```
# 1. Adventurer's Backpack Problem: You are an adventurer organizing a
backpack. Write a function that takes a list of items, uses a lambda
to filter only items of type str that have more than 4 characters, and
returns the filtered list. What happens if you accidentally include
numbers and None in your list? Test this with ["sword", 5, None,
"map", "lantern"].
# l=list(filter(lambda x:len(x)>5,["sword",5, None, "map",
"lantern"]))
# print(l)
# output:
# it throws an TypeError that Int or None object not hav len()

# 2. Treasure Chest Calculation: A treasure chest has gold coins in a
specific numerical pattern. If the coins in each row form a pattern
like:

#    1
#    2 3
#    4 5 6

#    Write a program using nested loops to generate and print the
first n rows of this pattern. Use enumerate to label each row.
# n=int(input("Enter no row "))
# z=1
# for i,k in enumerate(range(n),start=1):
#     print(i,end=" ")
#     for j in range(k+1):
#         print(z,end=" ")
#         z+=1
#     print()
# output:
# Enter no row 3
# 1 1
# 2 2 3
# 3 4 5 6
# # 3. The Password Validator: Write a function that checks if a
password meets the following conditions: contains at least one number,
one uppercase letter, and one special character (!, @, #, etc.). Use a
comprehension to filter invalid passwords from a list of passwords.
For example: ["Hello123", "world!@#", "Python"].
# l=["Hello123", "world!@#",'Lava@2203']
# def checkPassword(x):
#     l=[]
#     sp=0
#     lw=0
#     hp=0
#     np=0
#     for i in x:
#         if i in '!@#$%^&*()-.~`':
#             sp=1
#         elif ord(i)>=97 and ord(i)<=122:
#             lw=1
#         elif ord(i)>=65 and ord(i)<=90:
#              hp=1
#         elif int(i) in [1,2,3,4,5,6,7,8,9,0]:
#             np=1
#     if np==1 and hp==1 and lw==1 and sp==1:
#         return (i,"Login")
#     else:
#         return (x,"Login-Failed")


# lst=list(map(checkPassword,l))
# print(lst)
# output:[('Hello123', 'Login-Failed'), ('world!@#', 'Login-Failed'),
('3', 'Login')]
```

```python
# 4. Mystic Numbers: You encounter a scroll that says, “If a number is
divisible by 3, it’s mystic; if by 5, it’s legendary.” Write a program
using loops and conditionals that prints "Mystic" for multiples of 3,
"Legendary" for multiples of 5, and "Mystic-Legendary" for multiples
of both in the range 1-50.
'''
for i in range(1,50):
    if i%3==0:
        if(i%5==0):
            print(i,'Mystic-Legendary')
        else:
            print(i,'Mystic')
    elif(i%5==0):
        print(i,'Legendary')
        '''
# 3 Mystic
# 5 Legendary
# 6 Mystic
# 9 Mystic
# 10 Legendary
# 12 Mystic
# 15 Mystic-Legendary
# So on...


# 5. Treasure Map Grid: A treasure map is represented as a 2D list.
Write a comprehension to flatten the map into a single list. Example
map: [[1, 2], [3, 4], [5, 6]]. After flattening, how would you sum all
the treasures using a function and lambda?
# from functools import reduce
# l=[[1, 2], [3, 4], [5, 6]]
# lst=[j for i in l for j in i]
# print(lst)
# sum=reduce(lambda x,y:x+y,lst)
# print("Sum",sum)
# output:
# [1, 2, 3, 4, 5, 6]
# Sum 21
# 6. Treasure Hunter Function: Create a function that calculates the
sum of even treasures (numbers) from a given list. Then, pass this
function to a higher-order function that doubles the total sum and
returns the result. Use this on [4, 7, 12, 19, 22].
'''
def fun(l):
    sum=0
    for i in l:
        if i%2==0:
            sum+=i
    return sum
def HOF(fun):
    d=fun([4, 7, 12, 19, 22])
    return 2*d
print("Return Sum",HOF(fun))
'''
# output:
# 76




# 7. Animal Kingdom Tasks: You have a list of animals and their
habitats: ["Lion:Savana", "Shark:Ocean", "Eagle:Mountain"]. Write a
program to split the animals and habitats into two separate lists
using zip. How would you handle inconsistent entries like
"Fox,Forest"?
# l= ["Lion:Savana", "Shark:Ocean", "Eagle:Mountain",'A,B','Cat-
Sleep']
# name=[]
# habbit=[]
# for i in l:

#       for j in i:
#          if j==':':
#           temp=i.split(":")
#           name.append(temp[0])
#           habbit.append(temp[1])
#           break
#          elif j==',':
#                  t=i.split(",")
#                  name.append(t[0])
#                  habbit.append(t[1])
#                  break
#          elif j=='-':
#                  temp=i.split("-")
#                  name.append(temp[0])
#                  habbit.append(temp[1])
#                  break
# print(name)
# print(habbit)
# print(list(zip(name,habbit)))
# output:
# ['Lion', 'Shark', 'Eagle', 'A', 'Cat']
# ['Savana', 'Ocean', 'Mountain', 'B', 'Sleep']
# [('Lion', 'Savana'), ('Shark', 'Ocean'), ('Eagle', 'Mountain'),
('A', 'B'), ('Cat', 'Sleep')]


# 8. Village Lamp Posts: A village has a series of lamp posts numbered
sequentially. Each odd-numbered post should glow red, and each even-
numbered post should glow green. Write a function to generate this
pattern for n posts using a loop and enumerate.
