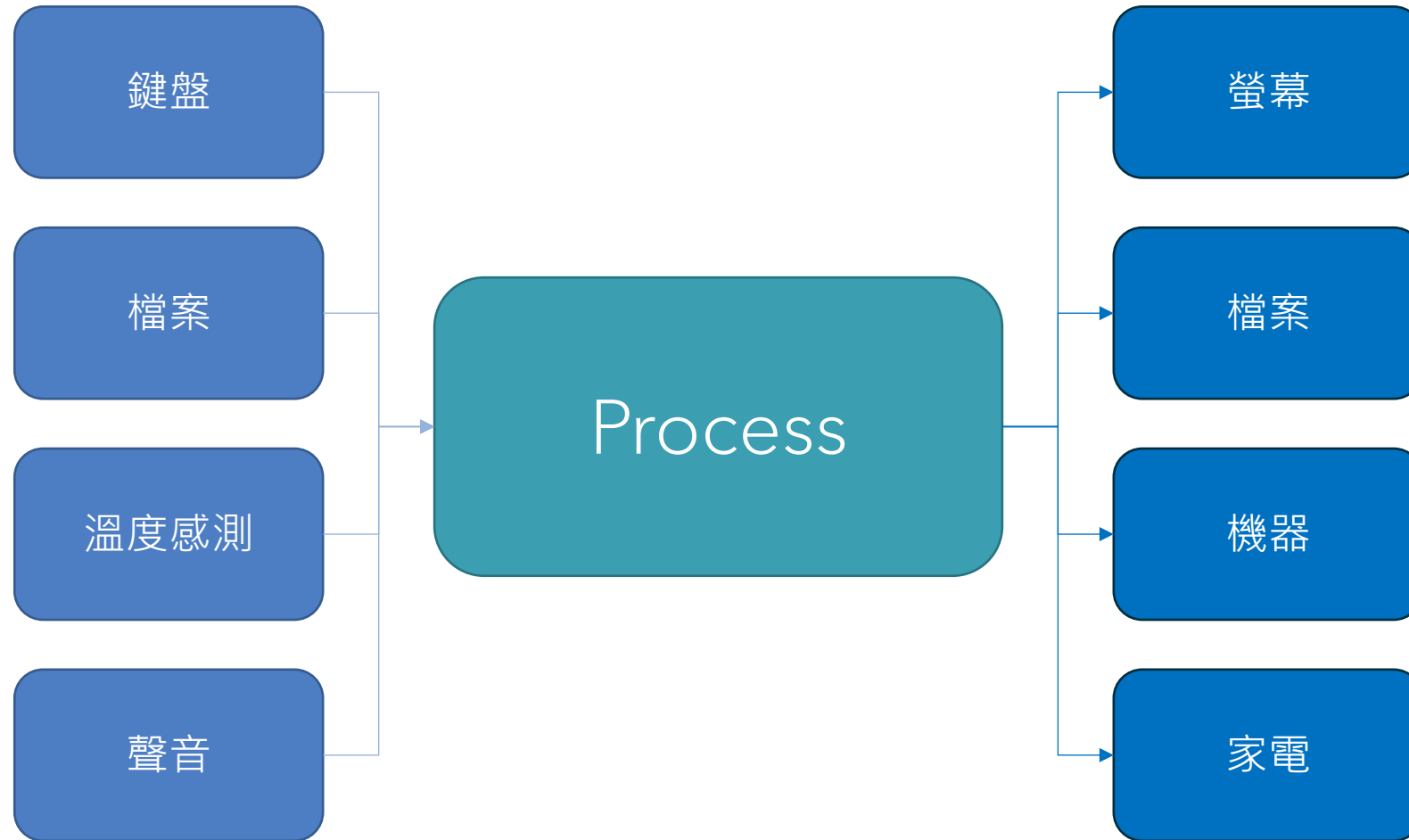
程式的基本結構

程式能做什麼呢？

- 運算： $+$ $-$ $*$ $/$
 - 演算：排序、導航、人工智慧...
- 有值才能運算 (input)
- 運算後一定要輸出 (output) 才有意義

從整體的角度來看，程式由輸入、運算、輸出所構成

Input, Process, Output



Input, Process, Output



```
eng = int(input("請輸入英文成績"))  
math = int(input("請輸入數學成績"))
```



