Python Definitions

1. Python **
   Python is a High Level, Interpreted, object Oriented Programming Language with easy syntax, dynamic typing and large library used for web development, data science, AI, automation, and more

2. How works Python Interpreter *
   - Source Code :- Tokenization read code and break into small units
   - Parsing :- check syntax and creates an abstract syntax tree (AST)
   - Bytecode Compilation :- AST convert into Bytecode
   - Execution :- Python Virtual Machine (PVM) read bytecode and execute instruction

3. Python Indentation
   It is the space or tabs used at the beginning of a line of code define a block of code (such as inside functions, loops, conditionals) instead of {}

4. Comment
   It is the non executable line in code used to explain code, improve readability or temporarily disable code
   - Inline Comment :- Single line Comment using (#)
   - Block Comment :- Multi-line Comment using (#)
   - DocString :- Trible-quotes strings(””” or ’’’)

5. Variables
   They are Fundamental Building blocks in Python. It is a Container using for store data in memory
   - Types of Variables
     - int :- Whole Numbers
     - float :- Decimal Numbers
     - str :- Text (string)
     - bool :- True or False
     - list :- Ordered Collection
     - tuple :- Ordered, unchangeable
     - dict :- Key-value pairs
     - set :- Unique unordered values

6. Type Casting
   It is the process converting one data type to another data type (int, float, str)
   - Types of Type Casting
     - Implicit Casting (Coercion) :- Automatically done by Python
     - Explicit Casting (Manual) :- We manually convert

7. How to Getting a Type of the Variable
   It check using Python build-in function like type() and isinstance()

8. How to Delete a Variable
   It means Removing its reference in memory using del keyword

9. Mutable and Immutable **
   - Mutable :- It means the value can be changed/modified after creation (list, set , dict, bytearray)
   - Immutable :- It means the value cannot be changed after creation (int, float, str, tuple, frozenset)

10. Variable Scope
    It defines where the variable can be accessed or modified
    - Global Variable :- Variable defines outside the functions
    - Local Variable :- Variable defines inside the function
    - Non-Local Keyword :- Variable used in Nested Function

11. Data Types **
    It is the Classification or Categorization of data items
    - Numeric Data Type :- Numeric values
      - Int :- Whole Numbers (+ve, -ve)
      - Float :- Decimal Numbers (3.23, -2.33)
      - Complex :- Number with Real & Imaginary Part
    - Sequence Data Type :- Collection of Similar data
      - Str :- Text
      - List :- Ordered , Mutable Collection
      - Tuple :- Ordered , Immutable Collection
      - Range :- Immutable, Efficient for Looping
    - Mapping Data Type :- Collection of key-value pairs
      - Dict :- Mutable, unordered
    - Set Data Type :-
      - Set :- Unordered, Mutable Unique Value
      - Frozenset :- Immutable version of set
    - Boolean Data Type :- It have 2 built-in values True/False
      - Bool :- Logical Values (True/False)
    - Binary Data Type :-
      - bytes() :- Immutable sequence of bytes
      - bytearray() :- Mutable sequence of bytes
      - memoryview() :- access memory without copying
    - None Data Type :- It represents absence of value

12. Operators *
    It is the Symbol or Keyword that Perform Operations on Variable and Value
    - Arithmetic Operators :- (+, -, *, /, //, %, **)
    - Comparison Operators :- (==, !=, >, <, <=, >=)
    - Assignment Operators :- (=, +=, -=, *=, /=, //=, %=, **=)
    - Logical Operators :- (and, or, not)

- Identity Operators :- 2 obj same memory (is, is not)
- Membership Operators :- Value exists (in, not in)
- Bitwise Operators :- (&, |, ^, ~, <<, >>)

13. Conditional Statement *
    It Control Program Flow using Boolean Logic
    - If Statement :- It Execute a Block of Code if the Given Condition is True
    - If-else Statement :- it Execute a Block of Code based on Condition
    - elif Statement :- It Stands for else if when Multiple Condition is Occured
    - Ternary Statement :- It is Short-Hand for if-else Statement
    - Match-Case Statement :- It Pattern MAtching like Switch Case and it to Handle Multiple Conditions

14. Loop
    It is used to Repeat a Block of Code Multiple Times Until a Condition is met
    - For Loop :- It Iterates over a Sequence (list, tuple, string, range, etc..)
    - While Loop :- It Repeats until a Condition become False
    - Nested Loop :- It Loop inside another Loop
    - Control Statement in Loops :-
        - break :- It Stop the Loop Immediately
        - continue :- It Skip the Current Iteration and go to next
        - pass :- It is a Placeholder . It does nothing

