In [2]:
```
from functools import reduce
import math
import datetime
```

In [ ]:
```
# 1. Filter users older than 25 and print only their names.
# users = [ {"name": "Amit", "age": 22}, {"name": "Priya", "age": 29}, {"nam
# 32}, {"name": "Meena", "age": 19} ]
```

In [6]:
```
users = [
    {"name": "Amit", "age": 22},
    {"name": "Priya", "age": 29},
    {"name": "Raj", "age": 32},
    {"name": "Meena", "age": 19}
]

res = list(map(lambda u: u["name"], filter(lambda u: u["age"] > 25, users)))
print(res)
```

```
['Priya', 'Raj']
```

In [ ]:
```
# 2. Clean list by removing empty/whitespace-only strings and stripping spac
# data = ["hello", " ", "", "python ", " dev ", ""]
```

In [7]:
```
data = ["hello", " ", "", "python ", " dev ", ""]

res = list(map(lambda x: x.strip(), filter(lambda x: x.strip() != "", data)))
print(res)
```

```
['hello', 'python', 'dev']
```

In [ ]:
```
# 3. Filter out invalid emails (without '@') and convert valid ones to upper
# spaces.
# emails = ["test@example.com", "invalid", " user@domain.com", "", "hello"]
```

In [8]:
```
emails = ["test@example.com", "invalid", " user@domain.com", "", "hello"]

res = list(map(lambda e: e.strip().upper(), filter(lambda e: "@" in e.strip(
print(res)
```

```
['TEST@EXAMPLE.COM', 'USER@DOMAIN.COM']
```

In [ ]:
```
# 4. Filter dictionary to include only items where the value is greater than
# data = {"a": 10, "b": 5, "c": 15, "d": 3}
```

In [9]:
```
data = {"a": 10, "b": 5, "c": 15, "d": 3}

res = dict(filter(lambda item: item[1] > 7, data.items()))
print(res)
```

```
{'a': 10, 'c': 15}
```

In [ ]:
```
# 5. Filter only students who have marks ≥ 50.
# students = [ {"name": "Ravi", "marks": 85}, {"name": "Riya", "marks": 40},
# "Neha", "marks": 76} ]
```

In [1]:
```
students = [
    {"name": "Ravi", "marks": 85},
    {"name": "Riya", "marks": 40},
    {"name": "Neha", "marks": 76}
]

res = list(filter(lambda s: s["marks"] >= 50, students))
print(res)
```

```
[{'name': 'Ravi', 'marks': 85}, {'name': 'Neha', 'marks': 76}]
```

In [ ]:
```
# 6. Filter numbers from 1 to 30 that are even and divisible by 3.
# nums = list(range(1, 31))
```

In [2]:
```
nums = list(range(1, 31))
res = list(filter(lambda n: n % 2 == 0 and n % 3 == 0, nums))
print(res)
```

```
[6, 12, 18, 24, 30]
```

In [ ]:
```
# 7. Filter the list to remove None and blank/whitespace-only strings.
# data = ["Hello", None, "", " ", "World", "Python "]
```

In [3]:
```
data = ["Hello", None, "", " ", "World", "Python "]
res = list(map(lambda x: x.strip(), filter(lambda x: x is not None and x.str
print(res)
```

```
['Hello', 'World', 'Python']
```

In [ ]:
```
# 8. Filter out blacklisted users from the user list.
# users = ["alice", "bob", "john", "admin", "eve"]
# blacklist = {"admin", "eve"}
```

In [5]:
```
users = ["alice", "bob", "john", "admin", "eve"]
blacklist = {"admin", "eve"}

res = list(filter(lambda u: u  in blacklist, users))
print(res)
```

```
['admin', 'eve']
```

In [ ]:
```
# 9. Use a filter to find all even numbers from a list of integers.
# nums = [1,2,3,4,5,6,7,8,9,10]
```

In [6]:
```
nums = [1,2,3,4,5,6,7,8,9,10]
res = list(filter(lambda n: n % 2 == 0, nums))
print(res)
```

```
[2, 4, 6, 8, 10]
```

In [ ]:
```
# 10. Use a filter to extract words that start with the letter "a".
# words = ["apple", "banana", "avocado", "grape", "apricot", "berry"]
```

In [7]:
```
words = ["apple", "banana", "avocado", "grape", "apricot", "berry"]
res = list(filter(lambda w: w.startswith("a"), words))
print(res)
```

```
['apple', 'avocado', 'apricot']
```

In [ ]:
```
# 11. Use a filter to remove all negative numbers.
# numbers = [-5, -2, 0, 3, 7, -1, 10]
```

In [8]:
```
numbers = [-5, -2, 0, 3, 7, -1, 10]
res = list(filter(lambda n: n >= 0, numbers))
print(res)
```

```
[0, 3, 7, 10]
```

