电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

实验报告

EXPERIMENT REPORT

STUDENT NAME:	JAHID SHAHIDUL ISLAM
STUDENT ID:	202324090107
COURSE NAME:	PYTHON PRACTICAL PROGRAMMING
TEACHER NAME:	PROF. RAO YUNBO
EXPERIMENT NO:	ONE
DATE:	6th MAY 2024

1. Experiment title：Install Python and PyCharm

2. Experiment hours：4h Experiment location: Software Building 400

3. Objectives

   At the end of this experiment, you will be able to:

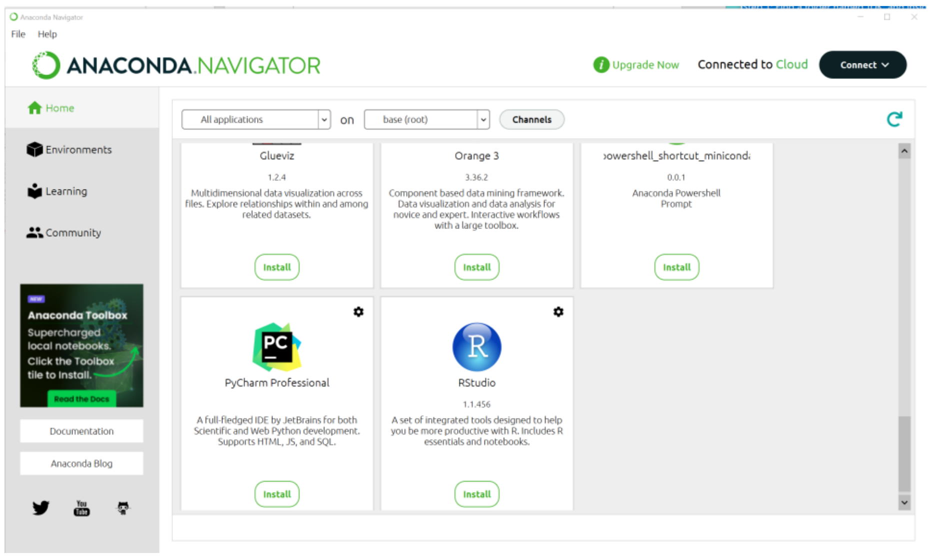
   - How to install Anaconda in your devices?
   - How to create your first python program and packaging?
   - How to install PyCharm?

4. Experimental contents & step

   1) Installing the Anaconda for Windows
   2) Install PyCharm process
   3) Create your first python project process
   4) python program packaging
   5) other

1. Installing Anaconda for Windows:

1) **Download**: Get the Anaconda installer for Windows from the official website.
2) **Launch**: Double-click the downloaded file to start the installation.
3) **Agree**: Read and accept the license agreement.
4) **Choose Install For:** Select "Just Me" unless you need it for all users.
5) **Destination**: Pick a folder in C drive without spaces or special characters for installation.
6) **Install**: Click "Install" and wait for the process to complete.
7) **Finish**: Click "Next" and "Finish" to exit the setup.



Multidimensional data visualization across files. Explore relationships within and among related datasets.

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Anaconda Powershell Prompt

A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

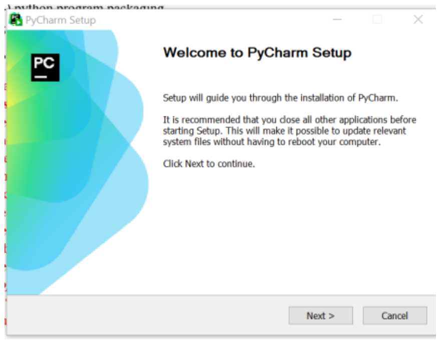
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

2. Installing PyCharm:

1) **Download**: Get the PyCharm installer (Community or Professional) from https://www.jetbrains.com/pycharm/download/.
2) **Run**: Open the downloaded file to start the installation.
3) **Next**: Click "Next" through the setup wizard, choosing options like installation location and shortcuts as desired.

4) **Install**: Click "Install" and wait for the process to finish.

5) **Finish**: Click "Finish" to exit the setup, optionally launching PyCharm.

