



## DS PORTFOLIO SESSION 8

**SKILLS  
FOR LIFE**

**SKILLS BOOTCAMPS**



Department  
for Education

## Data Science Lecture Housekeeping

---

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.  
**(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes.  
You can submit these questions here: [Open Class Questions](#)

## Data Science Lecture Housekeeping cont.

---

- For all **non-academic questions**, please submit a query: [www.hyperiondev.com/support](http://www.hyperiondev.com/support)
- Report a **safeguarding** incident: [www.hyperiondev.com/safeguardreporting](http://www.hyperiondev.com/safeguardreporting)
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

# Progression Criteria

## ✓ **Criterion 1: Initial Requirements**

- Complete 15 hours of Guided Learning Hours and the first four tasks within two weeks.

## ✓ **Criterion 2: Mid-Course Progress**

- Software Engineering: Finish 14 tasks by week 8.
- Data Science: Finish 13 tasks by week 8.

## ✓ **Criterion 3: Post-Course Progress**

- Complete all mandatory tasks by 24th March 2024.
- Record an Invitation to Interview within 4 weeks of course completion, or by 30th March 2024.
- Achieve 112 GLH by 24th March 2024.

## ✓ **Criterion 4: Employability**

- Record a Final Job Outcome within 12 weeks of graduation, or by 23rd September 2024.

# Recap of Week 8: Functions

## Defining functions

- In Python, user-defined functions are instantiated using the keyword `'def'`

## Parameters

- These are variables that are defined in the function definition. They are assigned the values which were passed as arguments when the function was called, elsewhere in the code.

## Return

- The values that a function returns when it completes.


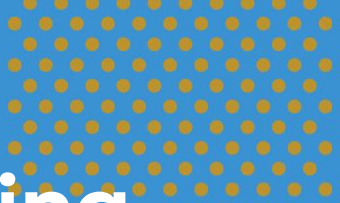
# Recap of Week 7: Functions

## Defining a Function

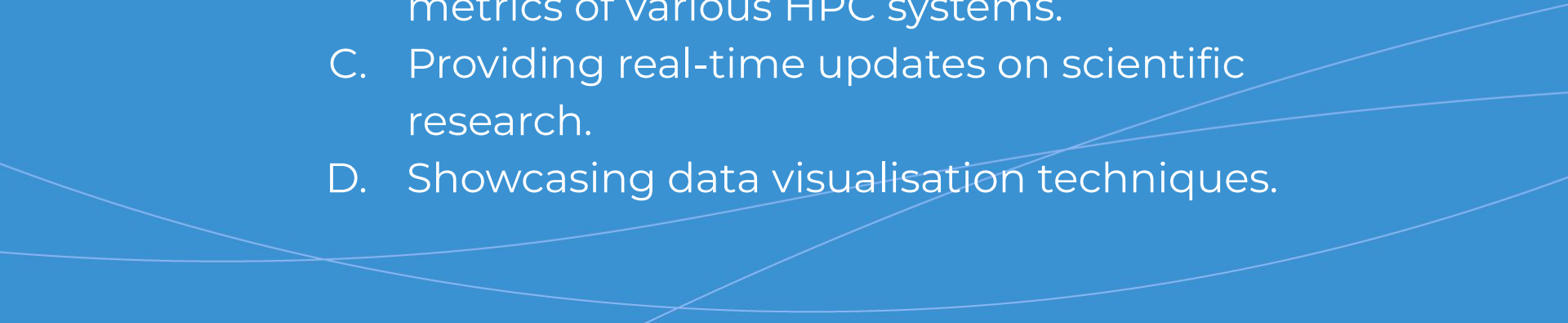
```
def add_one(x): # function called add_one
    y = x + 1
    return y
```

## Calling a function

```
result = add_one(5)
print(result)
```



# What is the purpose of using visualisations like bar graphs and line graphs?

- A. Enhancing graphics.
  - B. Comparing and contrasting the performance metrics of various HPC systems.
  - C. Providing real-time updates on scientific research.
  - D. Showcasing data visualisation techniques.
- 

# High-Performance Computing Dashboard

- **Background:** In the realm of data science, High Performance Computing (HPC) systems are the backbone that powers complex simulations, data processing, and scientific research. These systems vary in their capabilities, strengths, and speeds, making it crucial for data scientists to understand their nuances.
- **Challenge:** Create the HPC Insight Dashboard, a visualisation tool that showcases the performance metrics of these systems. Through this dashboard, users can compare and contrast the capabilities of different HPCs, making informed decisions about which system is best suited for specific tasks.