In [ ]:
```
# 12. Given a list of numbers, return cubes of numbers greater than 5.
# nums = [2,4,6,8,10]
```

In [9]:
```
nums = [2,4,6,8,10]
res = list(map(lambda n: n**3, filter(lambda n: n > 5, nums)))
print(res)
```

```
[216, 512, 1000]
```

In [ ]:
```
# 13. Given a list of (name, age) tuples, find all people older than 18.
# people = [("Amit",17),("Priya",22),("Ravi",30),("Meena",16)]
```

In [10]:
```
people = [("Amit",17),("Priya",22),("Ravi",30),("Meena",16)]
res = list(filter(lambda p: p[1] > 18, people))
print(res)
```

```
[('Priya', 22), ('Ravi', 30)]
```

In [ ]:
```
# 14. Use a filter to extract palindromes.
# words = ["madam","apple","level","hello","noon"]
```

In [11]:
```
words = ["madam","apple","level","hello","noon"]
res = list(filter(lambda w: w == w[::-1], words))
print(res)
```

```
['madam', 'level', 'noon']
```

In [ ]:
```
# 15. Use a filter to get perfect squares.
# nums = [1,2,4,5,9,10,16,20]
```

In [14]:
```
nums = [1,2,4,5,9,10,16,20]
res = list(filter(lambda n: int(math.sqrt(n))**2 == n, nums))
print(res)
```

```
[1, 4, 9, 16]
```

In [ ]:
```
# 16. Use a filter to remove all empty strings.
# words = ["python","","map"," ","reduce",""]
```

In [15]:
```
words = ["python", "", "map", " ", "reduce", ""]
res = list(filter(lambda w: w.strip() != "", words))
print(res)
```

```
['python', 'map', 'reduce']
```

In [ ]:
```
# 17. Use a filter to find words longer than 5 characters.
```

```
# words = ["python","java","developer","ai","filter","map"]
```

In [16]:
```
words = ["python", "java", "developer", "ai", "filter", "map"]
res = list(filter(lambda w: len(w) > 5, words))
print(res)
```

```
['python', 'developer', 'filter']
```

In [ ]:
```
# 18. Use filter to select only those dictionaries where status = active.
# users = [ {"name":"Alice","status":"active"}, {"name":"Bob","status":"inac
# {"name":"Charlie","status":"active"} ]
```

In [17]:
```
users = [
    {"name": "Alice", "status": "active"},
    {"name": "Bob", "status": "inactive"},
    {"name": "Charlie", "status": "active"}
]

res = list(filter(lambda u: u["status"] == "active", users))
print(res)
```

```
[{'name': 'Alice', 'status': 'active'}, {'name': 'Charlie', 'status': 'activ
e'}]
```

In [ ]:
```
# 19. Use filter and map to get names of employees whose salary > 50,000.
# employees = [ {"name":"Amit","salary":45000}, {"name":"Priya","salary":600
# {"name":"Raj","salary":75000} ]
```

In [18]:
```
employees = [
    {"name": "Amit", "salary": 45000},
    {"name": "Priya", "salary": 60000},
    {"name": "Raj", "salary": 75000}
]

res = list(map(lambda e: e["name"], filter(lambda e: e["salary"] > 50000, em
print(res)
```

```
['Priya', 'Raj']
```

In [20]:
```
# 20. Mask mobile numbers showing only last 4 digits.
# numbers = ["9876543210","9123456789","9988776655"]
```

In [21]:
```
numbers = ["9876543210", "9123456789", "9988776655"]
res = list(map(lambda n: "*" * (len(n) - 4) + n[-4:], numbers))
print(res)
```

```
['******3210', '******6789', '******6655']
```

In [ ]:
```
#  21. Extract file extensions from filenames.
# files = ["data.csv","image.jpeg","report.pdf","main.py"]
```

In [22]:
```
files = ["data.csv", "image.jpeg", "report.pdf", "main.py"]
res = list(map(lambda f: f.split(".")[-1] if "." in f else "", files))
print(res)
```

```
['csv', 'jpeg', 'pdf', 'py']
```

In [ ]:
```
# 22. Convert list of prices from USD to INR (1 USD = 83.20).
# prices_usd = [10,15.5,100,75.25]
```

In [23]:
```
prices_usd = [10, 15.5, 100, 75.25]
res = list(map(lambda p: round(p * 83.20, 2), prices_usd))
print(res)
```

```
[832.0, 1289.6, 8320.0, 6260.8]
```

In [ ]:
```
#  23. Remove special characters from names.
# names = ["R@hul","N!kita","Ami#t","P&riya"]
```