'''
# n=int(input("Enter no "))
# for i,k in enumerate(range(1,n+1),start=1):
#     if i%2==0:
#         print(i,'Red')
#     else:
#         print(i,'Green')
# output:
# Enter no 5
# 1 Green
# 2 Red
# 3 Green
# 4 Red
# 5 Green
'''
```

```
# 9. Inventory Eval: A store inventory system evaluates items using a
custom string formula like "20 + 30 - 10". Write a program that uses
eval to calculate the total value of each item string. How would you
secure this code from malicious inputs?
# s=input("Enter String formulla ")
# try:
#     print(s,"=",eval(s))
# except:
#     print("Wrong Input")
# output:
# Enter String formulla 10+20
# 10+20 = 30
# Enter String formulla abc+90
# Wrong Input


# 10. Potion Effectiveness: Write a function that takes a list of
potions' effectiveness percentages (e.g., [85, 90, 78]) and returns
their average using a lambda function. Now write a comprehension to
filter potions with effectiveness above 80%.
# l=[85, 90, 78,50,67,65]
# from functools import reduce
# sum=reduce(lambda x,y:x+y,l)
# avg=sum/len(l)
# print("avarge potion",avg)
# lst=list(filter(lambda x:x>80,l))
# print("potions with effectiveness above",lst)
# output:
# avarge potion 72.5
# potions with effectiveness above [85, 90]
# 11. Car Speed Tracker:
#     Create a class Car with an attribute speed. Add a method
set_speed to update the speed and another method get_speed to retrieve
it. Create an object of the class, set its speed to 80, and print the
current speed.
# class Car:
#     def __init__(self,speed):
#         self.speed=speed
#     def set_speed(self,speed):
#         self.speed=speed
#     def get_speed(self):
#         print('Current Speed',self.speed)
# c=Car(20)
# c.get_speed()
# c.set_speed(60)
# c.get_speed()
# output:
# Current Speed 20
# Current Speed 60


# 12. Book Tracker:
#   Create a class Book with attributes title and author. Add a method
display_details to print the book's details in the format "Title:
<title>, Author: <author>". Create an object, assign it a title and
author, and display the details using the method.
'''
class Book:
    def __init__(self,title,author):
        self.title=title
        self.author=author
    def display_details(self):
        print(f'Title: {self.title}, Author: {self.author}')
b=Book('1984','DR jarvin')
b.display_details()
'''
# output:
# Title: 1984, Author: DR jarvin


# 13. Bank Account Manager:
#     Write a class BankAccount with an attribute balance. Add methods
deposit and withdraw to increase and decrease the balance,
respectively. Create an account object, deposit 1000, withdraw 300,
and print the remaining balance.
"'
class Bank:
    def __init__(self,balance):
        self.balance=balance
    def Balance(self):
        print("Current Balance:",{self.balance})
    def Deposit(self):
        amt=int(input("Enter amout "))
        self.balance+=amt
    def Withdrawl(self):
        amt=int(input("Enter amount "))
        if amt>self.balance:
            print("Insuffient Balance")
        else:
            self.balance-=amt

print("""Welcome to Bank:
      1-Check Balance
      2-Deposit
      3-withdrawl
      4-Exit
      """)
ch=int(input("Enter your Choice: "))
bnk=Bank(0)
while(ch!=4):
    if ch==1:
        bnk.Balance()
    elif ch==2:
        bnk.Deposit()
    elif ch==3:
        bnk.Withdrawl()
    ch=int(input("Enter your Choice "))
'''

'''
Output:
    Welcome to Bank:
      1-Check Balance
      2-Deposit
      3-withdrawl
      4-Exit

Enter your Choice: 2
Enter amout 1000
Enter your Choice 3
Enter amount 300
Enter your Choice 1
Current Balance: {700}
Enter your Choice
'''
```

```
# 14. Simple Calculator:
#     Create a class Calculator with methods add, subtract, multiply,
and divide. Each method should take two arguments and return the
result of the operation. Create an object of the class and call each
method with different values.
# class cal:
#     def add(self):
#         a=int(input("Enter numbee1 "))
#         b=int(input("Enter numbee1 "))
#         return a+b
#     def sub(self):
#         a=int(input("Enter numbee1 "))
#         b=int(input("Enter numbee1 "))
#         return a-b
#     def mul(self):
#         a=int(input("Enter numbee1 "))
#         b=int(input("Enter numbee1 "))
#         return a*b
#     def div(self):
#         a=int(input("Enter numbee1 "))
#         b=int(input("Enter numbee1 "))
#         return a/b
# obj=cal()
# print("Sum",obj.add())
# print("Sub",obj.sub())
# print("Mul",obj.mul())
# print("Div",obj.div())
# output:
# Enter numbee1 10
# Enter numbee1 20
# Sum 30
# Enter numbee1 20
# Enter numbee1 10
# Sub 10
# Enter numbee1 10
# Enter numbee1 10
# Mul 100
# Enter numbee1 20
# Enter numbee1 10
# Div 2.0
# 15. Student Grade Tracker:
#    Write a class Student with attributes name and grade. Add a
method update_grade to modify the grade and another method
display_student to print the student’s name and grade. Create an
object, update the grade to "A+", and display the details.
'''
class student:
    def __init__(self,name,grade):
        self.name=name
        self.grade=grade
    def update_grade(self,grade):
            self.grade=grade
    def display_student(self):
            print(f'{self.name} {self.grade}')
s=student('lavanya','B')
s.display_student()
s.update_grade('A+')
s.display_student()
'''
# output:
# lavanya B
# lavanya A+
```