What is Python?

- **Answer:** Python is an interpreted, high-level, general-purpose programming language known for its readability and support for multiple programming paradigms, including procedural, object-oriented, and functional programming.

What are the key features of Python?

- **Answer:** Key features include simplicity, readability, extensive standard library, support for multiple programming paradigms, dynamic typing, and automatic memory management.

How do you declare a variable in Python?

- **Answer:** Variables in Python are declared by assigning a value to a variable name, without needing an explicit type declaration. Example: x = 10

What is PEP 8?

- **Answer:** PEP 8 is the Python Enhancement Proposal that provides guidelines and best practices for writing Python code, ensuring readability and consistency.

What is the difference between lists and tuples in Python?

- **Answer:** Lists are mutable (can be changed), whereas tuples are immutable (cannot be changed).

How do you create a list in Python?

- **Answer:** my_list = [1, 2, 3, 4]

How do you create a tuple in Python?

- **Answer:** my_tuple = (1, 2, 3, 4)

What is a dictionary in Python?

- **Answer:** A dictionary is an unordered collection of key-value pairs.
  Example: my_dict = {'key1': 'value1', 'key2': 'value2'}

How do you create a dictionary in Python?

- **Answer:** my_dict = {'key1': 'value1', 'key2': 'value2'}

What is a set in Python?

- **Answer:** A set is an unordered collection of unique elements. Example: my_set = {1, 2, 3, 4}

How do you create a set in Python?

- **Answer:** my_set = {1, 2, 3, 4}

What are list comprehensions?

- **Answer:** List comprehensions provide a concise way to create lists. Example: [x**2 for x in range(10)]

What is a lambda function?

- **Answer:** A lambda function is an anonymous function defined with the lambda keyword. Example: lambda x: x + 1

How do you handle exceptions in Python?

Answer: Using try, except, finally blocks

```
try:
   # code that may raise an exception
except Exception as e:
   # code that runs if an exception occurs
finally:
   # code that runs no matter what
```

What is the difference between append() and extend() methods in a list?

- **Answer:** append() adds a single element to the end of the list, while extend() adds all elements of an iterable (e.g., list, tuple) to the end of the list.

Python eval() Function

The eval() function evaluates the specified expression, if the expression is a legal Python statement, it will be executed.

```
# Evaluate the expression 'print(55)':
x = 'print(55)'
eval(x)
```

What is the use of the map() function?

- **Answer:** map() applies a given function to all items in an iterable and returns a map object (an iterator).

How do you convert a string to an integer in Python?

- **Answer:** Using the int() function. Example: int('123')

How do you convert an integer to a string in Python?

- **Answer:** Using the str() function. Example: str(123)

What is the purpose of the split() method in a string?

- **Answer:** split() divides a string into a list of substrings based on a specified delimiter. Example: "hello world".split() results in ['hello', 'world']

How do you check if a key exists in a dictionary?

- **Answer:** Using the in keyword. Example: 'key1' in my_dict

What are the different types of loops in Python?

- **Answer:** for loop and while loop.

How do you define a function in Python?

Answer: Using the def keyword.

```
def my_function():
  print("Hello, World!")
```

What is the purpose of the return statement in a function?

- **Answer:** The return statement is used to exit a function and return a value to the caller.

What are *args and kwargs in function definitions?

- **Answer:** *args allows a function to accept any number of positional arguments, while **kwargs allows a function to accept any number of keyword arguments.

How do you handle files in Python?

Answer: Using the open() function to read or write files

```
with open('file.txt', 'r') as file:
  content = file.read()
```

What is the difference between read() and readlines() methods in file handling?

Answer: read() reads the entire file content as a string, while readlines() reads the file content into a list of lines.

What is the purpose of the with statement in file handling?

- **Answer:** The with statement ensures proper acquisition and release of resources. It automatically closes the file after the block of code inside it is executed.

How do you import a module in Python?

- **Answer:** Using the import keyword. Example: import math

What is the use of the dir() function?

- **Answer:** dir() returns a list of the attributes and methods of an object.