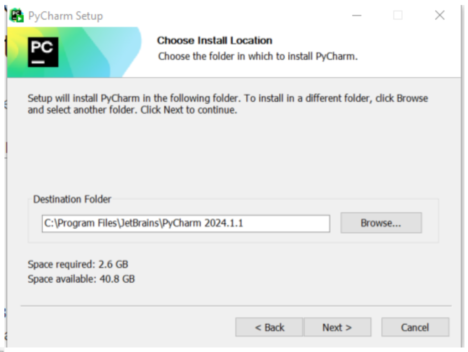

PyCharm Setup

Welcome to PyCharm Setup

Setup will guide you through the installation of PyCharm.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.


PyCharm Setup

Choose Install Location

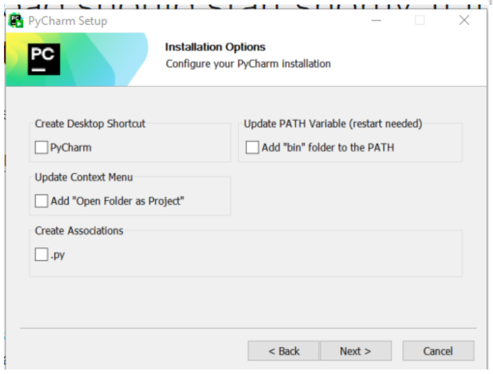
Choose the folder in which to install PyCharm.

Setup will install PyCharm in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Program Files\JetBrains\PyCharm 2024.1.1

Space required: 2.6 GB

Space available: 40.8 GB


PyCharm Setup

Installation Options

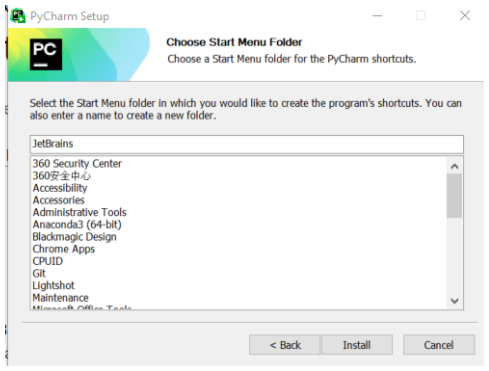
Configure your PyCharm installation

Create Desktop Shortcut: [ ] PyCharm

Update PATH Variable (restart needed): [ ] Add "bin" folder to the PATH

Update Context Menu: [ ] Add "Open Folder as Project"

Create Associations: [ ] .py


PyCharm Setup

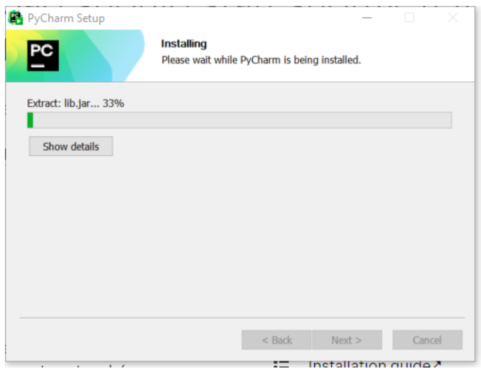
Choose Start Menu Folder

Choose a Start Menu folder for the PyCharm shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

JetBrains

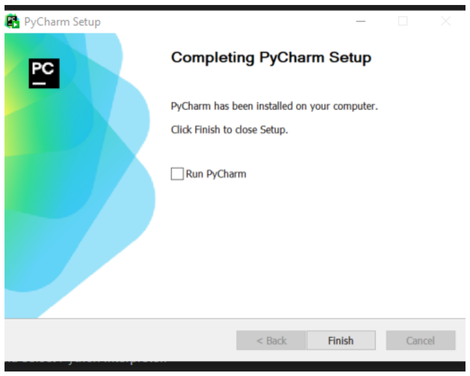
360 Security Center
360安全中心
Accessibility
Accessories
Administrative Tools
Anaconda3 (64-bit)
Blackmagic Design
Chrome Apps
CPUID
Git
Lightshot
Maintenance


PyCharm Setup

Installing

Please wait while PyCharm is being installed.

Extract: lib.jar... 33%


PyCharm Setup

Completing PyCharm Setup

PyCharm has been installed on your computer.

