Computer science

Data structures

What is the purpose of a list (array in some programming languages) data structure? Name some methods of it!

The purpose of arrays is to store and manage a collection of elements of the same type. Lists provide a way to organize and access data in a sequential manner.

(append, pop, insert, len stb)

What is the difference between a list/array and a set?

Set cannot contain the same element twice or more times.

What is the purpose and methods of a dictionary/map data structure?

The purpose of a dictionary (or map) data structure in programming is to store and retrieve data in key-value pairs. It provides an efficient way to associate values with unique keys, allowing fast lookup and retrieval of values based on their corresponding keys.

Algorithms

Fibonacci sequences. Write a method (or pseudo code), that generates the Fibonacci sequences.

```
def generate_fibonacci_sequence(number_of_terms):
    sequence = []
    a, b = 0, 1
    for _ in range(number_of_terms):
        sequence.append(a)
        a, b = b, a + b
    return sequence
```

OR:

```
def fibonacci_recursive(n):
    if n <= 1:
        return n
    else:
        return fibonacci_recursive(n - 1) +
                      fibonacci_recursive(n - 2)
```

How do you find a max value in a list/array if you can't use any built-in functions?

```
def find_max_value(arr):
     if len(arr) == 0:
           return None
     max_val = arr[0]

     for i in range(1, len(arr)):
          if arr[i] > max_val:
               max_val = arr[i]

     return max_val
```

How do you find the average of values in a list/array if you can't use any built-in functions?

```
def find_average(arr):
    if len(arr) == 0:
        return None

    sum_val = 0
    count = 0

    for num in arr:
        sum_val += num
        count += 1

    average = sum_val / count
    return average
```

Explain an algorithm which sorts a list!

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr

    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]

    return quicksort(left) + middle + quicksort(right)
```

Programming paradigms - procedural

What is the call stack?

A call stack is a data structure used by programming languages to manage function calls. It keeps track of the order in which functions are called and helps the program keep its place and return to the appropriate location after each function completes.
It acts like a stack data structure, following a "last-in, first-out" (LIFO) approach.

What is “Stack overflow”?

A stack overflow occurs when the call stack exceeds its limit due to excessive function calls or deep recursion without proper termination.
A stack overflow error typically causes the program to terminate abnormally and may lead to a crash or exception. It indicates that the program's call stack has exceeded the available memory allocated for it.
To avoid stack overflow errors, it's important to ensure that recursive functions have proper base cases or termination conditions and that the depth of function calls is within the capacity of the call stack.

What are the main parts of a function?

- signature (name, parameters, return type)
- body (code)
- parameters (inputs)
- return statement (for output)
- local variables (temporary storage).

Programming languages - Python

How do you use a dictionary in Python?

What does it mean that an object is immutable in Python?

Immutable is when no change is possible over time. In Python, if the value of an object cannot be changed over time, then it is known as immutable. Once created, the value of these objects is permanent.

This means that any operation that modifies an immutable object returns a new object with the modified value. In contrast to mutable objects, immutable objects are those whose state cannot be modified once they are created. Examples of immutable objects in Python include strings, tuples, and numbers.

What is variable shadowing? (context: variable scope)

Variable shadowing occurs when a variable declared in an inner scope has the same name as a variable in an outer scope. The inner variable takes precedence within its scope, hiding the outer variable with the same name. This can lead to confusion and unintended consequences when accessing or modifying variables.

What can happen if you try to delete/drop/add an item from a List, while you are iterating over it in Python?

