1.__init__

Thursday, September 26, 2024 12:18 PM

What is __init__ keyword?
Short: Initializes an object’s attributes when a class is instantiated.
Long: __init__ is a constructor method in Python that runs as soon as an object is created. It allows for initialization of the object’s attributes, taking parameters and assigning them to the object's instance variables. This is crucial for setting up the object's state.

2.self

Thursday, September 26, 2024 12:20 PM

What is self keyword?
Short: Refers to the instance of the class.
Long: The self keyword allows access to instance variables and methods within a class. It represents the current object instance and is passed implicitly as the first parameter in instance methods. Without self, Python would be unable to differentiate between instance variables and local variables.

3.lambda function

Thursday, September 26, 2024 12:20 PM

What is lambda function?
Short: A small, anonymous function.
Long: A lambda function is an anonymous function that is defined using the lambda keyword. It's used for short, inline functions that don’t need a formal definition. These functions can have multiple arguments but are limited to a single expression, and they're commonly used in places where small throwaway functions are needed.

4. Lambda and Function

Thursday, September 26, 2024 12:20 PM

Difference between lambda and normal function?
Short: Lambda is anonymous and concise, normal function is named and can be complex.
Long: A lambda function is a small, anonymous function that is limited to a single line of code and has no name, while a normal function is defined using def, can have a name, and support multiple lines, complex logic, and more features like docstrings. Lambda functions are generally used for quick, throwaway tasks like filtering or mapping.

5.Generators

Thursday, September 26, 2024 8:53 PM

What are generators? When to use?
Short: Functions that yield values one at a time, useful for memory efficiency.
Long: Generators are a type of iterable that yield values lazily, producing them one at a time and only when needed. Instead of returning a complete list, they yield one item at a time, saving memory. They're useful when working with large datasets or streams of data. A generator function contains the yield statement and maintains its state between executions.
Copy code

```
def count_up_to(n):
    i = 1
    while i <= n:
        yield i
        i += 1
```

6.Intepreted Language

Thursday, September 26, 2024 8:56 PM

Python is compiled or interpreted language? What does it mean?
Short: Interpreted, executed line-by-line at runtime.
Long: Python is an interpreted language, meaning the code is executed line by line by the Python interpreter at runtime, rather than being compiled into machine code beforehand. This makes Python highly dynamic and platform independent, but often slower compared to compiled languages like C or Java.

7. Lists and Tuples

Thursday, September 26, 2024 8:59 PM

What is the difference between lists and tuples in Python?
Short: Lists are mutable; tuples are immutable.
Long: Lists and tuples are both sequence data types, but the key difference is that lists are mutable, meaning their contents can be changed after creation, while tuples are immutable, meaning they cannot be modified once created. Lists are commonly used for dynamic data, while tuples are preferred for fixed data that shouldn’t be altered.

8.Lists and Sets

Thursday, September 26, 2024 9:33 PM

What is the difference between lists and sets in Python?
Short: Lists allow duplicates; sets don’t.
Long: Lists in Python are ordered and can contain duplicate elements, while sets are unordered collections of unique elements. Lists maintain their insertion order, while sets do not. Sets are typically used when uniqueness is required and membership checks need to be fast.

9.Use of Dictionary

Thursday, September 26, 2024 9:01 PM

When to use dictionary?
Short: When you need fast lookups with key-value pairs.
Long: Dictionaries are used when you need to store data in key-value pairs, providing fast O(1) access to values based on their keys. They are particularly useful when you have a set of labels (keys) and corresponding data (values), such as for configurations, lookup tables, or mappings.