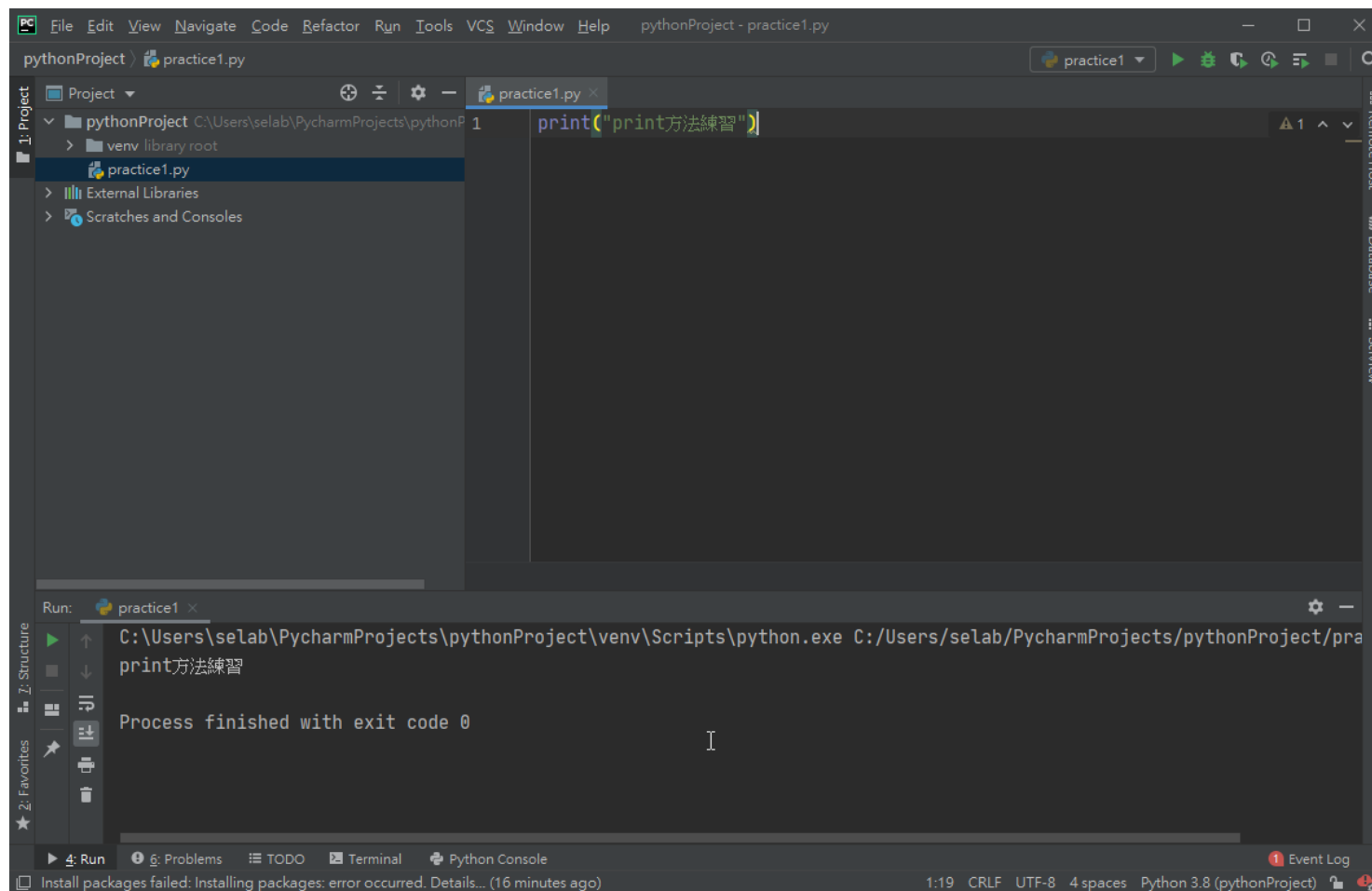
```
average = (eng + math) / 2
```



```
print("平均分數為:", average)
```