Click Finish to close Setup.

[ ] Run PyCharm

3. Creating Your First Python Project:

1) **Open PyCharm:** Launch the PyCharm application.

2) **New Project:** Click "Create New Project" on the welcome screen.

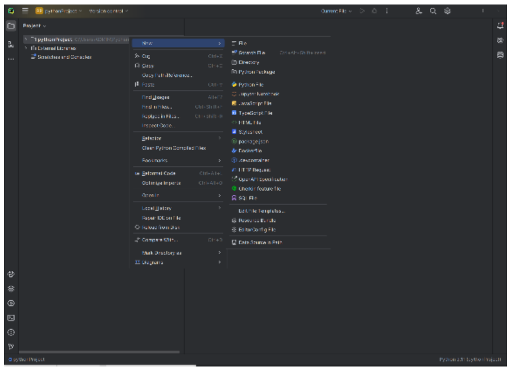
3) **Set Up:** Choose location, name, and interpreter, then click "Create".

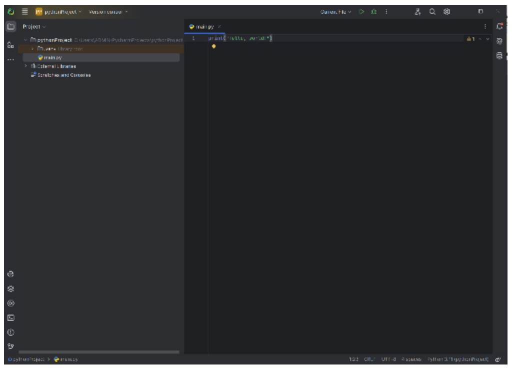
4) **Make File:** Right-click project folder -> New -> Python File (e.g., "main.py").

5) **Write Code:** Type your Python code in the file (e.g., print("Hello, world!")).

6) **Run:** Right-click in the code -> Run 'main.py'.

7) **See Results:** Check the bottom of PyCharm for your code's output.





Open Existing Python Project:

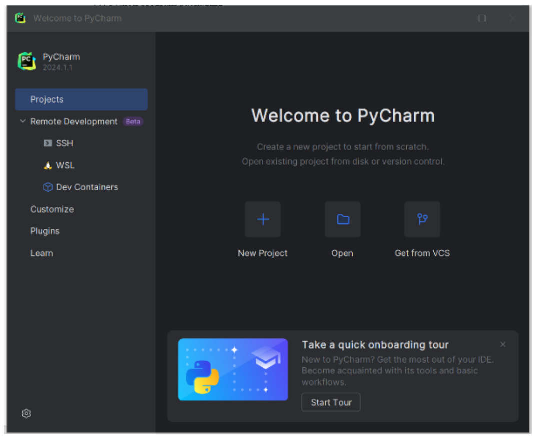
1) **Open PyCharm:** Launch the PyCharm application.

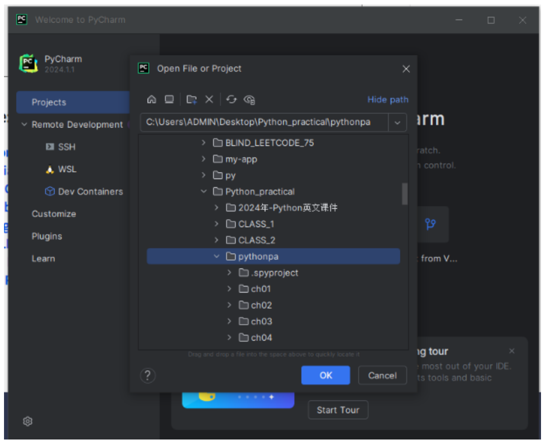
2) **Open Project:** Click "Open" on the welcome screen.

3) **Choose Folder:** Find and select your existing project folder.

4) **Open as Project:** PyCharm will recognize it as a Python project and open it.

