American Express Data Analyst Interview Questions (0-3 Years) 15-17 LPA

Python Questions

1. What is the difference between a list, tuple, set, and dictionary in Python?

List: Ordered, mutable, allows duplicates. Example: [1, 2, 3]
Tuple: Ordered, immutable, allows duplicates. Example: (1, 2, 3)
Set: Unordered, mutable, no duplicates. Example: {1, 2, 3}
Dictionary: Key-value pairs, unordered (ordered as of Python 3.7+), mutable. Example: {'a': 1, 'b': 2}

2. How is memory managed in Python?

Uses private heap space.

Memory allocation is done by the Python memory manager.

Automatic garbage collection via gc module to manage unreferenced objects.

3. Explain Python’s GIL (Global Interpreter Lock).

Ensures only one thread executes Python bytecode at a time.
Simplifies memory management.
Limits true parallelism in multi-threaded CPU-bound tasks.

4. What are *args and kwargs in Python?

*args = non-keyword variable-length arguments (tuple).

**kwargs = keyword variable-length arguments (dict).

```
def demo(*args, **kwargs):
        print(args)
        print(kwargs)
```

5. What is list comprehension? Write a one-liner to flatten a 2D list.

Concise way to create lists using [expression for item in iterable].

Flatten 2D list: [item for sublist in matrix for item in sublist]

6. How is exception handling done in Python?

```
try:
    x = 10 / 0
except ZeroDivisionError as e:
    print("Error:", e)
finally:
    print("Cleanup done")
```

try: risky code
except: handles errors
finally: runs always

7. What are decorators in Python?

```
def decorator(func):

    def wrapper():

        print("Before function")
        func()
        print("After function")

    return wrapper

@decorator
def greet():

    print("Hello")Modify function behavior without changing code.
```

8. What is a lambda function?

Anonymous one-liner function.

add = lambda x, y: x + y

Used in map(), filter(), sorted().

```
try:
    x = 10 / 0
except ZeroDivisionError as e:
    print("Error:", e)
finally:
    print("Cleanup done")
```

try: risky code
except: handles errors
finally: runs always

9. What are generators? How is yield used?

Return iterator using yield. Saves memory.

```
def count():

    for i in range(5):
        yield i
```

10. What are generators? How is yield used?

Shallow Copy: New object, references old nested objects.

Deep Copy: Full independent copy.

```
import copy
shallow = copy.copy(obj)
deep = copy.deepcopy(obj)
```

11. Handling missing values in pandas

```
df.isnull().sum()
df.fillna(0)
df.dropna()
```

12. Difference between .loc[] and .iloc[]

.loc[]: label-based
.iloc[]: index-based

```
df.loc["row_label", "col_label"]
df.iloc[2, 3]
```

13. Group by region and get average sales

```
df.groupby('region')['sales'].mean()
```

14. Merge datasets on multiple keys

```
pd.merge(df1, df2, on=['id', 'date'], how='inner')
```

15. Explain the difference between UNION and UNION ALL.

```
Q1 = df['value'].quantile(0.25)

Q3 = df['value'].quantile(0.75)

IQR = Q3 - Q1

outliers = df[(df['value'] < Q1 - 1.5*IQR) | (df['value'] > Q3 + 1.5*IQR)]
```

16. Broadcasting in NumPy

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
arr + 10
```

Adds 10 to each element without looping.

17. Matrix multiplication

```
np.dot(A, B)
# or
A @ B
```

18. Difference: array(), zeros(), **linspace()

```
np.array([1,2,3])        # from list
np.zeros((2,2))          # matrix of zeros
np.linspace(0, 1, 5)     # [0. , 0.25, 0.5 , 0.75, 1. ]
```

19. Palindrome check

```
def is_palindrome(s):
    return s == s[::-1]
```

20. 2nd largest without sort

```
def second_largest(nums):

    first = second = float('-inf')

    for num in nums:

        if num > first:
            second, first = first, num

        elif first > num > second:
            second = num

    return second
```

21. Word frequency count

```
from collections import Counter
Counter(paragraph.lower().split())
```

22. Time complexity: set vs list search

list: O(n)
set: O(1) average case (hash-based)

23. Read/write JSON

```
import json

with open('data.json') as f:
    data = json.load(f)

with open('out.json', 'w') as f:
    json.dump(data, f)
```

24. Efficient CSV reading

```
pd.read_csv('large.csv', chunksize=10000)
```

25. Flatten nested dict

```
def flatten(d, parent_key='', sep='_'):

    items = {}

    for k, v in d.items():

        new_key = parent_key + sep + k if parent_key else k

        if isinstance(v, dict):
            items.update(flatten(v, new_key, sep))

        else:
            items[new_key] = v

    return items
```

26. Class vs static methods

```
class Demo:

    @staticmethod

    def greet(): print("Hello")

    @classmethod

    def info(cls): print(cls.__name__)
```

27. __init__, __str__, **__repr__

__init__: constructor

__str__: readable print

__repr__: unambiguous

28. Inheritance

```
class Animal:
    def sound(self): print("Generic")



class Dog(Animal):
    def sound(self): print("Bark")
```

29. Calculate ARPU

```
def arpu(transactions):

    revenue = sum(t['amount'] for t in transactions)

    users = len(set(t['user_id'] for t in transactions))

    return revenue / users
```

30. Identify churned customers

```
from datetime import datetime, timedelta

def get_churned(transactions):

    recent = datetime.now() - timedelta(days=60)

    active_users = {t['user_id'] for t in transactions if t['date'] > recent}

    all_users = {t['user_id'] for t in transactions}

    return all_users - active_users
```