What is the purpose of the help() function?

- **Answer:** help() provides the documentation of modules, classes, functions, and keywords.

What is the difference between deep copy and shallow copy?

- **Answer:** A shallow copy creates a new object but inserts references into it to the objects found in the original. A deep copy creates a new object and recursively copies all objects found in the original.

How do you create a class in Python?

Answer: Using the class keyword.

```
class MyClass:
  def __init__(self, value):
    self.value = value
```

What is inheritance in Python?

- **Answer:** Inheritance is a feature that allows a class (derived class) to inherit attributes and methods from another class (base class).

What is polymorphism in Python?

- **Answer:** Polymorphism allows objects of different classes to be treated as objects of a common superclass. It is typically achieved through method overriding and operator overloading.

What is encapsulation in Python?

- **Answer:** Encapsulation is the concept of bundling data and methods within a single unit (class) and restricting access to some components (using private/protected access modifiers).

What are the different types of access modifiers in Python?

- **Answer:** Public (default), Protected (single underscore _), and Private (double underscore __).

How do you create an object in Python?

- **Answer:** By calling the class name as if it were a function. Example: obj = MyClass()

What is method overloading?

- **Answer:** Method overloading is the ability to define multiple methods with the same name but different signatures. Python does not support traditional method overloading but can be achieved using default arguments.

What is method overriding?

- **Answer:** Method overriding is the ability of a derived class to provide a specific implementation of a method that is already defined in its base class.

What is the use of the super() function?

- **Answer:** super() is used to call the constructor or method of a parent class from a derived class.

What are decorators in Python?

- **Answer:** Decorators are functions that modify the behavior of other functions or methods. They are typically used to add functionality to existing code in a reusable way.

How do you define a decorator in Python?

- **Answer:** Using the @ symbol followed by the decorator function name above the function to be decorated. Example:

```
def my_decorator(func):
  def wrapper():
    print("Something is happening before the function is called.")
    func()
    print("Something is happening after the function is called.")
  return wrapper

@my_decorator
def say_hello():
  print("Hello!")
```

What is a generator in Python?

- **Answer:** A generator is a function that returns an iterator that produces a sequence of values. It uses the yield keyword to produce a value and suspend execution until the next value is requested.

What are the benefits of using generators?

- **Answer:** Generators are memory efficient, can be used to model infinite sequence
- s, and can provide lazy evaluation.

What is a context manager in Python?

- **Answer:** A context manager is an object that defines the runtime context to be established when executing a with statement. It typically manages the setup and teardown of resources.

How do you create a context manager?

- **Answer:** By defining a class with __enter__ and __exit__ methods or using the contextlib.contextmanager decorator.

What is a metaclass in Python?

- **Answer:** A metaclass is a class of a class that defines how a class behaves. A class is an instance of a metaclass.

What is the purpose of the __init__.py file in a Python package?

- **Answer:** The __init__.py file indicates that the directory should be treated as a package. It can also execute initialization code for the package.

What is the difference between __str__ and __repr__ methods?

- **Answer:** __str__ is used to create a string representation of an object for human consumption, while __repr__ is used to create a string representation of an object for debugging and development.

How do you handle memory management in Python?

- **Answer:** Python uses automatic memory management with a built-in garbage collector to reclaim memory occupied by objects that are no longer in use.

What is the Global Interpreter Lock (GIL)?

- **Answer:** The GIL is a mutex that protects access to Python objects, preventing multiple native threads from executing Python bytecodes at once. This ensures thread safety but can be a bottleneck in CPU-bound multi-threaded programs.

What is the difference between deepcopy and copy in the copy module?

- **Answer:** copy.copy() creates a shallow copy of an object,
  whereas copy.deepcopy() creates a deep copy of an object, including recursively copying all objects found in the original.

What are the differences between Python 2 and Python 3?

- **Answer:** Some key differences include print function syntax (print vs. print()), integer division behavior, Unicode support, and the renaming of several built-in functions and libraries.

What is the purpose of the pass statement?