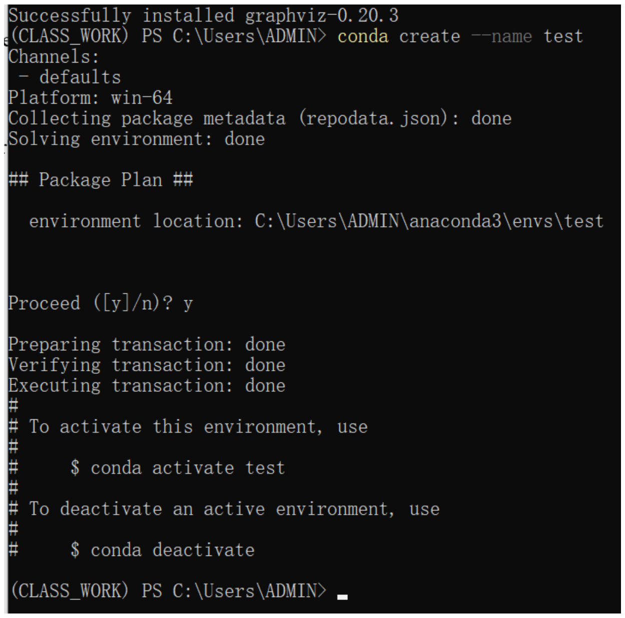
5) **Explore & Code:** You can now view your old Python files, edit them, and start coding!


Welcome to PyCharm
PyCharm
2024.1.1
Projects
Remote Development
Beta
SSH
WSL
Dev Containers
Customize
Plugins
Learn
Welcome to PyCharm
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
Take a quick onboarding tour
New to PyCharm? Get the most out of your IDE. Become acquainted with its tools and basic workflows.
Start Tour

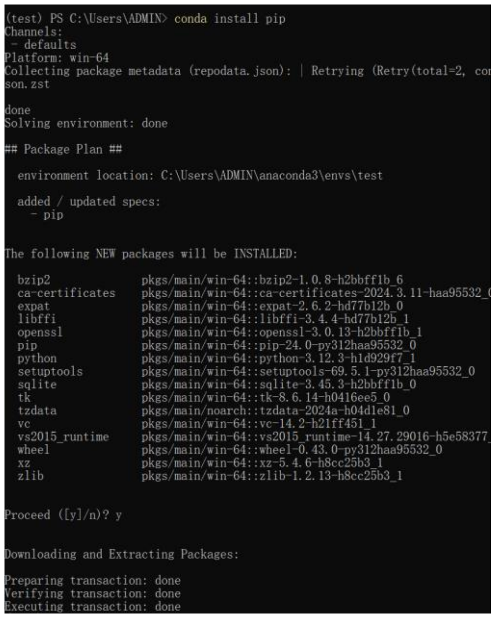


4. Packaging Python Programs with Conda:

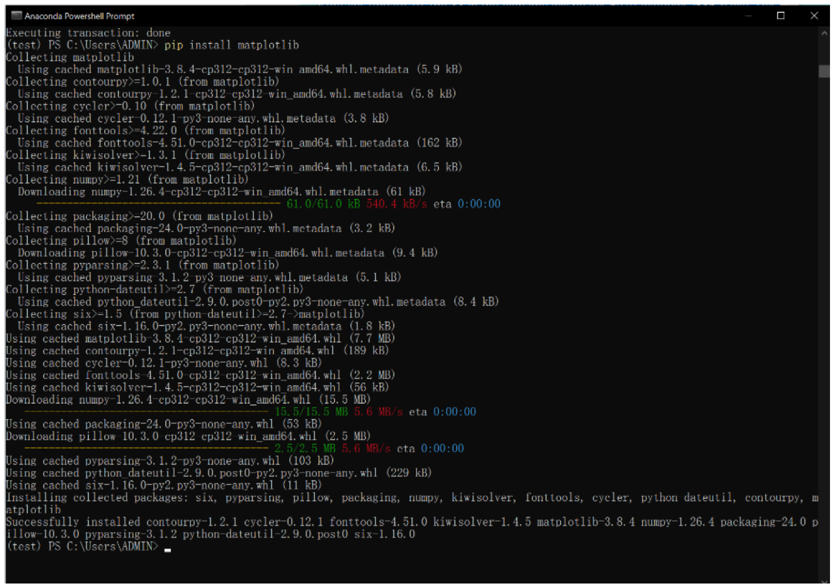
1) **Create Environment**: conda create -n test
2) **Activate:** conda activate test
3) **Install Pip (if needed):** conda install pip
4) **Install Packages:** pip install matplotlib graphviz
5) **List Packages (optional):** conda list
6) **Deactivate (optional):** conda deactivate



