

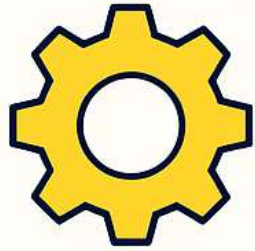
10 real-world Python interview questions

Each with clear answers and practical examples to help you shine in technical interviews, especially for roles in data, automation, and analytics.



Code With A S

Make your coding Easy

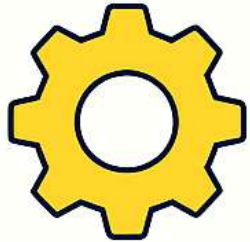


How would you handle missing data in a large dataset using Pandas?

Answer: Use `dropna()`, `fillna()`, or interpolation depending on context.

```
import pandas as pd
from datetime import datetime
df = pd.read_csv('Alice', `Bb, 'Charlie'],
'Score':[85, None, 90]
# Fill missing with mean
df['Score' = df['Score'].fillna(df['Score'].mean())
```

@Code_with_AS



How would you automate a weekly report generation process?

Answer: Use Python scripts with pandas, openpyl, and schedule with cron or schedule.

```
import pandas as pd
from datetime import datetime
df = pd.read_csv('sales.csv')
summary = df.groupby('region')['revenue'].sum()
summary.to_excel(f'report_
df['Score'] = df['Score'].fillna(datetime.now().date().xlsx')
```

@Code_with_AS



How do you extract and store data from a REST API?

Answer: Use Python scripts with pandas, openpyxl, and schedule with cron or schedule.

```
import requests as pd
from datetime import datetime
df = pd.read_csv('sales.csv')
summary = requests.get('https://apiemmple.com/data')
data = response.json()
df = pd.DataFrame(data)
df.to_csv('api_data.csv', index=False)
```

@Code_with_AS



What's the difference between shallow and deep copy?

Answer: Use *requests* to fetch, *json* to parse, and duplicates nested objects.

```
import copy
original = [[1.2], [ 3, 4]]
shallow = copy.copy(original)
deep = copy.deepcopy(original)
import copy
df.to.csv('ai_data.csv', index=False)
```

@Code_with_AS



How do you optimize a slow Python script?

Answer: Use vectorization (NumPy/Pandas), generators, or multiprocessing.

```
# Generator for memory efficiency

def read_large_file(file):

    with open(file) as f:
        for line in f:
            yield line

# Generator for memory efficiency
```

@Code_with_AS



How would you anonymize sensitive data?

Answer: Use hashing, pseudonymization, or masking.

```
# Generator for memory efficiency

import hashlib

def hash_email(email):

    return hashlib.sha256(email.encode())
    hexdigest()
```

@Code_with_AS

Decorators with a Real-World Use Case

Decorators modify function behavior—used for logging, timing, etc.

```
def logger(func):  
    def wrapper(*args, **kwargs):  
        print(f"Running {func.__name__}")  
        return func(*args, **kwargs)  
    return wrapper
```

```
@logger
```

```
def greet(name):  
    print(f"Hello, {name}!")
```

@Code_with_AS

How Do You Make Python Code Production-Ready?

Use testing (pytest), linting (flake8), type hints, and logging.

Example

```
def add(a: int, b: int) → int  
    return a + b
```

Follow @Code_with_AS



Check out my channel
Code with AS

- 👉 Python Programming Beginner to Advance Level (<https://lnkd.in/d8ysxfi5>)
- 👉 GitHub (<https://bit.ly/3ZFsW2E>)
- 👉 and YouTube (<https://bit.ly/3Jd0gss>).