10. Decorators

Thursday, September 26, 2024 9:07 PM

What are decorators? When to use?
Short: Functions that modify other functions, used to add extra functionality.
Long: Decorators are higher-order functions that allow modification of a function's behavior without changing its code. They wrap another function, adding functionality before or after its execution. Common use cases include logging, authentication, and access control.
Example:

```
def log_decorator(func):
    def wrapper(*args, **kwargs):
        print(f"Calling {func.__name__}")
        return func(*args, **kwargs)
    return wrapper
@log_decorator
def greet(name):
    print(f"Hello, {name}")
```

11.Iterators

Thursday, September 26, 2024 9:10 PM

What are iterators?
Short: Objects that can be iterated over using __next__() method.
Long: Iterators are objects in Python that implement the __iter__() and __next__() methods, allowing them to be iterated over in a loop. They represent a sequence of values that can be traversed one by one, and once exhausted, raise a StopIteration exception.

12. Slicing

Thursday, September 26, 2024 9:11 PM

What is slicing?
Short: Extracting a portion of a sequence.
Long: Slicing refers to extracting a portion of a sequence (like a list, string, or tuple) by specifying a range of indices. You can use the format [start:stop:step] to access a slice of elements. For example, my_list[1:3] returns elements from index 1 to index 2 (stop-1).

13.Mutable and Immutable

Thursday, September 26, 2024 9:15 PM

What is mutable and immutable?
Short: Mutable can change; immutable cannot change.
Long: Mutable objects (like lists and dictionaries) can be modified after creation, meaning you can add, remove, or update elements. Immutable objects (like tuples and strings) cannot be changed once they're created. This distinction affects how data is passed between functions and stored in memory.

14.Thread or Multithread

Thursday, September 26, 2024 9:17 PM

Python is single thread or multithread?
Short: Supports both, but GIL limits true multithreading.
Long: Python can handle both single-threading and multithreading. However, due to the Global Interpreter Lock (GIL), only one thread can execute Python bytecode at a time in standard CPython, which can limit true parallelism in CPU-bound tasks. For I/O-bound tasks, multithreading works effectively.

15.GIL

Thursday, September 26, 2024 9:20 PM

What is GIL?
Short: Global Interpreter Lock restricts one thread at a time in CPython.
Long: GIL (Global Interpreter Lock) is a mutex in CPython that ensures only one thread executes Python bytecode at a time, preventing true parallel execution in multithreaded CPU-bound programs. It is a tradeoff to simplify memory management in Python but can be a bottleneck for multithreaded programs.

16.Don't like Python

Thursday, September 26, 2024 9:22 PM

What you don’t like about Python?
Short: Slower execution and GIL limits multithreading.
Long: Python’s interpreted nature makes it slower compared to compiled languages like C or Java, especially for CPU-bound tasks. The Global Interpreter Lock (GIL) also limits true multithreading, which can be a drawback for developers working on performance-critical applications that require parallel execution.

17.List Comprehension

Thursday, September 26, 2024 9:24 PM

What is list comprehension?
Short: A concise way to create lists.
Long: List comprehension is a compact way to generate a list by iterating over a sequence and optionally applying a condition or transformation. It allows for concise and readable code compared to using traditional loops.
Example: [x**2 for x in range(10)] generates a list of squares from 0 to 9.

18.Dunder Method

Thursday, September 26, 2024 9:25 PM

What are Dunder methods? Give examples.
Short: Special methods with double underscores, like __init__.
Long: Dunder methods (short for double underscore) are special methods in Python that start and end with double underscores (__). They define special behaviors for objects, such as __init__ for initialization, __str__ for string representation, and __add__ for addition. They allow for operator overloading and customization of object behavior.

19.__init__ method

Thursday, September 26, 2024 9:27 PM

What does __init__ method do?
Short: Initializes an object's attributes.
Long: The __init__ method is the constructor in Python that is automatically called when a new instance of a class is created. It allows for initialization of the object's state by setting its attributes. It takes parameters and assigns them to instance variables for further use.

20.Array&Numpy Array

Thursday, September 26, 2024 9:29 PM

Difference between array and numpy library?
Short: Numpy arrays are faster and support more functionality than Python's arrays.
Long: Python’s built-in arrays are limited to one-dimensional arrays and only support a single data type. numpy, on the other hand, provides a powerful array object that supports multi-dimensional arrays, complex mathematical operations, and a variety of data types, making it more suitable for scientific computing and large-scale data processing.