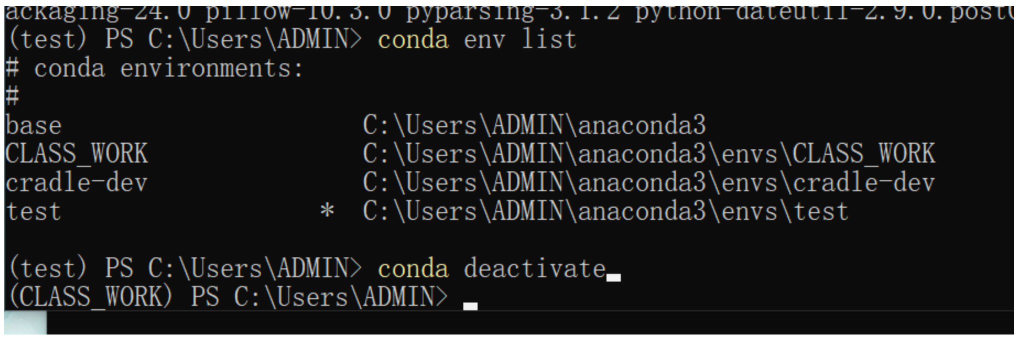
```
Successfully installed graphviz-0.20.3
(CLASS_WORK) PS C:\Users\ADMIN> conda create --name test
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\ADMIN\anaconda3\envs\test



Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate test
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(CLASS_WORK) PS C:\Users\ADMIN> _
```



```
(test) PS C:\Users\ADMIN> conda install pip
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): | Retrying (Retry(total=2, co
son.zst

done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\ADMIN\anaconda3\envs\test

  added / updated specs:
    - pip


The following NEW packages will be INSTALLED:

  bzip2              pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_6
  ca-certificates    pkgs/main/win-64::ca-certificates-2024.3.11-haa95532_
  expat              pkgs/main/win-64::expat-2.6.2-hd77b12b_0
  libffi             pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
  openssl            pkgs/main/win-64::openssl-3.0.13-h2bbff1b_1
  pip                pkgs/main/win-64::pip-24.0-py312haa95532_0
  python             pkgs/main/win-64::python-3.12.3-h1d929f7_1
  setuptools         pkgs/main/win-64::setuptools-69.5.1-py312haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
  tk                 pkgs/main/win-64::tk-8.6.14-h0416ee5_0
  tzdata             pkgs/main/noarch::tzdata-2024a-h04d1e81_0
  vc                 pkgs/main/win-64::vc-14.2-h21ff451_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377
  wheel              pkgs/main/win-64::wheel-0.43.0-py312haa95532_0
  xz                 pkgs/main/win-64::xz-5.4.6-h8cc25b3_1
  zlib               pkgs/main/win-64::zlib-1.2.13-h8cc25b3_1


Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

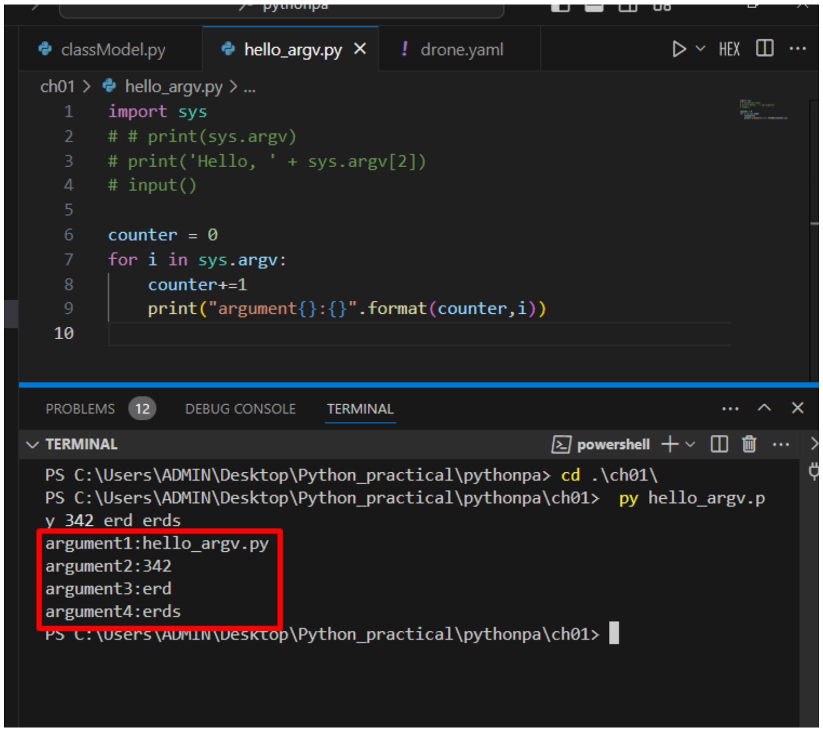