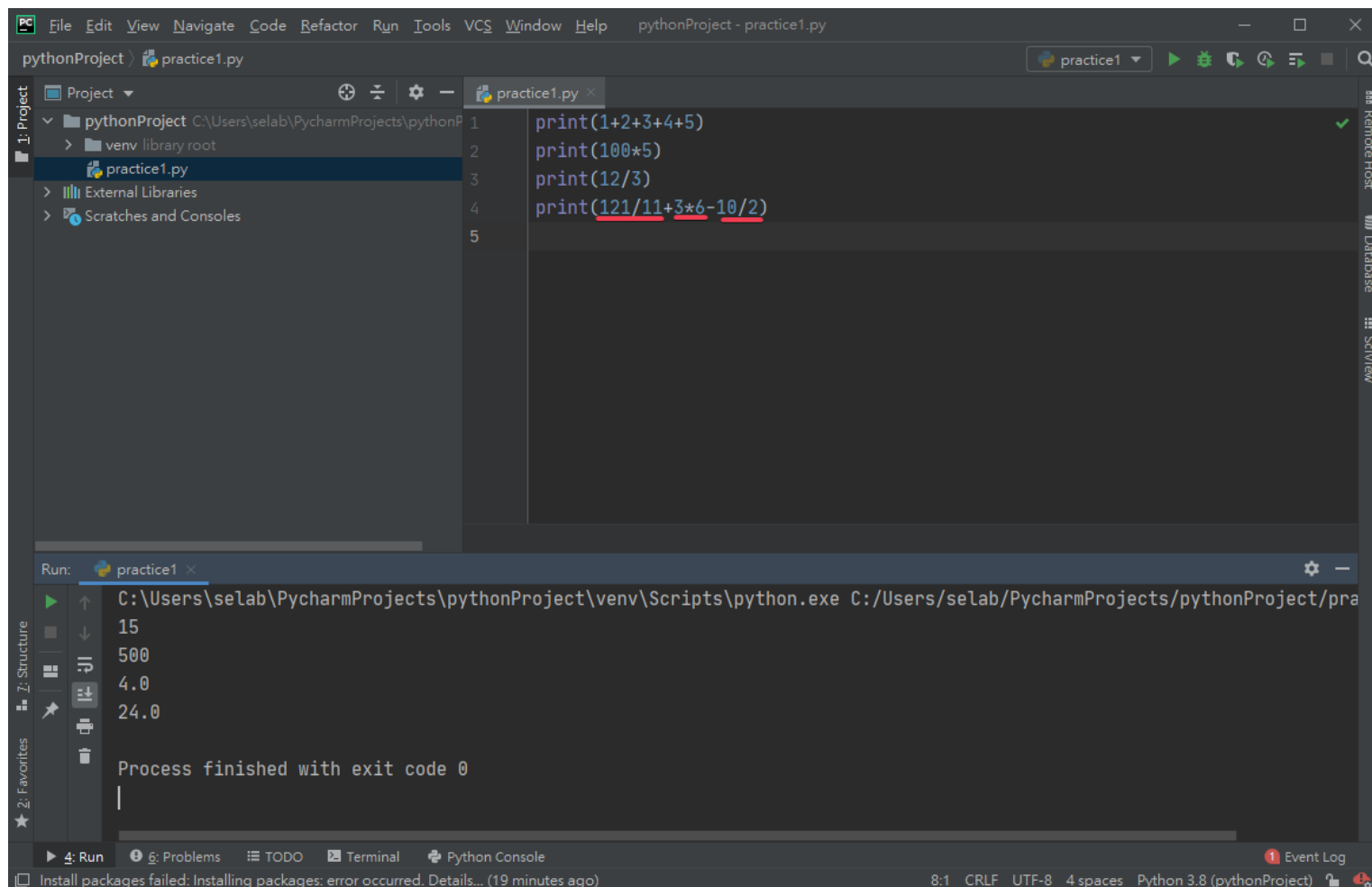
Practice: Output

- `print()`
 - Python中用來輸出內容到畫面上的方法
- 在PyCharm新增一個 `practice1.py` 檔案
- 輸入如右圖程式碼



Practice: Process

- 四則運算：+，-，*，/
 - 數字運算中最常被使用的運算子
- 修改practice1.py，輸入右圖程式碼



The screenshot shows the PyCharm IDE interface. The main editor window displays a file named `practice1.py` with the following Python code:

```
1 print(1+2+3+4+5)
2 print(100*5)
3 print(12/3)
4 print(121/11+3*6-10/2)
5
```

The bottom panel shows the Run output for the script. The command executed is `C:\Users\seLab\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/seLab/PycharmProjects/pythonProject/practice1.py`. The output is:

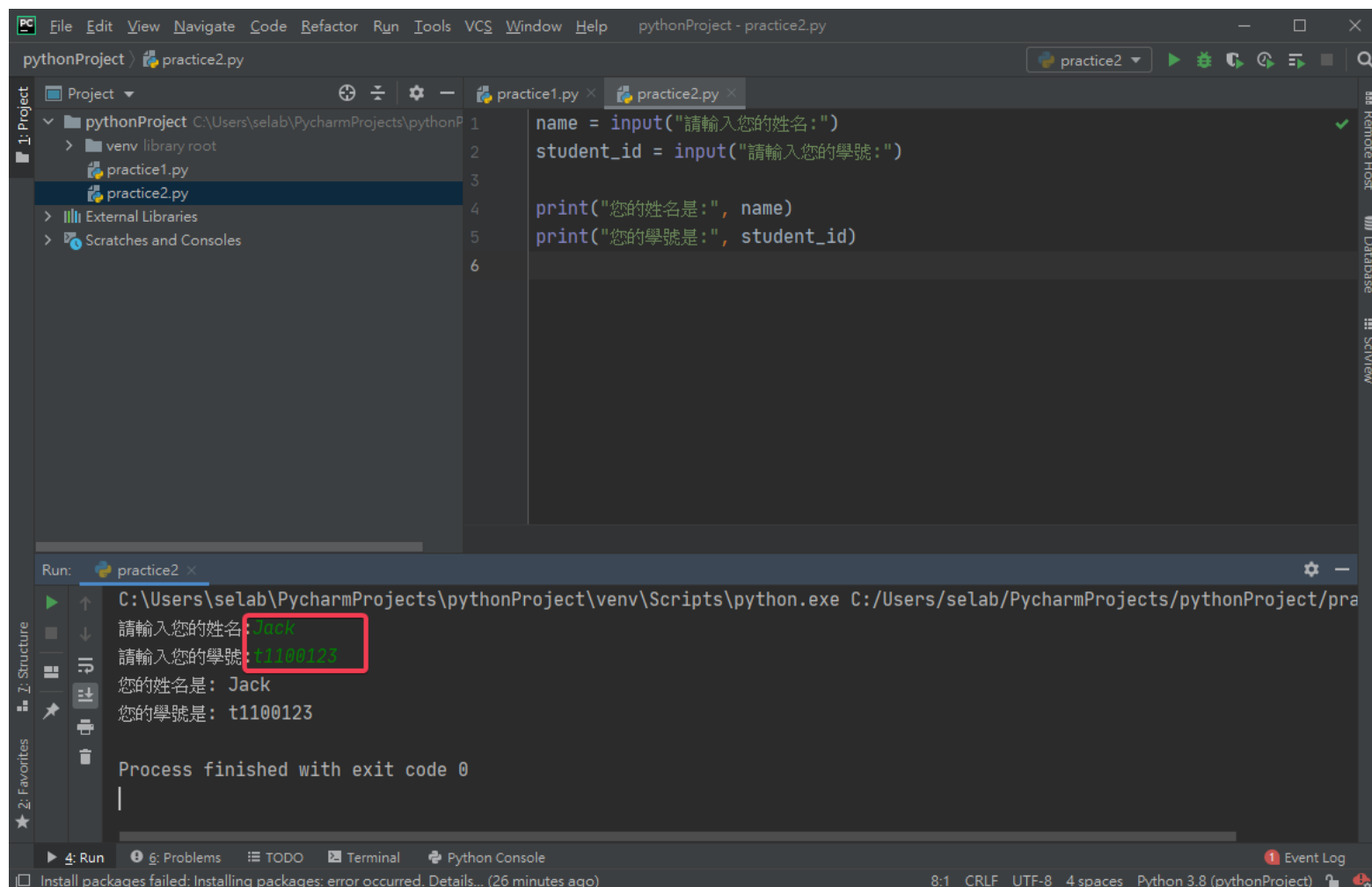
```
15
500
4.0
24.0

Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, the line ending is CRLF, and the Python version is 3.8.

Practice: Input

- `input()`
 - Python中可以讓使用者輸入內容的一個方法
 - 使用者輸入的內容均視為字串
- 在PyCharm新增一個 `practice2.py` 檔案，並輸入右圖之程式碼
- 在輸出結果的視窗中輸入姓名與學號



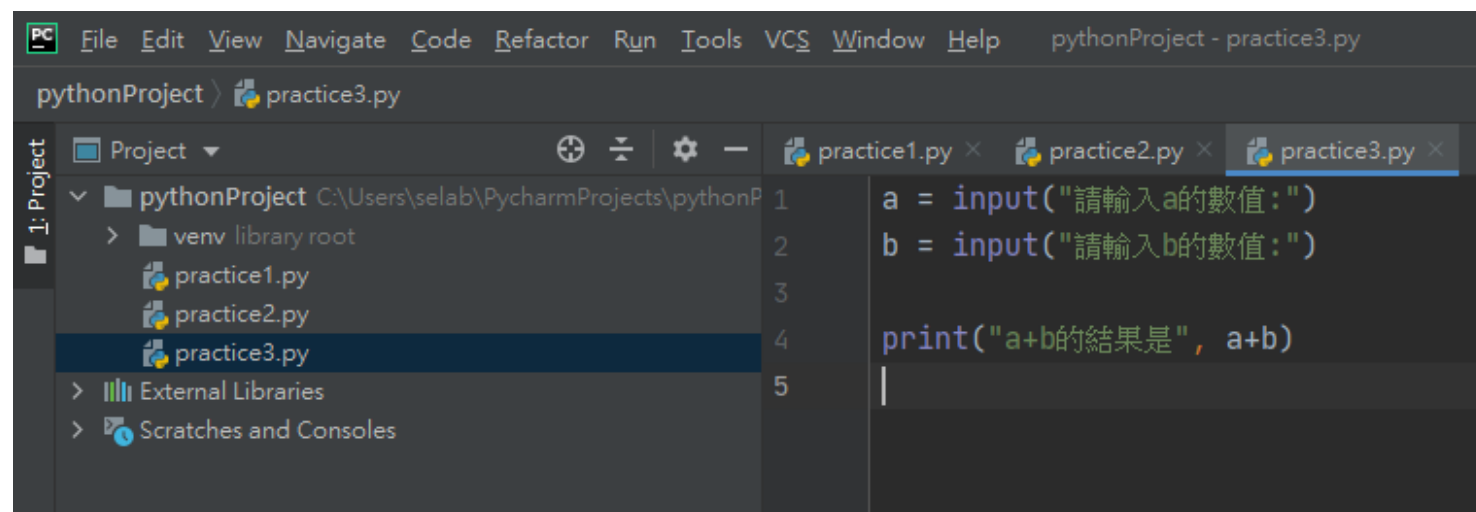
The screenshot displays the PyCharm IDE interface. The top pane shows the project structure with `pythonProject` containing `venv`, `library root`, `practice1.py`, and `practice2.py`. The `practice2.py` file is open, showing the following code:

```
1 name = input("請輸入您的姓名:")
2 student_id = input("請輸入您的學號:")
3
4 print("您的姓名是:", name)
5 print("您的學號是:", student_id)
6
```

The bottom pane shows the Run window for `practice2`. The command executed is `C:\Users\seLab\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/seLab/PycharmProjects/pythonProject/practice2.py`. The output shows the program prompts for a name and a student ID, with the inputs `Jack` and `t1100123` (highlighted by a red box) being entered. The output then displays `您的姓名是: Jack` and `您的學號是: t1100123`. The process finished with exit code 0.

Practice: Input+Process+Output

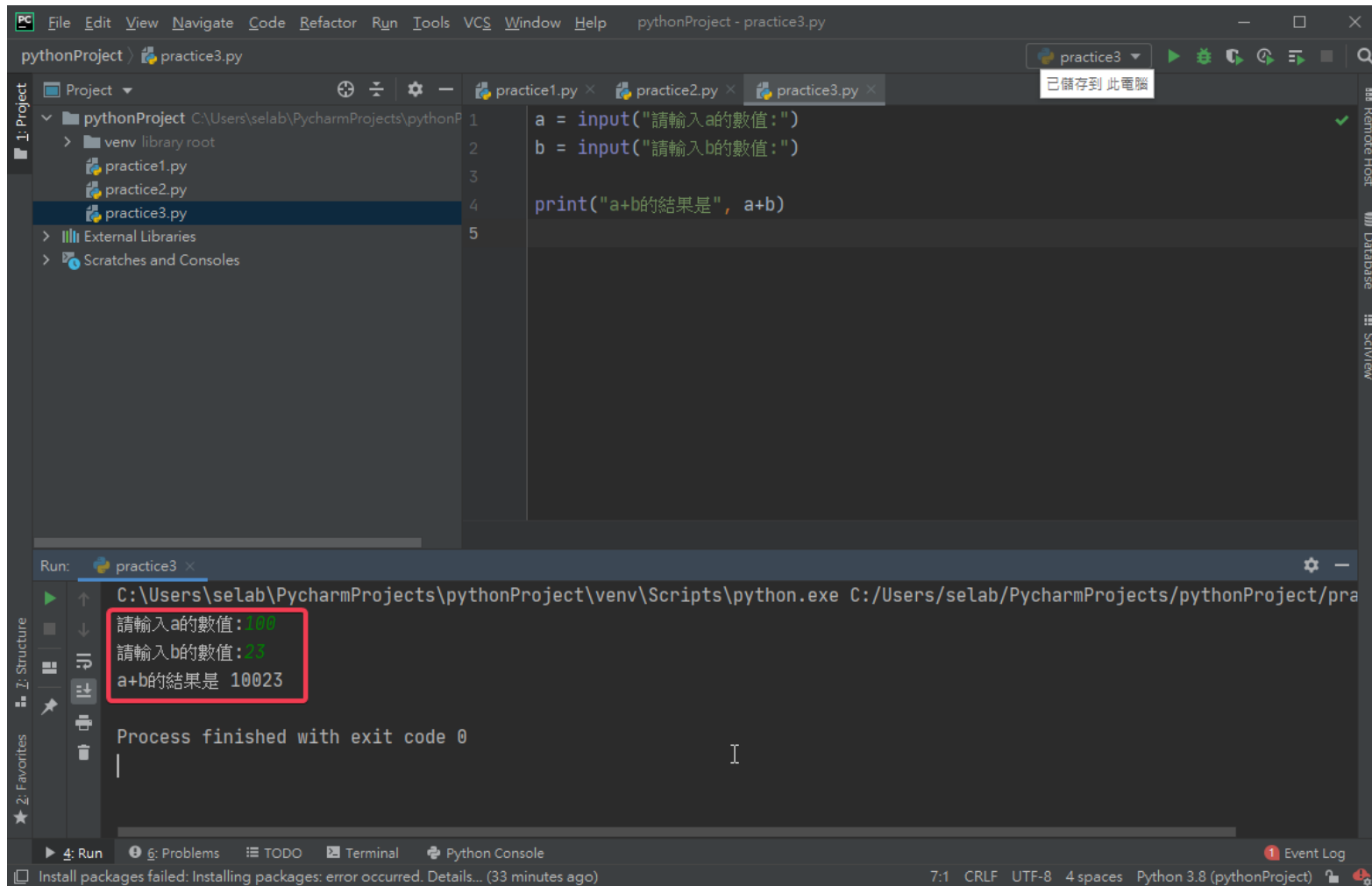
- 撰寫一個兩數相加的程式
- 新增practice3.py
- 讓使用者分別輸入兩個數a, b，計算相加的值後印出結果



The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar indicates the project is 'pythonProject - practice3.py'. The left sidebar shows the 'Project' view with a tree structure: 'pythonProject' (C:\Users\sclab\PycharmProjects\pythonP) containing 'venv library root', 'practice1.py', 'practice2.py', and 'practice3.py'. The 'practice3.py' file is selected and highlighted. The right pane shows the code editor for 'practice3.py' with the following Python code:

