Python Interview questions

1. WHAT IS THE DIFFERENCE BETWEEN PYTHON 2 AND PYTHON 3?

- **Answer**: Python 3 introduced several improvements such as better Unicode support, the print function, division behavior (`/` returns float), and more. Python 2 is no longer supported.

2. WHAT IS A PYTHON DECORATOR?

- **Answer**: A decorator is a function that wraps another function to extend or alter its behavior without changing its actual implementation.

3. WHAT ARE `*ARGS` AND `**KWARGS`?

- **Answer**: `*args` collects extra positional arguments as a tuple, and `**kwargs` collects extra keyword arguments as a dictionary in a function.

4. HOW DO YOU MANAGE MEMORY IN PYTHON?

- **Answer**: Python uses automatic memory management, primarily through reference counting and garbage collection to free objects that are no longer needed.

5. WHAT IS JSON AND HOW DO YOU HANDLE IT IN PYTHON?

- **Answer**: JSON (JavaScript Object Notation) is a lightweight data format. Use Python's `json` module to serialize (convert Python objects to JSON) and deserialize (convert JSON to Python objects).

6. WHAT ARE THE COMMON HTTP STATUS CODES?

- **Answer**:
  - 200: OK
  - 201: Created
  - 400: Bad Request
  - 401: Unauthorized
  - 404: Not Found
  - 500: Internal Server Error

7. WHAT IS THE DIFFERENCE BETWEEN `DEEPCOPY()` AND `COPY()`?

- **Answer**: `copy()` creates a shallow copy, meaning it only copies the references of nested objects, whereas `deepcopy()` creates a fully independent copy, duplicating all nested objects.

8. WHAT IS A LAMBDA FUNCTION?

- **Answer**: A lambda function is an anonymous, inline function defined using the `lambda` keyword, typically for simple operations.

9. EXPLAIN EXCEPTION HANDLING IN PYTHON.

- **Answer**: Python uses `try`, `except`, `finally`, and `else` blocks for exception handling. The `try` block contains code

that may raise exceptions, and the `except` block handles specific exceptions.

10. WHAT IS THE DIFFERENCE BETWEEN `==` AND `IS` IN PYTHON?

- **Answer**: `==` checks for value equality, while `is` checks if two variables point to the same object (identity).

11. HOW DO YOU REVERSE A LIST IN PYTHON?

- **Answer**: You can reverse a list using `list.reverse()` in-place or by creating a new list using slicing: `reversed_list = my_list[::-1]`.

12. WHAT ARE METACLASSES IN PYTHON?

- **Answer**: A metaclass defines the behavior of a class itself, meaning that a class is an instance of a metaclass. Metaclasses are used to modify class creation.

13. WHAT IS THE DIFFERENCE BETWEEN `MAP()`, `FILTER()`, AND `REDUCE()`?

- **Answer**:
  - `map()` applies a function to all items in an iterable.
  - `filter()` extracts items from an iterable based on a condition.
  - `reduce()` (in `functools`) applies a function cumulatively to the items, reducing the iterable to a single value.

14. HOW DO YOU WORK WITH JSON IN PYTHON?

- **Answer**: Use the `json` module to parse (`json.loads()`) and serialize (`json.dumps()`) JSON data.

15. WHAT ARE PYTHON'S BUILT-IN DATA TYPES?

- **Answer**: Core types include `int`, `float`, `str`, `list`, `tuple`, `dict`, `set`, `bool`, and `None`.

16. HOW DO YOU IMPLEMENT REST APIS IN FLASK?

- **Answer**: Use `Flask-RESTful` or manually create routes and views to handle HTTP methods (`GET`, `POST`, `PUT`, `DELETE`) and return JSON responses.

17. HOW DOES MEMORY MANAGEMENT WORK IN PYTHON?

- **Answer**: Python has an automatic memory manager that uses reference counting, and cyclic references are handled by a garbage collector.

18. WHAT ARE PYTHON'S BUILT-IN LIBRARIES FOR HTTP REQUESTS?

- **Answer**: The most commonly used libraries are `urllib`, `urllib2`, and `requests` (third-party).

19. WHAT ARE PYTHON'S MUTABLE AND IMMUTABLE TYPES?

- **Answer**: Mutable types (e.g., `list`, `dict`, `set`) can be changed after creation, while immutable types (e.g., `str`, `tuple`, `int`) cannot be modified once created.

20. WHAT IS A `WITH` STATEMENT?

- **Answer**: The `with` statement simplifies exception handling by ensuring that resources like file streams are properly closed. It is used with context managers.

21. HOW DO YOU HANDLE CONFIGURATION IN A PYTHON APPLICATION?

- **Answer**: Use environment variables, config files (e.g., `configparser`, `json`, or `yaml`), or a dedicated library like `dynaconf` to manage configuration.

22. WHAT IS THE DIFFERENCE BETWEEN `LIST` AND `TUPLE`?

- **Answer**: A `list` is mutable, while a `tuple` is immutable. Lists are more flexible but tuples are generally faster due to their immutability.

23. HOW DO YOU HANDLE MULTITHREADING IN PYTHON?

- **Answer**: You can use the `threading` or `concurrent.futures` modules, but Python's GIL limits true parallelism in CPU-bound tasks. For I/O-bound tasks, multithreading can still be effective.

24. HOW DOES PYTHON HANDLE DATABASES?

- **Answer**: Python can connect to databases via libraries like `sqlite3`, `psycopg2` (PostgreSQL), or `SQLAlchemy` (ORM for different databases).

25. HOW DOES PYTHON HANDLE ASYNCHRONOUS PROGRAMMING?

- **Answer**: Python provides the `asyncio` library and `async/await` syntax for asynchronous programming, which is useful for I/O-bound and non-blocking tasks.

26. WHAT ARE CLOSURES IN PYTHON?

- **Answer**: Closures are functions defined inside other functions that remember the environment in which they were created, even after the outer function has finished execution.

27. WHAT IS THE PURPOSE OF `__INIT__.PY`?

- **Answer**: The `__init__.py` file is used to mark a directory as a Python package. It can also include initialization code for the package.

28. HOW DOES PYTHON MANAGE PACKAGES?

- **Answer**: Python uses `pip` for package management, allowing developers to install, update, and remove third-party libraries.

29. HOW DO YOU HANDLE FILE I/O IN PYTHON?

- **Answer**: Use the `open()` function to read, write, or append to files, and handle files using the `with` statement to ensure they are closed after the operation.

30. HOW DO YOU IMPLEMENT CACHING IN A PYTHON APPLICATION?

- **Answer**: Use libraries like `cachetools` or `functools.lru_cache()` for in-memory caching. For distributed caching, services like Redis or Memcached can be used.

31. WHAT IS THE DIFFERENCE BETWEEN `COPY()` AND `DEEPCOPY()` IN PYTHON?

- **Answer**: `copy()` performs a shallow copy of an object, while `deepcopy()` creates a copy of the object along with all nested objects, ensuring that changes to the copy don't affect the original.

32. HOW DO YOU MERGE TWO DICTIONARIES IN PYTHON?

- **Answer**: In Python 3.9+, use:

```
python
Copy code
merged_dict = {**dict1, **dict2}
```

Or:

```
python
Copy code
dict1.update(dict2)
```

- **Answer**: Multithreading allows multiple threads to run concurrently in the same process, but the GIL limits parallel execution. Multiprocessing creates separate processes with their own memory space, allowing true parallelism.

33. HOW DO YOU IMPLEMENT LOGGING IN PYTHON?

- **Answer**: Use the built-in `logging` module to set up loggers, handlers, and formatters for logging messages to different outputs.

  ○