15. List **
    It is a collection data type that allows to store multiple items in a single variable.
    It is the Mutable, Ordered Collection of Items Defined with Square Bracket [ ].
    It allows Mixed data Types
    - Methods :-
        - append() :- Add to Last
        - insert() :- Add with Index Number
        - extend() :- Add Multiple Items
        - remove() :- Remove with Item Name
        - pop() :- Remove last Item
        - clear() :- Remove All Items
        - sort() :- Sorting Items
        - sorted() :- Return New Sorted List without Modify Org List
        - reverse() :- Reversing Items
        - len() :- Find Length of Item
        - index() :- Find the Position of item
        - count() :- Find Count Occurrence
        - copy() :- Shallow Copy
        - min() :- Smallest Number
        - max() :- Largest Number
        - sum() :- Sum of Number

16. Comprehension **

It is a short and elegant way to create new sequences (like lists, sets, or dictionaries) from existing sequences or iterables (like lists, tuples, strings, ranges, etc.) in a single line.

Types of Comprehension :-

- List Comprehension :- Create a list in a Single Line
- Set Comprehension :- Like list , but Create a set (remove duplicates)
- Dictionary Comprehension :- Create dict in One line
- Generator Comprehension :- Like list, but use () instead of []

17. Tuple **

It is an Ordered , Immutable Collection of elements it defines using Parentheses ( ) . It allow Mixed data types

- Methods :-
  - index :- It Return First Index
  - count :- It Return the No of Values Appears in Tuple

18. Set **

It is an Mutable, Unordered Collection of Unique Items defined with Curly Braces { }. It Not Allow Duplicate Values

- Methods :-
  - add() :- Add a Single Item
  - update() :- Add Multiple Items
  - remove() :- It for Removing but there is no value it occurs error
  - discard() :- for Removing but there is no value it not occur error
  - pop() :- Removing Random Element
  - clear() :- Removing All Element
  - len() :- Find Number of Items
  - min() :- Find Smallest Item
  - max() :- Find Largest Item
  - sum() :- Find Sum of Number
  - sorted() :- Sorting Item
- Mathematical Set Operations
  - Union (|) :- It Combining Sets , Removing Duplicates
  - Intersection (&) :- It Find Common Element
  - Difference (-) :- It Check element in “a” but not in “b”
  - Symmetric (^) :- It Check elements in either set but not both
  - Subset (<=) :- All elements of “a” are in “b”
  - Superset (>=) :- All elements of “b” are in “a”
  - Proper Subset :- All elements in “a” in “b” and (a ≠ b) “a” must be smaller than “b”

19. Dictionary **

It is an Mutable, Unordered and Indexed Collection of key-value pairs

- Methods
  - keys() :- Get all keys
  - values() :- It Access Dictionary Values
  - clear() :- It Remove all Items

        - copy() :- Shallow Copy
        - items() :- It retrieves all (key, value) tuples
        - popitem() :- It remove & return (key,value)pairs in LIFO order
        - update() :- It add/update key-value pairs from another dict
        - setdefault() :- It get value for key if exist, else default & return

20. Copying Dictionaries *
    It means making another Dictionary with the same data
    - Shallow Copy :- It create a new dict with same outer value as the org
    - Deep Copy :- It create a new dict and it completely independent

21. String *
    It is a sequence of characters enclosed in single (‘’), double(“”) or triple quotes (“”” or ’’’). It is Immutable
    - Methods
        - upper() :- It convert into Upper Case
        - lower() :- It convert into Lower Case
        - capitalize() :- It Capitalize the first letter
        - title() :- It Capitalize the first letter of every word
        - strip() :- It remove leading & trailing Whitespaces
        - replace() :- It replace Old substring with new
        - split() :- It spits string into a list using separators
        - join() :- It Join elements of iterable into string
        - startswith(prefix) :- Check string start with given substring
        - endswith(suffix) :- Check string end with given substring
        - center(width, char) :- centered string with given width
        - count() :- Count how many times a substring appears
        - find(substring) :- Find index
        - isalnum() :- return true if string contain only letter & numbers
        - isalpha() :- return true if string contains only letters
        - isdigit() :- return true if string contains only digit
        - islower() :- Check all characters in lower case
        - isupper() :- Check all characters in upper case
        - isspace() :- Check string contain only whitespaces
        - swapcase() :- Converts uppercase to lowercase & vice versa