In [25]:
```
names = ["R@hul", "N!kita", "Ami#t", "P&riya"]
res = list(map(lambda n: ''.join(ch for ch in n if ch.isalpha()), names))
print(res)
```

```
['Rhul', 'Nkita', 'Amit', 'Priya']
```

In [ ]:
```
# 24. Add 5% GST to each product price.
# prices = [200,450,1000,150]
```

In [26]:
```
prices = [200, 450, 1000, 150]
res = list(map(lambda p: round(p * 1.05, 2), prices))
print(res)
```

```
[210.0, 472.5, 1050.0, 157.5]
```

In [ ]:
```
# 25. Convert full names to email usernames.
# names = ["Amit Sharma","Neha Rajput","Ravi Kumar"]
```

In [27]:
```
names = ["Amit Sharma", "Neha Rajput", "Ravi Kumar"]
res = list(map(lambda n: n.lower().replace(" ", "."), names))
print(res)
```

```
['amit.sharma', 'neha.rajput', 'ravi.kumar']
```

In [ ]:
```
# 26. Convert datetime strings to just date (YYYY-MM-DD).
# timestamps = ["2024-06-15 14:30:00","2024-12-01 09:45:10","2025-01-10 23:5
```

In [28]:
```
timestamps = ["2024-06-15 14:30:00", "2024-12-01 09:45:10", "2025-01-10 23:5
res = list(map(lambda ts: ts.split()[0], timestamps))
print(res)
```

```
['2024-06-15', '2024-12-01', '2025-01-10']
```

In [ ]:
```
# 27. Create hashtags from words.
# tags = ["python","devlife","code","bugfix"]
```

In [29]:
```
tags = ["python", "devlife", "code", "bugfix"]
res = list(map(lambda t: "#" + t, tags))
print(res)
```

```
['#python', '#devlife', '#code', '#bugfix']
```

In [ ]:
```
# 28. Convert binary strings to decimal.
# binaries = ["1010","111","1001","1100"]
```

In [30]:
```
binaries = ["1010", "111", "1001", "1100"]
res = list(map(lambda b: int(b, 2), binaries))
print(res)
```

```
[10, 7, 9, 12]
```

In [ ]:
```
# 30. Write a map function to square every number in a list.
# nums = [1,2,3,4,5]
```

In [31]:
```
nums = [1, 2, 3, 4, 5]
res = list(map(lambda n: n * n, nums))
print(res)
```

```
[1, 4, 9, 16, 25]
```

In [ ]:
```
# 31. Convert list of strings into uppercase.
# words = ["python","map","filter","reduce"]
```

In [32]:
```
words = ["python", "map", "filter", "reduce"]
res = list(map(lambda w: w.upper(), words))
print(res)
```

```
['PYTHON', 'MAP', 'FILTER', 'REDUCE']
```

In [ ]:
```
# 32. Get length of each word in a sentence.
# sentence = "Python makes coding easy"
```

In [34]:
```
sentence = "Python makes coding easy"
words = sentence.split()
res = list(map(lambda w: len(w), words))
print(res)
```

```
[6, 5, 6, 4]
```

In [ ]:
```
# 33. Convert integers into strings with prefix Num_.
# nums = [10,20,30,40]
```

In [35]:
```
nums = [10, 20, 30, 40]
res = list(map(lambda n: f"Num_{n}", nums))
print(res)
```

```
['Num_10', 'Num_20', 'Num_30', 'Num_40']
```

In [ ]:
```
# 34. Convert Celsius to Fahrenheit.
# temps_c = [0,25,37,100]
```

In [36]:
```
temps_c = [0, 25, 37, 100]
res = list(map(lambda c: round((c * 9/5) + 32, 2), temps_c))
print(res)
```

```
[32.0, 77.0, 98.6, 212.0]
```

In [ ]:
```
# 35. Replace spaces in strings with "-".
# words = ["hello world","python code","filter map"]
```

In [37]:
```
words = ["hello world", "python code", "filter map"]
res = list(map(lambda w: w.replace(" ", "-"), words))
```

```
print(res)
```

```
['hello-world', 'python-code', 'filter-map']
```

In [ ]:
```
# 36. Reciprocal of each number in a list (skip zeros).
# nums = [2,4,5,0,10]
```

In [38]:
```
nums = [2, 4, 5, 0, 10]
res = list(map(lambda n: 1/n, filter(lambda n: n != 0, nums)))
print(res)
```

```
[0.5, 0.25, 0.2, 0.1]
```

In [ ]:
```
# 37. Reverse each word in a sentence.
# sentence = "Python is powerful"
```

In [39]:
```
sentence = "Python is powerful"
words = sentence.split()
res = list(map(lambda w: w[::-1], words))
print(res)
```

```
['nohtyP', 'si', 'lufrewop']
```