```
1 a = input("請輸入a的數值:")
2 b = input("請輸入b的數值:")
3
4 print("a+b的結果是", a+b)
5
```

100 + 23 = 10023?



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `practice3.py` with the following code:

```
1 a = input("請輸入a的數值:")
2 b = input("請輸入b的數值:")
3
4 print("a+b的結果是", a+b)
5
```

The left sidebar shows the project structure with `pythonProject` containing `practice1.py`, `practice2.py`, and `practice3.py`. The bottom panel shows the Run console output for `practice3`:

```
C:\Users\selab\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/selab/PycharmProjects/pythonProject/practice3.py
請輸入a的數值: 100
請輸入b的數值: 23
a+b的結果是 10023
Process finished with exit code 0
```

The input values `100` and `23` are highlighted with a red box in the console output. The status bar at the bottom indicates the Python version is 3.8 (pythonProject).

變數

- 用來儲存特定型態的值
- 變數在 Python 中不需要事先指名型態，但每個變數會根據儲存的數值來改變型態
- 使用type(變數)方法可以顯示變數的型態

```
a = 100      # int: 整數(integer)
b = 8.5      # float: 浮點數
c = "Apple"  # str: 字串(string)
d = True     # bool: 布林值(boolean)
e = False    # bool: 布林值(boolean)
```

```
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
>>> type(c)
<class 'str'>
>>> type(d)
<class 'bool'>
>>> type(e)
<class 'bool'>
```

```
# 有意義的命名
grade = 100
temperature = 8.5
fruit = "Apple"
isTeacher = True
selled = False
```

變數的型態

```
>>> a = 1
... b = 2
... print(a+b)
...
... c = "1"
... d = "2"
... print(c+d)
```

```
...
3
12
```

```
>>> fruit1 = "apple"
... fruit2 = "orange"
... print(fruit1 + fruit2)
... print(fruit1 / fruit2)
```

```
...
appleorange
```

```
Traceback (most recent call last):
```

```
  File "<input>", line 4, in <module>
```

```
TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

變數型態不同，執行結果也不同!

