Theoretical Questions

Tuesday, June 18, 2024 2:52 PM

1. Why is Python called an interpreted language?

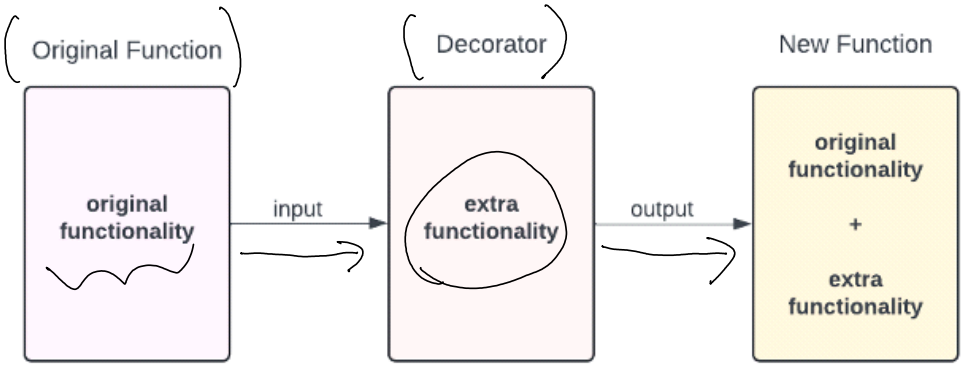
- Python executes each statement line by line
- There's no separate compilation step (C++, Java)
- Stops execution as soon as an error is encountered

2. How to modify the representation of a class in Python?

- Utilize the **__repr__** method

3. What is a decorator in Python?

- It's simply a function which takes another function as input, adds some functionality to it and returns a new function, without altering the execution of the input function
- This new function is the decorated version of the original function
- Decoration means adding additional functionality
- Builds on the concepts of **First-class functions** and **Closures**



4. How to write Python applications that don't throw large and complicated error messages at end users?

- Implement **Exception Handling**
- Involves using the **try**, **except**, **else** and **finally** blocks

5. How do scopes work in Python?

- A scope refers to a block of code within which an object remains relevant/active
- Every object in Python is attached to a scope
- These scopes are also known as namespaces
- There're 4 levels of scopes in Python ---> LEGB
  - Local
  - Enclosed
  - Global
  - Built-in

6. What's the difference between a module and a package?

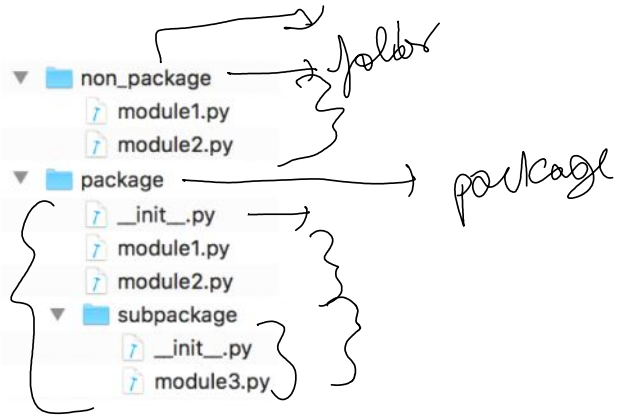
Module:

- A module is simply a file that contains Python code
- In other words, it's simply a Python (.py) file
- It can contain classes, functions, variables, etc.
- Its contents can be imported and used by other modules

Package:

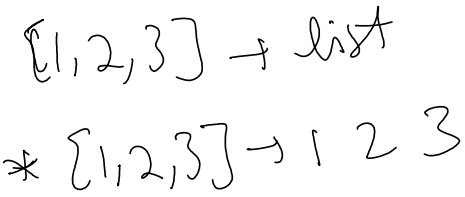
- A package is a collection of related modules
- It can be thought of as a directory that contains related files
- Every package contains an **__init__.py** module. It lets Python know that this is a group of modules which can be imported anywhere for use.

Python **packages** are collections of modules. In a directory structure, in order for a folder containing Python files to be recognized as a package, an `__init__.py` file is needed (even if empty).



7. From a given list of numbers, generate all even numbers in one line.

```
2, 4, 6, 8, 10, 12, 14, 16, 18, 20
```

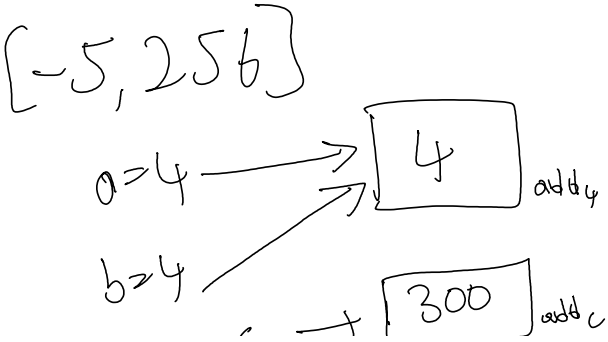


- Make use of list comprehensions
- They're more concise and compact over traditional loops
- It's also the more **Pythonic** way of doing things