22. Function **

It is a Reusable Building Block of code that performs a specific task. Instead of repeating the same code again and again , we define it once and call it whenever needed

- Types of Function
  - Built-in Function :-
    It is the Standard Function in Python that already available to use (print(), len(), type())
  - User defined Function :-
    We can create our own functions based on our requirements. Use def to create
  - Lambda Function :-
    It is a smallest anonymous function, written in one line using with lambda keyword
  - Recursion Function :-
    It is the technique where a function call itself in order to solve a smaller part of the same problem
- Types of Function Arguments
  - Default Argument :-
    It is a function parameter that takes default value if the caller does not provide one
  - Keyword Argument :-
    It is the way to passing argument to a function explicitly naming the parameter along with its value
  - Positional Argument :-
    It is an argument is passed to a function in the correct order as defined in the functions parameter list
  - Variable length Argument :-
    - Arbitrary Positional Argument (*args)
      It collects extra positional arguments into a tuple inside the function
    - Arbitrary Keyword Argument (**kwargs)
      It collects extra keyword arguments into dictionary inside the function

- Types of Function based on Argument & Return value
  - Without Argument , Without Return Value
    It does not take any input and does not return any value
  - With Argument , Without Return Value
    It takes input but does not return a value
  - Without Argument , With Return Value
    It does not take input but return a value
  - With Argument , With Return Value
    It takes input and return a value

23. Modules *
    It means a file contains Python code that can be imported and reused in other Python Programs
    - Built-in Modules
        - math :- Math operations (sqrt,pi,degree,redians,sin,cos,tan,etc)
        - random :- Generate Random Numbers
        - datetime :- It works with date and time
        - os :- Interact with Operating System
        - sys :- System-specific parameters & functions
        - json :- Work with JSON data

24. Exception Handling *
    It is a Mechanism in Python to handle runtime errors so that the program doesn't crash and can continue execution
    - Built-in Exception
        - ValueError :- Invalid Value Type
        - TypeError :- Wrong Data Type Operation
        - IndexError :- Invalid Sequence Index
        - KeyError :- Missing Dictionary Key
        - FileNotFoundError :- File doesn't exist
        - ZeroDivisionError :- Division by Zero (1/0)
        - ImportError :- Module not Found

25. Namespace *
    It is a container that holds names(identifiers) and maps them to objects
    - Types of Namespace
        - Local Namespace :- Inside a function
        - Enclosing Namespace :- Outer function
        - Global Namespace :- At Module Level
        - Built-in Namespace :- Python Built-in functions

26. Closure *
    It is a function defined inside another function(nested function) that remembers the variables from the outer function even after the outer function has finished executing

27. Decorator **
    It is a function that takes another function as input, add extra behaviour, and returns a new function, without changing the original functions code
    - Types of Decorators
        - Function Decorator
          It takes a function as input and return a new function
        - Method Decorator
          It is used to decorate methods with a class
        - Class Decorator
          It is used to modify and enhance the behaviour of class

- Built-in Decorators
    - @staticmethod
      It is a method that does not access the instance (self) or Class (cls)
    - @classmethod
      It is a method that receives the class(cls) as its first argument instead of an instance
    - @property
      It covert a method into read only attribute

28. Class **
It is a blueprint or template used for creating objects that have data (attributes) and behavior (methods). It is a collection of objects. It is created with class keyword

- Types of Classes
    - Normal/Regular Class :-
      It is Standard Class with Attribute (data) and Methods (function) for creating objects
    - Abstract Class :-
      It is a Class that cannot be instantiated directly and is meant to be inherited by other classes. It implemented using abc module
    - Concrete Class :-
      Normal class with complete implementation.
    - Nested / Inner Class :-
      Class defined inside another class
    - Derived / Child Class :-
      Class that inherits from another class
    - Singleton Class :-
      Class that allows only one object (instance) to be created throughout the program

- Methods In Class
    - Instance Method
      It works with an instance variable, Its first Argument is self. It accessed by Object
    - Class Method (@classmethod)
      It works with class variables, Its first argument is cls. It accessed by class/Object
    - Static Method (@staticmethod)
      It is a normal function inside class, It has no self or cls. It accessed by class/Object
    - Property Method
      It is a special method that converts methods into read only attributes. It allows getter, setter, deleter for attributes

- Types of Variables
    - Instance Variable

        Belong to each object. Declared using self inside __init__ or other instance methods. Each object gets its own copy.
    - Class Variable