```
Executing transaction: done
(test) PS C:\Users\ADMIN> pip install matplotlib
Collecting matplotlib
  Using cached matplotlib-3.8.4-cp312-cp312-win_amd64.whl.metadata (5.9 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Using cached contourpy-1.2.1-cp312-cp312-win_amd64.whl.metadata (5.8 kB)
Collecting cycler>=0.10 (from matplotlib)
  Using cached cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Using cached fonttools-4.51.0-cp312-cp312-win_amd64.whl.metadata (162 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Using cached kiwisolver-1.4.5-cp312-cp312-win_amd64.whl.metadata (6.5 kB)
Collecting numpy>=1.21 (from matplotlib)
  Downloading numpy-1.26.4-cp312-cp312-win_amd64.whl.metadata (61 kB)
     ---------------------------------------- 61.0/61.0 kB 540.4 kB/s eta 0:00:00
Collecting packaging>=20.0 (from matplotlib)
  Using cached packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-10.3.0-cp312-cp312-win_amd64.whl.metadata (9.4 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Using cached pyparsing-3.1.2-py3-none-any.whl.metadata (5.1 kB)
Collecting python-dateutil>=2.7 (from matplotlib)
  Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.7->matplotlib)
  Using cached six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Using cached matplotlib-3.8.4-cp312-cp312-win_amd64.whl (7.7 MB)
Using cached contourpy-1.2.1-cp312-cp312-win_amd64.whl (189 kB)
Using cached cycler-0.12.1-py3-none-any.whl (8.3 kB)
Using cached fonttools-4.51.0-cp312-cp312-win_amd64.whl (2.2 MB)
Using cached kiwisolver-1.4.5-cp312-cp312-win_amd64.whl (56 kB)
Downloading numpy-1.26.4-cp312-cp312-win_amd64.whl (15.5 MB)
   ---------------------------------------- 15.5/15.5 MB 5.6 MB/s eta 0:00:00
Using cached packaging-24.0-py3-none-any.whl (53 kB)
Downloading pillow-10.3.0-cp312-cp312-win_amd64.whl (2.5 MB)
   ---------------------------------------- 2.5/2.5 MB 5.6 MB/s eta 0:00:00
Using cached pyparsing-3.1.2-py3-none-any.whl (103 kB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pyparsing, pillow, packaging, numpy, kiwisolver, fonttools, cycler, python-dateutil, contourpy, m
atplotlib
Successfully installed contourpy-1.2.1 cycler-0.12.1 fonttools-4.51.0 kiwisolver-1.4.5 matplotlib-3.8.4 numpy-1.26.4 packaging-24.0 p
illow-10.3.0 pyparsing-3.1.2 python-dateutil-2.9.0.post0 six-1.16.0
(test) PS C:\Users\ADMIN> _
```



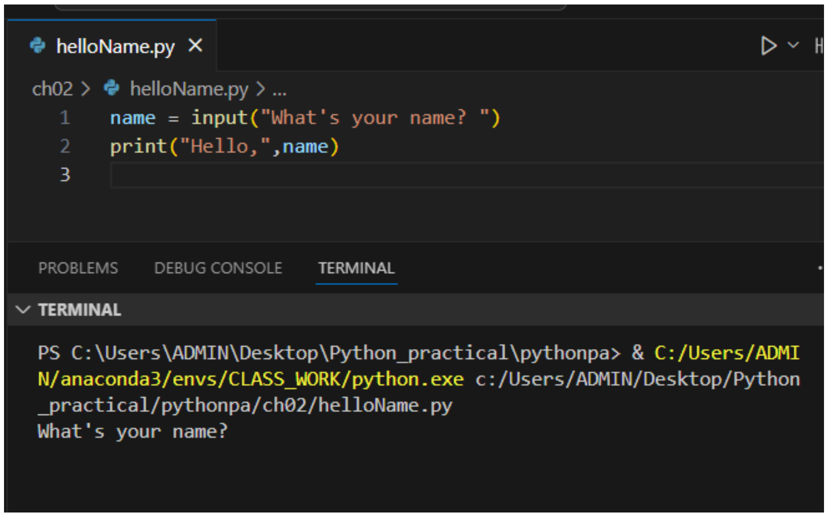
```
ackaging-24.0 pillow-10.3.0 pyparsing-3.1.2 python-dateutil-2.9.0.post
(test) PS C:\Users\ADMIN> conda env list
# conda environments:
#
base                     C:\Users\ADMIN\anaconda3
CLASS_WORK               C:\Users\ADMIN\anaconda3\envs\CLASS_WORK
cradle-dev               C:\Users\ADMIN\anaconda3\envs\cradle-dev
test                  *  C:\Users\ADMIN\anaconda3\envs\test