8. Explain the output of the below code snippet:

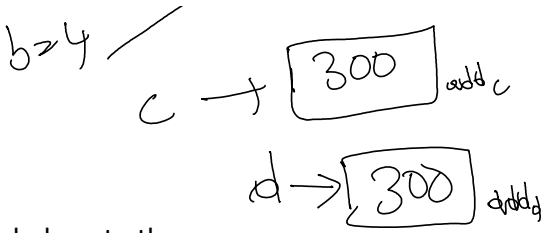
```
a = 4
b = 4

c = 300
d = 300
```



```
d = 300

print(a is b)   # line 1
print(c is d)   # line 2
```



- The **is** operator is used to test membership, if whether two objects belong to the same memory address
- This result is due to the concept of **Interning** in Python

9. What are generators? How are they useful?

- It's a special type of an iterable that returns values on-the-fly, unlike lists and tuples
- Makes use of the **yield** keyword
- When the **yield** keyword in encountered, the value is returned and the state of the function is saved for the next iteration
- Generators are useful when the requirement is to work on one value at a time, where all the other values don't need to be stored in memory. **Ex: working with very large text files**

10. What is shallow copy and deep copy?

- Shallow copy creates a copy of the entire object but maintains the references to each and every element of the object
- Deep copy creates a copy of the entire object as well as each and every element of the object
- Both can be implemented using the **copy** module in Python

11. Distinguish between append and extend methods of a list.

- Append is used for adding a single element to the end of a list
- Extend is used for adding multiple elements to the end of a list. Accepts any iterable for an argument.
- Both the operations modify the list in-place

12. Given a sequence, how to print values separated by space rather than new line?

Input: (4, 5, 6, 7)
Output: 4 5 6 7

13. What're the different ways of reversing a string?

- Using a for loop and 2 pointers
- Using the built-in function **reversed**
- Using slicing

14. How to remove duplicate elements from a list?

- Using a for loop
- Using a set

15. Explain the use of with statement.

- Mainly used for exception handling and resource management
- It ensures that resources are properly managed and cleaned up after use
- Simplifies the process of acquiring and releasing resources, such as file I/O, network/database connections, etc.
- **Ex:** working with files

16. What is __init__ in Python?

- It's a special method, also known as a **constructor.**
- It is used to initialize an instance of a class.
- When you create a new object of a class, Python automatically calls the **__init__** method to set up the initial state of the object
- This method can take arguments, which can be used to initialize the object's attributes.

17. What is a docstring in Python?

- A docstring in Python is a string literal that appears as the first statement in a module, function, class, or method definition.
- It's mainly used to document the object in which it appears.
- They are a key aspect of writing readable and maintainable code, as they describe what the code does, how it works, and how to use it.

18. What is meant by *args and **kwargs in Python?

- These are special arguments that can be passed to a function
- **args** denotes arbitrary number of positional arguments. They're packed as a tuple.
- **kwargs** denotes arbitrary number of keyword arguments. They're packed as a dictionary.

19. Calculate average of numbers given as input in Python.

20. What do the dir() and help() functions do and how are they different?

- The **dir** function is used to list all the attributes and methods associated to an object
- The **help** function will display the documentation associated to the object

21. What is serialization in Python?

- Refers to the process of converting an object into a form suitable for saving/storing it and then reconstructed later

- The reverse process of reconstructing a saved object is called **deserialization**
- **Pickle** is Python's built-in serialization format
- **Ex:** saving trained machine learning models

22. How to check if a class is really a child of a parent class?

- Create an object of the child class
- Utilize the **issubclass** function

23. How to interact with the parent class from within a child class?

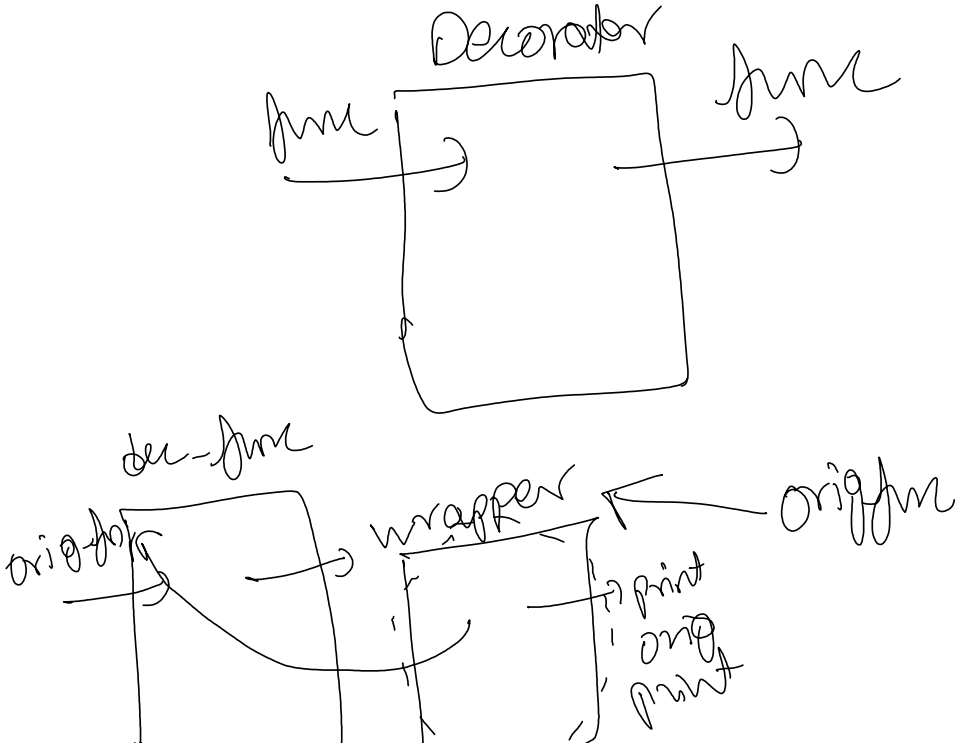
- Utilize the **super** keyword
- This is a function that provides a reference to the parent class