- **Objective:**

- Employ various data visualisations, such as bar graphs to compare raw performance metrics
- Line graphs to track performance over time.
- Bubble plots to visualise multidimensional data.
- Correlate findings with real-world events.

# Demo: Insight Dashboard

Here we see an example on how to our data using a simple nested dictionary.

```
# Simple example to demonstrate the need for the HPC Insight Dashboard
hpc_performance_data = {
    'HPC1': {'iteration_speed': 90, 'calculation_speed': 120},
    'HPC2': {'iteration_speed': 120, 'calculation_speed': 80},
    'HPC3': {'iteration_speed': 100, 'calculation_speed': 100}
}
```

# Demo: Insight Dashboard Continued

```
# Visualise data using bar graphs
# (Demonstration may include additional visualisations not covered in the
example)

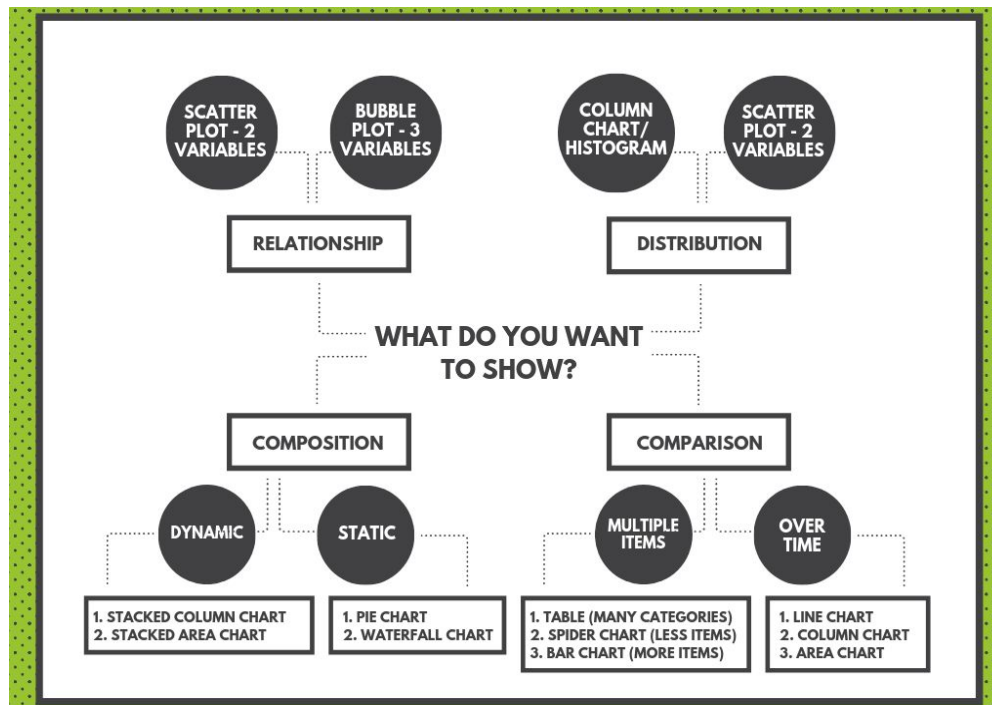
import matplotlib.pyplot as plt

hpcs = list(hpc_performance_data.keys())
iteration_speeds = [data['iteration_speed'] for data in
hpc_performance_data.values()]
calculation_speeds = [data['calculation_speed'] for data in
hpc_performance_data.values()]

plt.bar(hpcs, iteration_speeds, label='Iteration Speed')
plt.bar(hpcs, calculation_speeds, label='Calculation Speed',
bottom=iteration_speeds)

plt.xlabel('HPC Systems')
plt.ylabel('Performance Metrics')
plt.title('HPC Performance Metrics')
plt.legend()
plt.show()
```