- **Answer:** The pass statement is a null operation; it is a placeholder that does nothing and is used where syntactically some code is required but no action is needed.

How do you measure the execution time of a piece of code in Python?

- **Answer:** Using the time module or the timeit module.

```
import time
start_time = time.time()
# code to measure
end_time = time.time()
print(f"Execution time: {end_time - start_time} seconds")
```

What is a Python package?

- **Answer:** A package is a way of structuring Python's module namespace by using "dotted module names." A package is a collection of modules in directories that include a special __init__.py file.

What are the advantages of using NumPy?

- **Answer:** NumPy provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

What is the purpose of the os module in Python?

- **Answer:** The os module provides a way to interact with the operating system, allowing for file and directory manipulation, environment variable access, and process management.

How do you handle command-line arguments in Python?

- **Answer:** Using the sys.argv list or the argparse module for more complex argument parsing.

What is the purpose of the logging module?

- **Answer:** The logging module provides a flexible framework for emitting log messages from Python programs, allowing for easy debugging and tracking of application behavior.

What are magic methods in Python?

- **Answer:** Magic methods, also known as dunder methods (double underscore methods), are special methods with double underscores before and after their names (e.g., __init__, __str__) that allow for operator overloading and customization of class behavior.

How do you create a virtual environment in Python?

- **Answer:** Using the venv module.

python -m venv myenv

What is the purpose of the pip tool?

- **Answer:** pip is the package installer for Python, used to install and manage Python packages from the Python Package Index (PyPI) and other package repositories.

What is the purpose of the yield keyword?

- **Answer:** The yield keyword is used to create a generator function, which returns an iterator that produces a sequence of values. It allows the function to return a value and resume execution from the same point when called again.

How do you handle JSON data in Python?

- **Answer:** Using the json module.

```
import json
json_data = json.dumps({'key': 'value'})  # Serialize
python_dict = json.loads(json_data)  # Deserialize
```

What is the difference between __new__ and __init__ methods?

- **Answer:** __new__ is the method called to create a new instance of a class, whereas __init__ initializes the instance after it has been created.

What is the purpose of the inspect module?

- **Answer:** The inspect module provides several useful functions to help get information about live objects such as modules, classes, methods, functions, tracebacks, frame objects, and code objects.

How do you test Python code?

- **Answer:** Using testing frameworks such as unittest, pytest, or nose.

What is the purpose of the functools module?

- **Answer:** The functools module provides higher-order functions that act on or return other functions, such as reduce, partial, and lru_cache.

What is the Global Interpreter Lock (GIL) and why is it important?

- **Answer:** The GIL is a mutex that protects access to Python objects, preventing multiple native threads from executing Python bytecodes simultaneously. It is important because it ensures thread safety but can limit concurrency in CPU-bound multi-threaded programs.

How does Python manage memory?

- **Answer:** Python uses automatic memory management with a built-in garbage collector to reclaim memory occupied by objects that are no longer in use. It uses reference counting and a cyclic garbage collector to handle reference cycles.

What are Python's built-in types?

- **Answer:** Some of Python's built-in types
  include int, float, str, list, tuple, dict, set, bool, and NoneType.

How do you optimize Python code for performance?

- **Answer:** Optimization techniques include using built-in functions and libraries, avoiding global variables, using list comprehensions, employing generator expressions, and leveraging C extensions like NumPy.

What is the multiprocessing module and how is it different from threading?

- **Answer:** The multiprocessing module allows the creation of processes, each with its own memory space, enabling true parallelism. It is different from threading, which creates threads within a single process and is subject to the Global Interpreter Lock (GIL).

What are Python's data model and its significance?

- **Answer:** Python's data model defines the fundamental structure and behavior of Python objects, including how they are created, represented, and manipulated. It is significant because it underlies the language's object-oriented nature and supports the customization of class behavior through special methods.

How do you implement a singleton pattern in Python?

- **Answer:** A singleton pattern can be implemented using a class with a class-level attribute to store the single instance or using a decorator.

```
class Singleton:
  _instance = None

  def __new__(cls, *args, **kwargs):
    if not cls._instance:
      cls._instance = super(Singleton, cls).__new__(cls, *args, **kwargs)
    return cls._instance
```

What is the purpose of the abc module?