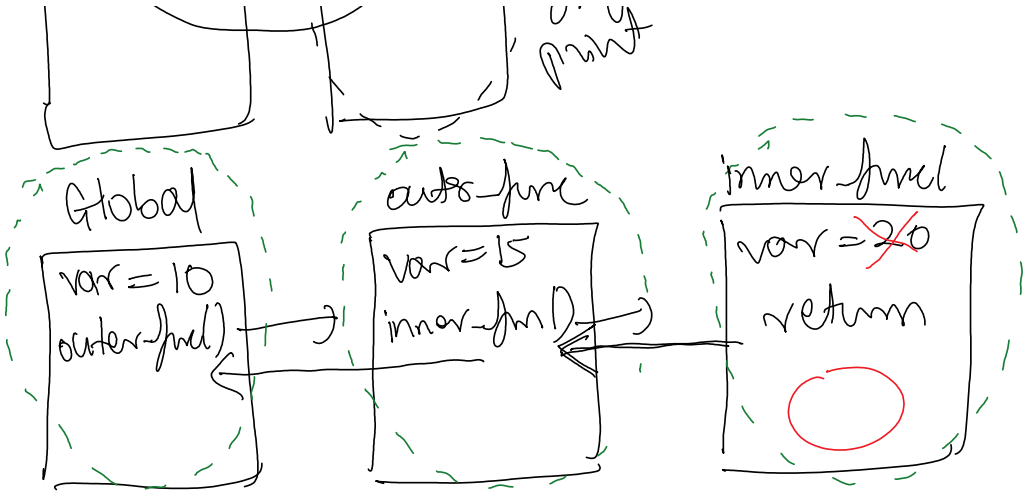
24. What are the main principles of OOP?

- **Encapsulation** - Bundling data (attributes) and related methods (functions) together
- **Inheritance** - Creating new classes from existing ones, enabling reusability of code
- **Abstraction** - Hiding away complex details while only showing relevant info
- **Polymorphism** - Methods behave differently based on the objects they're working with

25. Differentiate between static and class methods.

CLASS METHOD	STATIC METHOD
Can access and modify class-level attributes; modifications reflect on all instances	Cannot work with class nor instance-level attributes
Created using the decorator **@classmethod**	Created using the decorator **@staticmethod**
Can be used as an alternative/flexible constructor to create class instances	Used to perform simple operations meaningful to the class
By convention, the first implicit parameter is **cls**	No implicit first parameter





inside inner fun var = 20
inside outer fun var = 15
inside global var = 10

LinearRegression()
LogisticRegression()
Ridge()
Lasso()

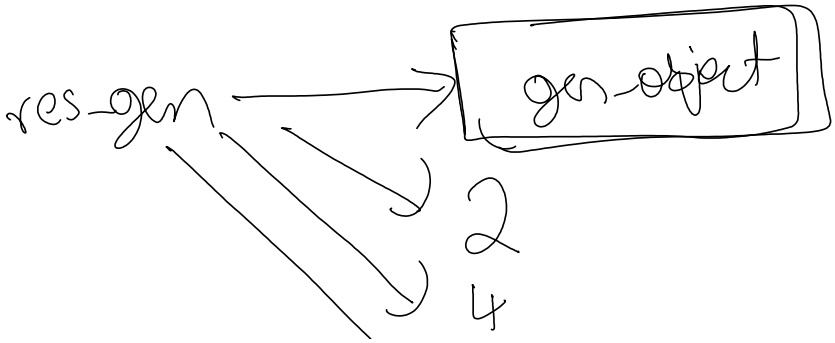
sklearn.linear_model

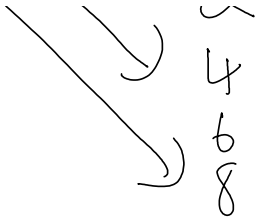
package

modules

OneHotEncoder()
OrdinalEncoder()
StandardScaler()

sklearn.preprocessing





li [1, 2, 3, [4, 5]]

↓

li2 [1, 2, 3, [4, 5]]