In [ ]:
```
# 38. Capitalize only first letter of each word.
# words = ["python","coding","rocks"]
```

In [40]:
```
words = ["python", "coding", "rocks"]
res = list(map(lambda w: w.capitalize(), words))
print(res)
```

```
['Python', 'Coding', 'Rocks']
```

In [ ]:
```
# 39. Extract first element from tuples.
# pairs = [(1,"a"),(2,"b"),(3,"c")]
```

In [41]:
```
pairs = [(1, "a"), (2, "b"), (3, "c")]
res = list(map(lambda p: p[0], pairs))
print(res)
```

```
[1, 2, 3]
```

In [ ]:
```
# 40. Apply 10% discount to all prices.
# products = [("Book",200),("Laptop",50000),("Pen",20)]
```

In [42]:
```
products = [("Book", 200), ("Laptop", 50000), ("Pen", 20)]
res = list(map(lambda p: (p[0], round(p[1] * 0.9, 2)), products))
print(res)
```

```
[('Book', 180.0), ('Laptop', 45000.0), ('Pen', 18.0)]
```

In [ ]:
```
# 41. Obfuscate email addresses (replace characters before @ with *).
# emails = ["test@example.com","user@domain.com"]
```

In [43]:
```
emails = ["test@example.com", "user@domain.com"]
res = list(map(lambda e: "*" * e.index("@") + e[e.index("@"):], emails))
print(res)
```

```
['****@example.com', '****@domain.com']
```

In [ ]:
```
# 42. Extract domain names from email addresses.
# emails = ["test@example.com","hello@gmail.com","world@yahoo.in"]
```

In [44]:
```
emails = ["test@example.com", "hello@gmail.com", "world@yahoo.in"]
res = list(map(lambda e: e.split("@")[-1], emails))
print(res)
```

```
['example.com', 'gmail.com', 'yahoo.in']
```

In [ ]:
```
# 43. Factorial of each number in a list.
# nums = [3,4,5]
```

In [1]:
```
nums = [3, 4, 5]
res = list(map(lambda n: reduce(lambda a, b: a * b, range(1, n + 1)), nums))
print(res)
```

```
[6, 24, 120]
```

In [ ]:
```
# 44. Use reduce to find the sum of all elements in a list.
# nums = [1,2,3,4,5]
```

In [3]:
```
nums = [1, 2, 3, 4, 5]
res = reduce(lambda a, b: a + b, nums)
print(res)
```

```
15
```

In [ ]:
```
# 45. Use reduce to find the maximum number in a list.
# nums = [10,45,32,99,5]
```

In [4]:
```
nums = [10, 45, 32, 99, 5]
res = reduce(lambda a, b: a if a > b else b, nums)
print(res)
```

```
99
```

In [ ]:
```
# 46. Use reduce to calculate the product of all numbers in a list.
# nums = [2,3,4,5]
```

In [6]:
```
nums = [2, 3, 4, 5]
res = reduce(lambda a, b: a * b, nums)
print(res)
```

```
120
```

In [ ]:
```
# 47. Use reduce to concatenate a list of strings.
# words = ["Python","is","fun"]
```

In [7]:
```
words = ["Python", "is", "fun"]
res = reduce(lambda a, b: a + " " + b, words)
print(res)
```

```
Python is fun
```

In [ ]:
```
# 48. Use reduce to find the longest string in a list of words.
```

```
# words = ["apple","banana","watermelon","grape"]
```

In [8]:
```
words = ["apple", "banana", "watermelon", "grape"]
res = reduce(lambda a, b: a if len(a) > len(b) else b, words)
print(res)
```

```
watermelon
```

In [ ]:
```
# 49. Use reduce to count total characters in a list of words.
# words = ["Python","map","reduce"]
```

In [9]:
```
words = ["Python", "map", "reduce"]
res = reduce(lambda a, b: a + b, map(lambda w: len(w), words))
print(res)
```

```
15
```

In [ ]:
```
# 50. Use reduce to merge a list of dictionaries into one.
# dicts = [{"a":1},{"b":2},{"c":3}]
```

In [10]:
```
dicts = [{"a": 1}, {"b": 2}, {"c": 3}]
res = reduce(lambda a, b: {**a, **b}, dicts, {})
print(res)
```

```
{'a': 1, 'b': 2, 'c': 3}
```

In [ ]:
```
# 51. Use reduce to calculate the final balance from transactions.
# transactions = ["+200","-100","+50","-30"]
```

In [11]:
```
transactions = ["+200", "-100", "+50", "-30"]
res = reduce(lambda a, b: a + int(b), transactions, 0)
print(res)
```

```
120
```

In [ ]:
```

```