        Declared inside class but outside methods. Shared by all objects of that class. Accessed using Class.var or self.var.
    - Global Variable

        Declared outside all classes/functions. Accessible everywhere unless shadowed by a local variable. Can be modified inside a function with global keyword

29. Special Method (Magic / Dunder) *

    It is the Special Method in python with double underscore at the start and end of their name

    - Methods
        - __init__ :-

            Constructor method, automatically called when an object created
        - __str__ :-

            It is the Human-readable string representation of an Object (for print()) - for user
        - __repr__ :-

            It is the official string representation of an object. It make string unambiguous(it debugging)- for developers
        - __add__ :-

            It define the behavior of the + operator for objects of a class
        - __sub__ :-

            It is used to overload the subtraction operator (-)
        - __mul__ :-

            It is used to overload the multiplication operator (*)
        - __truediv__ :-

            It is used to overload the division operator (/)
        - __del__ :-

            It is Destructor method, automatically called when an object is deleted (or goes out of scope)
        - __len__ :-

            It define the behaviour of built-in function len() for custom object
        - __call__ :-

            It means make object behaviour like a function
        - __eq__ :-

            It define the behaviour of equality operator (==) for objects of a class

    - __lt__ :-
      It define the behaviour of less than operator (<)
    - __gt__ :-
      It define the behaviour of less than operator (<)
    - __getitem__ :-
      It define assign value using Indexing operator [ ]
    - __setitem__ :-
      It define the behaviour of less than operator (<)
    - __delitem__ :-
      It define the behaviour when use the del keyword with indexing
    - __enter__ :-
      It defines what happens when you enter a `with` block
    - __exit__ :-
      It defines what happens when you exit a `with` block

30. Object
    It is an instance of a class that represents a real-world entity. It is a collection of data and behavior defined by its class
    - Self Parameter
      It is a reference to the current instance of the class. It allows us to access the attributes and methods of the object

31. OOP (Object-Oriented Programming) **
    It organizes code into objects and classes which combine data (attributes) and functions (methods).
    Main Pillars of OOP :-
    - Inheritance
      It Means a Child Class can reuse attributes and methods of a Parent Class.
      Types of Inheritance :-
        - Single Inheritance :-
          A child class inherits from a single parent class
        - Multiple Inheritance :-
          A child class inherits from two or more parent class
        - Multilevel Inheritance :-
          A child class inherits from a parent class, and another class inherits from that child
        - Hierarchical Inheritance :-
          Multiple child class inherits from a single parent class
        - Hybrid Inheritance :-
          A combination of two or more types of inheritance

        - Other Topics
            - MRO (Method Resolution Order)
              It defines the order in which python looks for methods & attributes when multiple inheritance is involved.
              Python uses the C3 Linearization algorithm to decide MRO , it looks parent classes from left to right
            - super() in Inheritance
              Used to call parent class constructor and method

- Polymorphism
  It allows methods to have the same name but behave differently based on the object context.
  Types of Polymorphism :-
    - Compile-Time Polymorphism (Static)
      It occurs when a method / operator has multiple forms, and the form to be used is determined at compile-time not run-time
        - Method Overloading
          It has multiple functions / methods with the same name but different parameters. It supported in C,C++,Java ,etc not in python because Python is dynamically typed
    - Run-Time Polymorphism (dynamic)
      It occurs when the method that is executed is determined at runtime
        - Method Overriding
          It occurs when a child class defines a method with the same name as a method in its parent class
        - Built-in (duck typing)
          It means we don't care about the type of the object (class name), we only care if the object has the method/behavior we want

- Encapsulation

  It is the binding of data and methods within the class and restricting direct access to some data for security and control

  Types of Encapsulation :-

  - Public Encapsulation

    Variables and methods of a class are accessible from anywhere. Data is fully visible for everyone. No restriction on reading,accessing and modifying (obj.data)
  - Protected Encapsulation

    It means the variables and methods of a class are intended to be accessed only within the class and its subclasses. (obj._data)
  - Private Encapsulation

    It means variables or methods are restricted to the class itself only (obj.__data)

    Name Mangled = It is a process in Python that automatically changes (or mangles) the name of private variables (those starting with `__`)

- Data Abstraction

  It is the process of hiding implementation details and showing only the essential features of an object to the users.
  Components of Abstraction :-

  - Abstract Class :-

    It is a Class that cannot be instantiated directly and is meant to be inherited by other classes. It is implemented using the abc module.
  - Abstract Method :-

    Methods declared in an abstract class, but without implementation. Subclasses must implement them.
  - Concrete Classes :-

    Classes that implement abstract methods. Can be instantiated normally.
  - Encapsulation (data hiding) :-

    Helps abstraction by restricting access using private/protected variables.