If you try to delete, drop, or add an item from a list while iterating over it in Python, it can lead to unexpected and potentially incorrect results. Modifying a list during iteration can cause the iterator to skip elements, process the same element multiple times, or result in an index error.

What is the "golden rule" of variable scoping in Python (context: LEGB)? What is the lifetime of variables?

The "golden rule" of variable scoping in Python, known as the LEGB rule, specifies the order in which Python searches for variables: Local, Enclosing, Global, and Built-in scopes. Variables are searched in this order until a match is found or until the built-in scope is reached.

The lifetime of variables depends on their scope:

- Local variables exist within the function and are created when the function is called and destroyed when it completes.
- Global variables exist at the module level and persist throughout the program's execution.
- Enclosing variables exist within nested functions and are created when the outer function is called and destroyed when the inner function completes.
- Built-in variables are provided by Python and are always available during program execution.

Understanding variable scoping and the order of search is important for proper variable access and avoiding naming conflicts.

If you need to access the iterator variable after a for loop, how would you do it in Python?

I can either store in an outside variable, or use enumerate(), which stores the last iterator and iterable variable.

What type of elements can a list contain in Python?

A list in Python can contain elements of any data type, including numbers, strings, booleans, lists, tuples, dictionaries, sets, and objects. Lists provide flexibility to store and manipulate heterogeneous data in a single data structure.

What is slice operator in Python and how to use?

In Python, the slice operator allows you to extract a portion, or slice, of a sequence, such as a string, list, or tuple. It provides

a concise way to extract a range of elements from the sequence based on specified start, stop, and step parameters.
sequence[start:stop:step]

What arithmetic operators (+,*,-,/) can be used on lists in Python? What do they do?

- Addition Operator (+): It concatenates two lists, creating a new list that contains all the elements from both lists.
- Multiplication Operator (*): It repeats the elements of a list a specified number of times, creating a new list.
- Others throw “TypeError”

What is the purpose of the in and not in membership operators in Python?

The “in” and “not in” membership operators in Python determine if an element is present or absent in a sequence, returning “True” or “False” accordingly, providing a concise and intuitive way to check for membership without explicit iteration or search logic.

What does the + operator mean when used with strings in Python?

The “+” operator, when used with strings in Python, performs string concatenation, merging two strings into a single string.

Explain f strings in Python!

F-strings, also known as formatted string literals, are a way to embed expressions inside string literals in Python, allowing you to dynamically insert variables, expressions, and even function calls directly into the string without the need for explicit concatenation or formatting. They are denoted by the prefix "f" and curly braces "{}" containing the expressions to be evaluated and inserted within the string.

Name 4 iterable types in Python!

- Lists (`[]`)
- tuples (`()`)
- strings (`""` or `''`)
- sets (`{}`)

What is the difference between list/set/dictionary comprehension and a generator expression in Python?

The main difference between list/set/dictionary comprehensions and generator expressions in Python lies in their execution and memory usage.

List/Set/Dictionary Comprehension:

- Comprehensions create a new list, set, or dictionary by evaluating an expression over an iterable.
- They construct the entire sequence in memory before returning the result.
- Comprehensions provide a concise way to generate a sequence of values.

Generator Expression:

- Generator expressions produce a generator object that generates values on-the-fly as they are needed.
- They do not construct the entire sequence in memory at once, but generate values one by one as requested.
- Generator expressions are more memory-efficient, especially for large or infinite sequences, as they only hold the currently generated value in memory.

