**Exercise 1: Squaring Numbers**

**Create a lambda function that takes a number and returns its square.**

**Solution:**

square = lambda x: x \*\* 2

result = square(5) # Expected output: 25

**Exercise 2: Doubling Numbers in a List**

**Use map to double each number in a list using a lambda function.**

**Solution:**

numbers = [1, 2, 3, 4, 5]

doubled\_numbers = list(map(lambda x: x \* 2, numbers)) # Expected output: [2, 4, 6, 8, 10]

**Exercise 3: Filtering Even Numbers**

**Use filter to get only even numbers from a list using a lambda function.**

**Solution:**

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even\_numbers = list(filter(lambda x: x % 2 == 0, numbers)) # Expected output: [2, 4, 6, 8, 10]

**Exercise 4: Finding the Maximum in a List of Tuples**

**Given a list of tuples, use max with a lambda function to find the tuple with the highest second value.**

**Solution:**

tuples = [(1, 5), (2, 3), (3, 10), (4, 1)]

max\_tuple = max(tuples, key=lambda x: x[1]) # Expected output: (3, 10)

**Exercise 5: Sorting a List of Dictionaries**

**Sort a list of dictionaries based on a specific key using a lambda function.**

**Solution:**

people = [

{"name": "Alice", "age": 30},

{"name": "Bob", "age": 25},

{"name": "Charlie", "age": 35}

]

sorted\_people = sorted(people, key=lambda x: x["age"])

# Expected output: [{"name": "Bob", "age": 25}, {"name": "Alice", "age": 30}, {"name": "Charlie", "age": 35}]

**Exercise 6: Using Lambda with reduce**

**Use reduce to find the product of all numbers in a list using a lambda function. You'll need to import reduce from functools.**

**Solution:**

from functools import reduce

numbers = [1, 2, 3, 4, 5]

product = reduce(lambda x, y: x \* y, numbers) # Expected output: 120

**Exercise 7: Extracting Last Names from a List**

**Given a list of full names, extract the last names using a lambda function.**

**Solution:**

names = ["John Doe", "Jane Smith", "Alice Johnson"]

last\_names = list(map(lambda x: x.split()[-1], names)) # Expected output: ["Doe", "Smith", "Johnson"]

**Exercise 8: Summing Odd Numbers**

**Use a lambda function with filter to get the sum of all odd numbers from a list.**

**Solution:**

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

odd\_numbers = list(filter(lambda x: x % 2 != 0, numbers))

sum\_odd = sum(odd\_numbers) # Expected output: 25

**Exercise 9: Creating a List of Cubes**

**Create a list of cubes from a list of numbers using a lambda function with map.**

**Solution:**

numbers = [1, 2, 3, 4, 5]

cubes = list(map(lambda x: x \*\* 3, numbers)) # Expected output: [1, 8, 27, 64, 125]

**Exercise 10: Checking for Palindromes**

**Given a list of words, use a lambda function with filter to find all palindromic words (words that are the same forwards and backwards).**

**Solution:**

words = ["level", "world", "radar", "python", "rotor"]

palindromes = list(filter(lambda x: x == x[::-1], words)) # Expected output: ["level", "radar", "rotor"]

**Exercise 11: Grouping by First Character**

**Group a list of strings by their first character using a lambda function and groupby from itertools.**

**Solution:**

from itertools import groupby

words = ["apple", "banana", "apricot", "berry", "avocado", "blueberry"]

grouped = {key: list(group) for key, group in groupby(sorted(words), lambda x: x[0])}

# Expected output: {'a': ['apple', 'apricot', 'avocado'], 'b': ['banana', 'berry', 'blueberry']}

**Exercise 12: Mapping Function with Conditional Logic**

**Create a lambda function to return "positive" for positive numbers, "negative" for negative numbers, and "zero" for zero.**

**Solution:**

numbers = [-5, 0, 3, 7, -2]

signs = list(map(lambda x: "positive" if x > 0 else "negative" if x < 0 else "zero", numbers))

# Expected output: ["negative", "zero", "positive", "positive", "negative"]

**Exercise 13: Converting to Uppercase**

**Use a lambda function with map to convert a list of strings to uppercase.**

**Solution:**

words = ["hello", "world", "python", "lambda"]

uppercased\_words = list(map(lambda x: x.upper(), words))

# Expected output: ["HELLO", "WORLD", "PYTHON", "LAMBDA"]

**Exercise 14: Calculating Factorials**

**Given a list of integers, use a lambda function to calculate the factorial for each integer. This will require using reduce from functools.**

**Solution:**

from functools import reduce

def factorial(n):

return reduce(lambda x, y: x \* y, range(1, n + 1))

numbers = [3, 4, 5]

factorials = list(map(lambda x: factorial(x), numbers)) # Expected output: [6, 24, 120]

**Exercise 15: Finding the Longest String**

**Use a lambda function with max to find the longest string in a list of strings.**

**Solution:**

words = ["apple", "banana", "cherry", "date"]

longest\_word = max(words, key=lambda x: len(x))

# Expected output: "banana"

**Exercise 16: Filtering Names by Initial**

**Use a lambda function with filter to get all names that start with a specific letter from a list of names.**

**Solution:**

names = ["Alice", "Bob", "Charlie", "David", "Eve"]

filtered\_names = list(filter(lambda x: x.startswith("A"), names))

# Expected output: ["Alice"]

**Exercise 17: Adding Lists Element-wise**

**Use a lambda function with map to add elements from two lists element-wise.**

**Solution:**

list1 = [1, 2, 3]

list2 = [4, 5, 6]

sum\_lists = list(map(lambda x, y: x + y, list1, list2))

# Expected output: [5, 7, 9]

**Exercise 18: Counting Characters in a String**

**Use a lambda function with map to count the number of characters in each string in a list.**

**Solution:**

words = ["hello", "world", "python"]

char\_counts = list(map(lambda x: len(x), words))

# Expected output: [5, 5, 6]

**Exercise 19: Removing Empty Strings**

**Use a lambda function with filter to remove empty strings from a list.**

**Solution:**

strings = ["hello", "", "world", "", "python"]

non\_empty\_strings = list(filter(lambda x: x != "", strings))

# Expected output: ["hello", "world", "python"]

**Exercise 20: Checking Divisibility by a Number**

**Use a lambda function with filter to find all numbers divisible by a given divisor from a list.**

**Solution:**

numbers = [10, 20, 25, 30, 35, 40, 45, 50]

divisible\_by\_five = list(filter(lambda x: x % 5 == 0, numbers))

# Expected output: [10, 20, 25, 30, 35, 40, 45, 50]