(test) PS C:\Users\ADMIN> conda deactivate
(CLASS_WORK) PS C:\Users\ADMIN>
```

5. Python Problems:



```python
import sys
# # print(sys.argv)
# print('Hello, ' + sys.argv[2])
# input()

counter = 0
for i in sys.argv:
    counter+=1
    print("argument{}:{}".format(counter,i))
```

```
PS C:\Users\ADMIN\Desktop\Python_practical\pythonpa> cd .\ch01\
PS C:\Users\ADMIN\Desktop\Python_practical\pythonpa\ch01>  py hello_argv.p
y 342 erd erds
argument1:hello_argv.py
argument2:342
argument3:erd
argument4:erds
PS C:\Users\ADMIN\Desktop\Python_practical\pythonpa\ch01>
```



```python
name = input("What's your name? ")
print("Hello,",name)
```

```
PS C:\Users\ADMIN\Desktop\Python_practical\pythonpa> & C:/Users/ADMI
N/anaconda3/envs/CLASS_WORK/python.exe c:/Users/ADMIN/Desktop/Python
_practical/pythonpa/ch02/helloName.py
What's your name?
```

6. Experimental Analysis:

This initial experiment was all about laying a strong foundation for our exploration of practical Python programming. Think of it as carefully setting up our workspace before embarking on a complex project. We started by installing Anaconda, a powerful toolkit that bundled together everything we need to start coding, from core Python libraries to specialized tools for data science and visualization like NumPy, SciPy, and matplotlib.

Anaconda's secret weapon, the "conda" package manager, became our trusted assistant in organizing this toolkit. With conda, we learned to create isolated project spaces, preventing conflicts and ensuring that each project had the right tools at hand. This is akin to organizing our workspace with clearly labeled compartments, ensuring everything has its place.

We also welcomed PyCharm into our workflow, a sophisticated IDE that acts like an intelligent assistant, streamlining our coding with features like code completion and error detection. Imagine having a helpful guide whispering suggestions and catching mistakes before they cause problems – that's PyCharm!

By creating a dedicated "test" environment using conda, we learned to manage project dependencies, ensuring smooth operation and reproducibility. This is crucial, like ensuring all the parts of a machine work together harmoniously. We then expanded our toolbox by installing packages like matplotlib, which will empower us to create insightful data visualizations, and graphviz, a powerful tool for representing complex relationships.

Finally, we got our hands dirty with introductory Python problems, practicing our skills with basic exercises. Think of this as testing the sharpness of our tools and getting a feel for the materials before starting a masterpiece.

This experiment has successfully equipped us with the essential tools and techniques for navigating the world of Python. We've established a solid workspace, organized our tools, and practiced our skills. Now, with confidence and excitement, we're ready to dive deeper into more advanced concepts and tackle real-world challenges with Python. Our journey has just begun, and we're eager to see what we can build!

Report score: ___________

Instructor's signature: ___________