```
my_list = [x * 2 for x in range(10)]

my_generator = (x * 2 for x in range(10))
```

Does the order of the function definitions matter in Python? Why?

Yes, the order of function definitions matters in Python because it is an interpreted language that executes code from top to bottom, so functions need to be defined before they are called or referred to in order to avoid errors. However, it's worth noting that within a single function or local scope, the order of function definitions does not matter. Functions can call each other within the same scope without any issues, regardless of their order within that scope.

What does unpacking mean in Python?

Unpacking in Python involves assigning values from an iterable to multiple variables in a single statement. For example, `x, y, z = (1, 2, 3)` assigns the values 1, 2, and 3 to variables `x`, `y`, and `z` respectively.
OR:
my_list = [1, 2, 3, 4, 5]
a, *rest, c = my_list
1, [2, 3, 4], 5

What happens when you try to assign the result of a function which has no return statement to a variable in Python?

Functions in Python implicitly return None if there is no explicit return statement or if the return statement doesn't specify a value. Consequently, when you assign the result of such a function to a variable, that variable will be assigned the value None.

Software engineering

Debugging

What techniques can you use while debugging a program in Python?

Some techniques for debugging in Python include using print statements, using a debugger like “pdb” module (Python provided built-in debugging tools), logging relevant information, handling exceptions, seeking code reviews, and writing unit tests.

What does step over, step into and step out mean while using the debugger?

- "Step over" moves to the next line of code, executing any function calls as a single step.
- "Step into" enters a function call, allowing you to debug within the called function.
- "Step out" quickly advances the debugger's execution to the end of the current function, allowing you to exit and return

to the calling context without stepping through each line of the function's code.

How can you start to debug a program from a certain line using the debugger?

To start debugging a program from a specific line using a debugger, you can set a breakpoint at the desired line. A breakpoint is a designated point in the code where the debugger pauses program execution, allowing you to inspect variables and step through the code from that point onwards.

Version control

What are the advantages of using a version control system?

Version control systems offer advantages such as tracking changes, facilitating collaboration, providing backups, enabling branching and parallel development, supporting code reviews, ensuring traceability, and facilitating continuous integration and deployment.

What is the difference between the working directory, the staging area and the repository in git?

The working directory is where you make changes to your files, the staging area is where you prepare changes for the next commit, and the repository is where Git permanently stores your project's history.

What are remote repositories in git?

Remote repositories in Git are like duplicate copies of a project stored on a server. They help developers work together by sharing and syncing their code changes. With remote repositories, developers can easily get updates from others, send their own changes, and collaborate on projects.

Why does a merge conflict occur?

I usually ask the same question.

Merge conflicts happen in Git when there are conflicting changes between branches that are being merged. This occurs when the same part of a file has been modified in different ways or when one branch deletes a file while the other modifies it. Resolving merge conflicts involves manually addressing and reconciling these conflicting changes to ensure a successful merge.

Through what series of commands could you put a new file into a remote repository connected to your existing local repository?

- Use `git add <file>` to add the new file to the staging area.
- Commit the changes using `git commit -m "Commit message"` to save the file to the local repository.
- Push the committed changes to the remote repository with `git push <remote> <branch>` (e.g., `git push origin main`) to make the file available remotely.

What does it mean atomic commits and descriptive commit messages?

- Atomic commits involve making small, focused commits that address a single logical change, improving codebase maintainability and facilitating easier tracking and reverting of changes.
- Descriptive commit messages provide detailed summaries of changes, explaining the purpose and context of modifications, aiding collaboration, code review, and historical tracking.

What’s the difference between git and GitHub?

Git is the underlying version control system, while GitHub is a cloud-based service that hosts Git repositories and adds collaboration and management features on top of Git. Many developers use Git locally for version control and then use GitHub to host and share their repositories with others.

Software design

Clean code

What does clean code mean?

"Clean code" refers to code that is easy to read, understand, and maintain. It follows coding conventions, principles, and best practices to enhance readability, minimize complexity, and promote efficient collaboration among developers. Clean code is well-structured, concise, properly documented, and demonstrates good software engineering practices. It prioritizes clarity, simplicity, and readability to facilitate code comprehension, debugging, and future modifications.

What steps do we usually do during a clean code refactoring?

- Identify areas for improvement.
- Understand the existing code and its dependencies.
- Plan the refactoring strategy and goals.
- Break down changes into smaller tasks.
- Write or update tests to ensure correctness.
- Refactor the code, following best practices.
- Review and validate the changes.
- Iterate if necessary for further improvements.

Error handling

What is exception handling?

Exception handling is a programming approach that enables the detection and handling of unexpected situations, referred to as exceptions. By using try-except blocks, programmers can anticipate and gracefully respond to these exceptions, preventing program crashes and maintaining control over the program's execution flow.

What are the basics of exception handling in Python?

In Python, exception handling involves using try-except blocks to handle potential errors or exceptional conditions. The code that may raise an exception is placed in the try block, and if an exception occurs, it is caught and handled in one or more except blocks. This allows for customized responses to different types of exceptions, preventing program crashes and enabling graceful error handling. Additionally, an optional finally block can be used to specify cleanup code (e.g. closing files or releasing resources.) that always runs, regardless of whether an exception was raised or not.

In which case should we catch an exception? Why?

Exceptions should be caught when you have a specific way to handle and recover from the exceptional situation, providing alternative logic and preventing abrupt program termination. Catching exceptions allows for customized responses and enables error recovery mechanisms, while also facilitating logging and monitoring for debugging and analysis purposes.

What can/should we do with an exception in the ‘except’ block?

In the “except” block, you can perform actions such as logging the exception, implementing error handling logic, re-raising the exception, providing custom responses, or performing necessary cleanup operations. The specific actions depend on the requirements of your program and the nature of the exception.

What does the else and finally statement do in a try-except block in Python?

- “else” statement, which is optional, is executed after the “try” block if no exceptions are raised. This allows you to define code that should run specifically in the absence of exceptions
- “finally” statement, also optional, is always executed regardless of whether an exception is raised or caught. It is commonly used for cleanup tasks or ensuring certain operations are performed consistently, irrespective of exceptions.