# Deciding on Type of Visualisation



# Demo: Extending with Numpy

```
# Extended example to demonstrate additional visualisations in the HPC  
Insight Dashboard
```

```
import numpy as np
```

```
# Visualise data using line graphs and bubble plots
```

```
hpc_years = np.array([2020, 2021, 2022, 2023])
```

```
performance_data = {
```

```
    'HPC1': np.array([90, 95, 100, 105]),
```

```
    'HPC2': np.array([120, 110, 100, 90]),
```

```
    'HPC3': np.array([100, 105, 110, 115])
```

```
}
```

## Demo: Extending with Numpy (continued.)

```
plt.figure(figsize=(10, 6))

for hpc, performance in performance_data.items():
    plt.plot(hpc_years, performance, label=hpc)

bubble_sizes = [500, 1000, 1500] # Corresponding to multidimensional data

for i, hpc in enumerate(hpc_performance_data.keys()):
    plt.scatter(hpc_years[-1],
                hpc_performance_data[hpc]['calculation_speed'], s=bubble_sizes[i],
                alpha=0.5, label=hpc)

plt.xlabel('Years')
plt.ylabel('Performance Metrics')
plt.title('HPC Performance Over Time')
plt.legend()
plt.show()
```

# HPC

High Performance Computing (HPC) systems are the backbone that powers complex simulations, data processing, and scientific research. These systems vary in their capabilities, strengths, and speeds, making it crucial for data scientists to understand their nuances.

Vital libraries and concepts for this task:

|                            |
|----------------------------|
| Numpy Arrays               |
| Matplotlib (PyPlot)        |
| List Comprehension         |
| For loops to traverse data |

## Important Concepts:

1. **Decide on approach:** Implement features for visualising HPC benchmark data using various techniques.
2. **Visualisations tell a story:** Focus on meaningful interpretations of graphical representations and the correlation of data with real-world events.
3. **Error Analysis and Debugging:** Analyse the code for errors, apply debugging techniques, and optimise the HPC Insight Dashboard code.

## Advanced

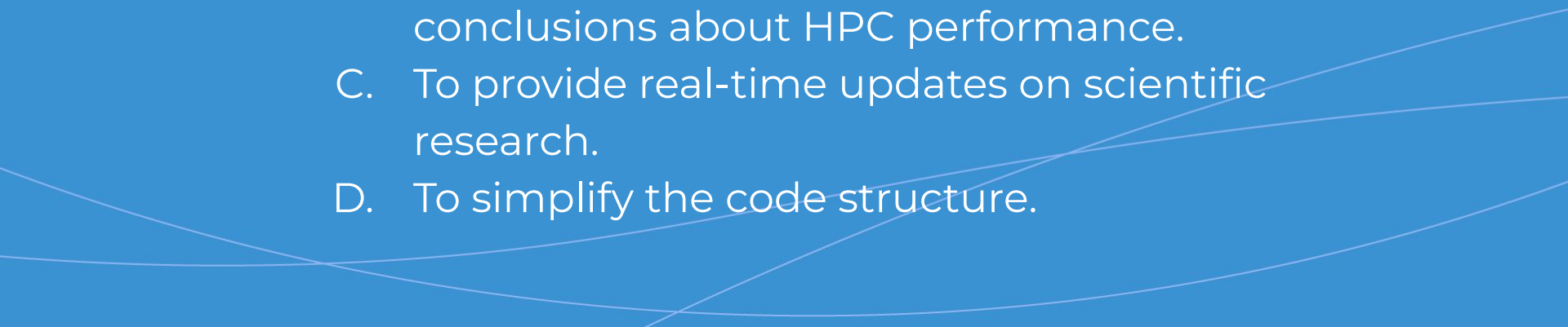
## Challenge:

- Include functionality into your program that tracks and stores historical data for comparison and visualisation purposes.



# Why is it important for the HPC Insight Dashboard to tell a story through its visualisations?



- A. To impress users with visually appealing graphics.
  - B. To help data scientists draw meaningful conclusions about HPC performance.
  - C. To provide real-time updates on scientific research.
  - D. To simplify the code structure.
- 



# Summary

---

## User-defined Functions

- ★ User-defined functions assist in encapsulating and reusing functionality in your program/code.

## Higher-order Functions

- ★ Where functions take other functions as parameters or return functions as results.



# Questions and Answers

Questions around the Case Study