變數型態轉換

- 可以透過變數型態的方法強制轉換變數的型態

```
a = '10'  
print("a轉變型態之前的數值是:", a)  
print("a轉變型態之前的型態是:", type(a))  
  
a = int(a)  
print("a轉變型態之後的數值是:", a)  
print("a轉變型態之後的型態是:", type(a))
```

```
a = 'abc'  
print("a轉變型態之前的數值是:", a)  
print("a轉變型態之前的型態是:", type(a))  
  
a = int(a)  
print("a轉變型態之後的數值是:", a)  
print("a轉變型態之後的型態是:", type(a))
```

```
a轉變型態之前的數值是: 10  
a轉變型態之前的型態是: <class 'str'>  
a轉變型態之後的數值是: 10  
a轉變型態之後的型態是: <class 'int'>
```

```
a轉變型態之前的數值是: abc  
a轉變型態之前的型態是: <class 'str'>  
Traceback (most recent call last):  
  File "C:/Users/selab/PycharmProjects/pythonProject/example4.py", line 5, in <module>  
    a = int(a)  
ValueError: invalid literal for int() with base 10: 'abc'
```

Practice: 變數轉換型態

- 修改practice3.py的程式碼使其能正確執行
- 先看看使用者透過input方法輸入進來的變數型態
- 再將a、b兩個變數的型態轉換成整數型態

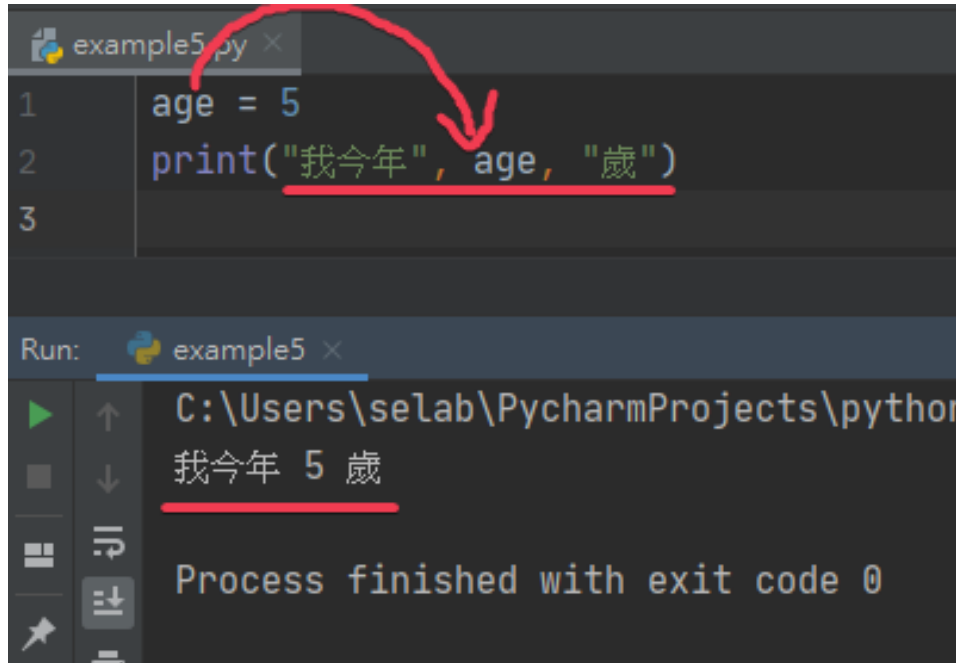
```
a = input("請輸入a的數值:")  
b = input("請輸入b的數值:")  
  
print("a的型態:", type(a))  
print("b的型態:", type(b))  
  
print("a+b的結果是", a+b)
```

```
請輸入a的數值:100  
請輸入b的數值:23  
a的型態: <class 'str'>  
b的型態: <class 'str'>  
a+b的結果是 10023
```

```
a = input("請輸入a的數值:")  
b = input("請輸入b的數值:")  
  
print("a的型態:", type(a))  
print("b的型態:", type(b))  
  
a = int(a)  
b = int(b)  
print("轉換後a的型態:", type(a))  
print("轉換後b的型態:", type(b))  
  
print("a+b的結果是", a+b)
```

```
請輸入a的數值:100  
請輸入b的數值:23  
a的型態: <class 'str'>  
b的型態: <class 'str'>  
轉換後a的型態: <class 'int'>  
轉換後b的型態: <class 'int'>  
a+b的結果是 123
```

