System level custom modules

Type safety in python

Fastapi

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 complex
   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. For input output operations
   1. Print() – to send output to the console : sys.stdout
   2. Input() – to take input from the console: sys.stdin
7. Operators
   1. Arithmetic + - \* / % \*\* //
   2. Relational < > == <> <= >=
   3. Logical: and or not
   4. Membership: in, not in
8. Conditional constructs
   1. If-else
   2. Switch-case -> since 3.11
9. Looping constructs
   1. For
   2. While
   3. Jump statements break and continue
   4. Nested loops
10. 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
11. 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**
12. 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
    5. **NONLOCAL**  nested functions
13. eval() function that maintains the types of the value provided as input
14. 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
15. Nested functions
16. Functions can be returned from functions and can also be passed as parameters to other functions
17. 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
18. map(func, iterable) – that is used to transform the values in iterable and return a list of transformed values. The original list and the transformed list may or may not be of same type
19. filter(function, iterable) -> Returns those values from the iterable for which the function returns true
20. from functools import reduce(func, iterable, initial): it is used to reduce the result to a single value
21. File Handling:
    1. Open(filename, mode): mode : r(reading), w(writing), a+(append)
    2. Reading contents: read, readline , readlines, seek, tell
    3. Writing contents: write , writelines
    4. Close the file.
    5. CSV files: import csv module
22. Built-in modules: help(‘modules’): list of all the modules available  
     os, sys, time, datetime, random, math, statistics etc…
23. Custom modules: Every .py file in python is a module.
    1. To access functionalities of a module in another .py file, use import keyword  
       import <modulename>  
       from <modulename> import <function/ variable/ class>, <function/ variable/ class>  
       from <modulename> import \*
    2. Default python searches in current directory and the python path:  
       import sys  
       sys.path => is the path where python searches for the imported module, it is a list of string
    3. The module that are imported, python creates a .pyc file within a \_\_pycache\_\_ folder
    4. To access a module in other directory
       1. Import sys  
          sys.path.insert(0, ‘<path to your directory>’)
       2. In the command prompt/ terminal  
          MAC – export PATH=‘<path to your directory>’  
          Windows – set PATH=‘<path to your directory>’
    5. To add custom module in the system path, to make module available at system level
       1. Create folder and file structure as follows :   
          packagedemo -> sample -> area.py, messages.py, utility.py
       2. Create \_\_init\_\_.py file within sample folder
       3. Create setup.py file within packagedemo folder
       4. Then from the command prompt execute below command from within the **packagedemo folder  
          pip install .  
          [ NOTE : . refers to current directory ]**
    6. Exception handling: allows to handle the exception raised by a certain block of code and not stop the application abruptly.
       1. try : the block of code that you identify may throw an exception wrap it around the try block
       2. Try is always followed either by except or finally or both
       3. except : is used to handle the exception thrown by the try block
       4. else: is a clause that gets executed only when the try block does not throw an exception
       5. finally: block is executed irrespective if the try block throws an exception or not
       6. raise: used to manually throw an exception by the code to be handled by the user of the code
    7. Classes in Python:
       1. Use class keyword to create a class
       2. 2 magic methods:
          1. \_\_init\_\_() : is used to initialize the members of the class, and it internally calls \_\_new\_\_() method
          2. \_\_new\_\_() : creates the object of the class
          3. \_\_str\_\_() : to provide the string representation of the class object
       3. Class can have any number of methods which can be called using the . operator
       4. Self : it refers to current object instance
       5. To make the variables private , declare variable preceding by \_\_

Create a class Jumbled that will have following attributes and methods

words=[]

correctans=''

jumbledword=''

categories=[]

methods:

storeCategory(category;''): that will append the category in the file

storeWords(words:[], category:''): store words in the file for the respective category

readCategories(): will read list of categories from the file and store in categories[]

readWords(category):will read list of words from the file for the category and