- **Answer:** The abc module provides tools for defining abstract base classes (ABCs), enabling the creation of abstract methods that must be implemented by subclasses.

How do you use the asyncio module?

- **Answer:** The asyncio module provides support for asynchronous programming, including coroutines, event loops, and tasks.

```
import asyncio

async def main():
  print('Hello')
  await asyncio.sleep(1)
  print('World')

asyncio.run(main())
```

What is the purpose of the dataclasses module?

- **Answer:** The dataclasses module provides a decorator and functions for automatically adding special methods to user-defined classes, such as __init__, __repr__, and __eq__, to create data classes.

How do you perform static type checking in Python?

- **Answer:** Using type hints and tools like mypy to check type correctness at compile time.

What is a coroutine in Python?

- **Answer:** A coroutine is a special type of function that can be paused and resumed, allowing for asynchronous programming. It is defined using async def and can be awaited using the await keyword.

What is monkey patching and how is it used in Python?

- **Answer:** Monkey patching refers to the dynamic modification of a module or class at runtime. It is used to change or extend the behavior of libraries or classes without modifying their source code.

What are Python descriptors?

- **Answer:** Descriptors are objects that define how attribute access is interpreted by implementing methods like __get__, __set__, and __delete__. They are used to manage the behavior of attributes in classes.

What is the sys module used for?

- **Answer:** The sys module provides access to system-specific parameters and functions, such as command-line arguments, standard input/output, and the Python interpreter's runtime environment.

What is the subprocess module used for?

- **Answer:** The subprocess module allows for spawning new processes, connecting to their input/output/error pipes, and obtaining their return codes, enabling interaction with the system's command line.

How do you work with binary data in Python?

- **Answer:** Using the struct module to interpret bytes as packed binary data and the io module for handling binary streams.

What is the difference between bytes and bytearray in Python?

- **Answer:** bytes is an immutable sequence of bytes, whereas bytearray is a mutable sequence of bytes.

What is duck typing in Python?

- **Answer:** Duck typing is a concept where the type or class of an object is less important than the methods it defines or the way it behaves. “If it looks like a duck and quacks like a duck, it must be a duck.”

How do you handle circular imports in Python?

- **Answer:** By using import statements inside functions or methods, using absolute imports, or restructuring the code to avoid circular dependencies.

What is the purpose of the enum module?

- **Answer:** The enum module provides support for creating enumerations, which are a set of symbolic names bound to unique, constant values.

How do you create a custom exception in Python?

- **Answer:** By defining a new class that inherits from the built-in Exception class

```
class CustomError(Exception):
  pass
```

What is the purpose of the itertools module?

- **Answer:** The itertools module provides a collection of fast, memory-efficient tools for creating iterators and performing iterator algebra.

What is method resolution order (MRO) in Python?

- **Answer:** MRO is the order in which base classes are searched when executing a method. It is determined by the C3 linearization algorithm and can be viewed using the __mro__ attribute.

How do you manage package dependencies in Python?

- **Answer:** Using tools like pip, virtualenv, pipenv, or poetry to create isolated environments and manage dependencies.

What is the collections module and what does it provide?

- **Answer:** The collections module provides specialized container datatypes like namedtuples, deques, Counter, OrderedDict, defaultdict, and ChainMap.

What is the purpose of the warnings module?

- **Answer:** The warnings module provides a way to issue and control warning messages in Python, which can be used to alert users about deprecated features or other issues.

What is the abc module and how is it used?

- **Answer:** The abc module provides tools for defining abstract base classes (ABCs), enabling the creation of abstract methods that must be implemented by subclasses.

How do you handle concurrency in Python?

- **Answer:** Using the threading, multiprocessing, and asyncio modules to manage threads, processes, and asynchronous tasks, respectively.

What is the typing module used for?

Answer: The typing module provides support for type hints, enabling static type checking of Python code.