關於print()

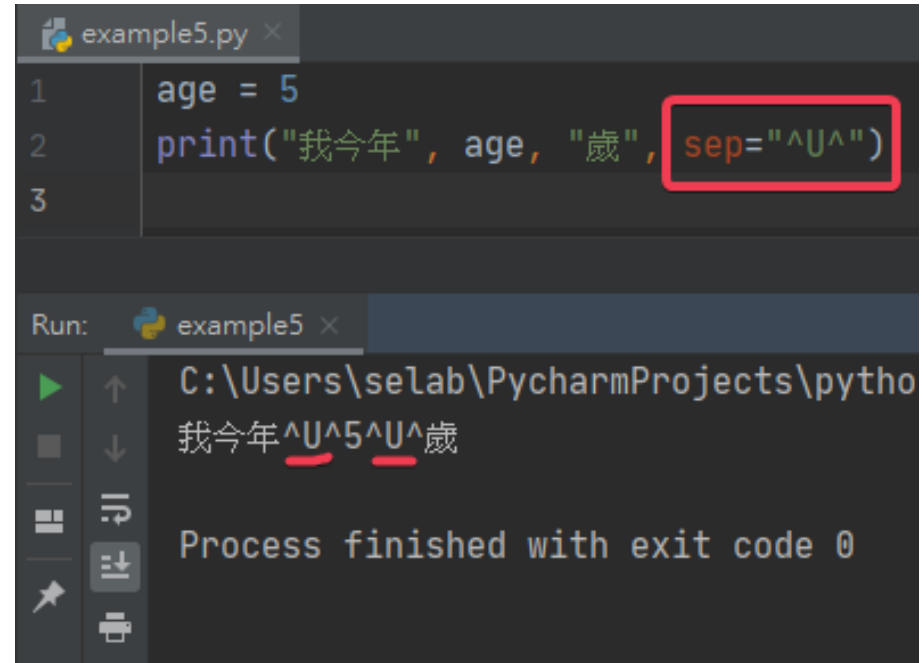


The screenshot shows a PyCharm IDE window with a file named 'example5.py'. The code in the editor is:

```
1 age = 5
2 print("我今年", age, "歲")
3
```

A red arrow points from the 'age' variable in the print statement to the output in the Run console. The Run console shows the output:

```
Run: example5 x
C:\Users\selab\PycharmProjects\pytho
我今年 5 歲
Process finished with exit code 0
```



The screenshot shows a PyCharm IDE window with a file named 'example5.py'. The code in the editor is:

```
1 age = 5
2 print("我今年", age, "歲", sep="^U^")
3
```

The 'sep="^U^"' argument is highlighted with a red box. The Run console shows the output:

```
Run: example5 x
C:\Users\selab\PycharmProjects\pytho
我今年^U^5^U^歲
Process finished with exit code 0
```

如果沒有特別設定sep，預設會以1個空格連接

關於print()

```
example5.py x
1 age = 5
2 work_year = 20
3 print("我今年", age, "歲")
4 print("我有", work_year, "年工作經驗")
5
```

Run: example5 x

C:\Users\selab\PycharmProjects\pythonP
我今年 5 歲
我有 20 年工作經驗
Process finished with exit code 0

```
example5.py x
1 age = 5
2 work_year = 20
3 print("我今年", age, "歲", sep='*')
4 print("我有", work_year, "年工作經驗", sep='#')
5
```

Run: example5 x

C:\Users\selab\PycharmProjects\pythonProject\
我今年*5*歲
我有#20#年工作經驗
Process finished with exit code 0

關於print()

```
example5.py x
1 age = 5
2 work_year = 20
3 print("我今年", age, "歲", sep='*')
4 print("我有", work_year, "年工作經驗", sep='#')
5
```

Run: example5 x

C:\Users\selab\PycharmProjects\pythonProject\

我今年*5*歲
我有#20#年工作經驗

Process finished with exit code 0

```
example5.py x
1 age = 5
2 work_year = 20
3 print("我今年", age, "歲", sep='*', end=' , ')
4 print("我有", work_year, "年工作經驗", sep='#')
5
```

Run: example5 x

C:\Users\selab\PycharmProjects\pythonProject\

我今年*5*歲 , 我有#20#年工作經驗

Process finished with exit code 0

如果沒有特別設定end，預設會以\n(換行符號)連接