RECURSION

- Recursion is a process where a function calls itself to produce a result with a terminating condition included within it.
- Recursion is implemented using stack because activation records are to be stored in LIFO order (last in first out).
- An activation record contains arguments,return addresses and local variables of the function.
- STACK IS A LINEAR DATA STRUCTURE IN WHICH ELEMENTS IS ADDED TO THE TOP AND DELETED FROM THE TOP.

ADVANTAGES OF CALLBACK:

1] Calling function can call the callback function as many as possible to complete the specified task.

2] Calling function can pass appropriate parameters according to the task to be completed, to the called functions .This allows info hiding.

3] Calling function acts as an interface through which other functions can be called.

USAGE OF CALLBACK :

1] Event driven programming

2] Extensively used in functional programming

3] Separate functionality and its call

4] Asynchronous programming

ADVANTAGES OF RECURSION:

1] It makes the code look clean and elegant

2] A complex task can be broken down into simpler sub-problems

3] Sequence generation is easier

DISADVANTAGES OF RECURSION:

1] They take up a lot of time and memory

2] They can be computationally expensive

3] Hard to debug and sometimes tough to understand the logic behind recursion

4] It can lead to infinite recursion if not used properly

- A CLOSURE in python is a function object that remembers values in enclosing scopes even if they're not present in memory.
- Nested functions are important in closures in Python because they allow you to access variables from outer func even after the outer func has returned which makes the python code more concise and reusable.
- ALL NESTED FUNCTIONS ARE NOT CLOSURES BUT ALL CLOSURES ARE NESTED FUNCTIONS.

CHARACTERISTICS OF CLOSURES:

1] It is a nested func

2] It has access to a free variable in outer scope

3] Nested func is returned by the enclosing func

ADVANTAGES OF CLOSURES:

1] It provides data hiding

2] It is used while building python micro services

3] We can avoid using the global values by calling the inner func outside its scope.

- DECORATORS are a powerful and useful tool in python since it allows programmers to modify the behavior of a func or class without changing the source code of it.
- It wraps another func in order to extend the behavior of wrapped func, without permanently modifying wrapped func.
- CHAINING DECORATORS – Decorating a func with multiple decorators

USES OF DECORATORS:

1] Logging – Debugging and tracking performance

2] Performance testing – Optimizing the performance

3] Catching – Improving performance and reducing database load

4] Memorization – Improves performance by avoiding unnecessary calculations

5] Authorization – Implementing security features.

ADVANTAGES OF DECORATORS:

1] Code reusability

2] Extensibility

3] Debugging and logging made easy

- GENERATOR is an iterable object that has the capability to produce desired sequence of values when iterated for it.
- Generators are a great way to save memory when working with large datasets.
- GENERATOR FUNC – Returns an iterated object that produces a sequence of values when iterated over it

CHARACTERISTICS OF GENERATORS:

1] Has one or more yields

2] Generators are lazy and do not produce all results at a time.

PIPELINING GENERATORS – Multiple generators can be used to pipeline a series of operations.

BENEFITS OF GENERATORS:

1] Supports lazy evaluation

2] Generators are really nice tool to express certain ideas in a very clean and concise fashion.

GUI – GRAPHICAL USER INTERFACE:

An interface that allows users to interact with electronic devices.

BENEFITS OF GUI:

1] Easy to use

2] Easy to communicate

3] Attractive

4] Provides shortcuts

5] Allows for multi tasking

Tkinter:

A standard python interface to the Tk GUI toolkit skipped with python.

Easiest and fastest way to create GUI applications

Each element in tkinter is a widget. Each widget is a python object

GEOMETRY MANAGER CLASSES:

1. Pack() – Organizes the widgets in blocks placing in parent widget.
2. Grid() – Organizes the widgets in 2-D table before placing in parent widget .
3. Place() – Organizes the widgets by placing them on specific position on the window as directed by the programmer

Color options:

- **activebackground**

– Background color for the widget when the widget is ac

- **activeforeground**

– Foreground color for the widget when the widget is ac

- **background**

– Background color for the widget. This can al

- **disabledforeground**

– Foreground color for the widget when the widget is disabled.

- **foreground**

– Foreground color for the widget. This can also be represented as fg.

- **highlightbackground**

– Background color of the highlight region when
focus.

- **highlightcolor**

– Foreground color of the highlight region when the widget has focus.

- **selectbackground**

– Background color for the selected items of the widget.

- **selectforeground**

– Foreground color for the selected items of the widget.

MODULES:

It is a file consisting of python code.

WHY MODULES:

1. Code organization
2. Namespace Management
3. Dependency Management

TYPES OF MODULES:

1. Built-in functions
2. User Defined modules

IMPORT MECHANISM:

The import keyword is used to load and execute the code from a module.

USAGE OF SYS MODULE:

The module sys provides access to some objects used or maintained by the interpreter and to functions that interact strongly with the interpreter.