System level custom modules

Type safety in python

REVISION

1. Python as a programming language
2. Python compiler and python interpreter  
   computers understand only 1 language -> binary
3. Interpreters -> line by line
4. Python hello.py -> hello.pyc -> created by the compiler
5. Tokens -> [ parts of speech ]
   1. Data types – string int float bool
   2. Variables / identifiers – name, calculatediscount() , print(), int(), str()
   3. Keywords / reserved words – int, float, set, dict, if , elif
   4. Separators -> spaces, [,]
   5. Punctuators -> ‘, :
   6. Operators -> =
   7. Literals -> constants
6. Operators
   1. Arithmetic + - \* / % \*\* //
   2. Relational < > == <> <= >=
   3. Logical and or not
   4. Membership in, not in
7. Conditional constructs
   1. If-else
   2. Switch-case -> since 3.11
8. Looping constructs
   1. For
   2. While
   3. Jump statements break and continue
   4. Nested loops
9. Data structures
   1. String
   2. Tuple
   3. List
   4. Set
   5. Dict
   6. Slicing and indexing
   7. Various methods and functions with respective data structures
10. Functions
    1. Reusable block of code
    2. Allows for modularity
    3. Breakdown of complex application or logic in small parts
    4. To create a function
       1. Use def
       2. Def followed by function name , () and a :
       3. Input -> pass parameters if any
       4. Body consists of the business logic
       5. Output -> if any then use the return keyword
    5. We can create functions
       1. With no parameters and no return type
       2. With parameters and no return type
       3. With no parameters and return type
       4. With parameters and return type
    6. By default function returns None
    7. Functions need to be called by the caller using the function name, followed by () and values if any for the **positional parameters**
11. Scope in python
    1. Default identifiers have a module level scope. Every .py file is called as a module. This is **GLOBAL SCOPE**
    2. Immutable variables defined in global scope are read only inside a function block.
    3. Mutable variables defined in global scope are can be modified inside a function block.
    4. To modify the immutable variables inside the function block use the “global” keyword
12. eval() function that maintains the types of the value provided as input
13. Function with default arguments
    1. Arguments can have default values
    2. Default arguments are always after all the positional arguments
    3. If no value provided then the default value is used
14. Nested functions
15. Functions can be returned from functions and can also be passed as parameters to other functions
16. Lambda functions
    1. lambdas are a concise way of writing the implementation of a function
    2. small, anonymous function that can take any number of arguments but only have 1 expression
    3. lambdas return an object or the reference to the function definition that can be called at a later stage
    4. lambda functions are restricted to s single expression and no need of return keyword it is implicitly evaluated
    5. syntax:  
       lambda arguments : body/ expression
17. map(func, iterable) -