32. Iterator **
    It is an object that allows to traverse through a collection of elements (like list, tuple, set, etc.) one at a time.
    - __iter__()
        It returns the iterator object itself.
    - __next__()
        It returns the next element from the collection. Raises StopIteration when no items are left.

33. Generator **
    It is a special type of iterator that is created using a function with yield keyword. It allows you to generate values on the fly (lazy evaluation) instead of storing them all in memory at once.

34. File Handling
    It allows us to create, read, write, and modify files stored on the system. We use the built-in open() function to work with files
    - Modes :-
        - r :- Read (default). A file must exist.
        - w :- Write. Creates a new file or overwrites if it exists.
        - a :- Append, Add data at end of file
        - x :- Create, Creates file but gives error if file already exists
        - t :- Text mode (default)
        - b :- binary mode (image, video, etc)
        - r+ :- Read + Write
        - w+ :- Write + Read (overwrite)
        - a+ :- Append + Read

35. filter() *
    It is a function used to select elements from an iterable(like list,set,tuple) that satisfy a specific condition
    Syntax = “ filter(function, iterable) ”
        Function :- It return true / false for each element
        Iterable :- the sequence to filter (list, tuple, etc..)

36. map() *
    It is a function applies a given function to each item of an iterable (list,set,tuple) and returns a map object(iterator) with the result
    Syntax = “ map(function, iterable1, iterable2, ...) ”
        Function :- the function apply to each element
        Iterable1, iterable2,... :- one of more iterables to process

37. reduce() *
    It is a function that repeatedly applies a function to the elements of an iterable, reducing the entire sequence to a single value. It’s a part of the functools module.

38. zip() *
    It is a function that combines two or more iterables (list,tuple,set) element-wise into a single iterable of tuples
    Syntax = zip(iterable1, iterable2, ...)
        iterable1, iterable2 … :- Sequences to combine.

39. enumerate()
    It is a function adds an index (counter) to an iterable and returns it as an enumerate object, which can be converted into a list, tuple, or used directly in loops
    Syntax = enumerate(iterable, start=0)
        Iterable :- Any sequence (list, tuple, string, etc)
        start=0 :- Optional; starting index(default is 0)

40. Pass by value **
    It is a copy of the actual value passed to the function. It changes made to the parameter inside the function do not affect the original variable outside the function

41. Pass by Reference **
    It means the function receives a reference to the original object, not a copy. Changes made inside the function affect the original variable

42. Monkey Patching **
    It is the technique of dynamically modifying or extending a class or module at runtime without changing its original source code

43. Memory Management
    It is the process of allocating, managing, and freeing memory for objects automatically during program execution
    Key Components :-
    - Memory Allocation
        Python automatically allocates memory for objects and variables. Every object has a reference count indicating how many references point to it.
    - Reference Counting
        Each object stores a count of references pointing to it.
        When the reference count becomes 0, the object is eligible for garbage collection.
    - Garbage Collection
        Python automatically frees memory for objects that are no longer in use.
        Detects cyclic references (objects referencing each other) using the gc module.
    - Memory Pools (private heep)
        Python uses a private heap to store all objects and data structures. Developers cannot access this memory directly.

44. Global Interpreter Lock (GIL) **
    It is a mutex (mutual exclusion) in CPython that allows only one thread to execute Python bytecode at a time, even on multi-core systems.

45. Multithreading **
    It is a technique in which running multiple tasks (threads) inside a single process, sharing the same memory space.

46. Multiprocessing *
    It is a technique in which multiple processes run concurrently, each with its own memory space, allowing true parallel execution on multiple CPU cores

Optional; But to know

47. Pickling
    It is The process of Converting a Python Object into a byte stream so that it can be saved to a file or transferred over a network

48. UnPickling
    It is the reverse process of Pickling. It means converting byte stream back into a Python object

49. First-Class Function
    It means functions can be assigned to variables, passed as argument, returned from other functions, and stored in data structure, just like any other object

50. Higher-Order Function
    It is a function that either takes one or more functions as arguments or returns another function as its result(or both)

51. Data Class (dataclasses)
    is a special type of class introduced in Python 3.7 (in the dataclasses module) that automatically generates common methods like __init__, __repr__, __eq__, and more, so you don't have to write boilerplate code.

FINISHED😁