perplexity

create a pdf having the solutions of all the programs in python

Okay, I will create a PDF document containing the solutions to the Python programming assignments you've provided.

Python Assignment Solutions

Here are the solutions to the assignment questions, along with explanations:

1. Lambda Functions & Map

Problem: Write a Python program using a lambda function that takes a list of numbers and squares each number. Use the `map()` function to apply this lambda function to a range of numbers from 1 to 20.

```
# Solution:
numbers = range(1, 21)
squared_numbers = list(map(lambda x: x**2, numbers))
print(squared_numbers)
```

Explanation:

- We create a range of numbers from 1 to 20 using `range(1, 21)`.
- We define a lambda function `lambda x: x**2` that takes a number `x` and returns its square.
- We use the `map()` function to apply the lambda function to each number in the `numbers` range. `map()` returns an iterator, so we convert it to a list using `list()`.
- Finally, we print the list of squared numbers.

2. Custom Module Creation and Importing

Problem: Create a Python module named `utilities.py` containing a function `factorial(num)` that returns the factorial of a number. Then, write a script in another file that imports this module and uses the function to print the factorial of numbers from 1 to 10.

utilities.py:

```
# utilities.py
def factorial(num):
    """
    This function calculates the factorial of a number.
    """
    if num == 0:
        return 1
    else:
        return num * factorial(num-1)
```

main.py:

```
# main.py
import utilities

for i in range(1, 11):
    print(f"Factorial of {i} is {utilities.factorial(i)}")
```

Explanation:

- utilities.py**:** This file defines a function `factorial(num)` that calculates the factorial of a given number using recursion.
- main.py**:** This file imports the `utilities` module. It then iterates through numbers from 1 to 10, calculates the factorial of each number using the `utilities.factorial()` function, and prints the result. Make sure both files are in the same directory.

3. Exception Handling in Functions

Problem: Define a function `divide(x, y)` which attempts to divide two numbers. Include exception handling to catch and print messages for cases like division by zero or if the input values are not numbers.

```
def divide(x, y):
    """
    This function divides two numbers and handles potential exceptions.
    """
    try:
        result = x / y
        print(f"{x} / {y} = {result}")
    except ZeroDivisionError:
        print("Error: Cannot divide by zero.")
    except TypeError:
        print("Error: Invalid input. Please enter numbers only.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

# Example Usage:
divide(10, 2)
divide(10, 0)
divide(10, "a")
```

Explanation:

- The `divide(x, y)` function attempts to divide `x` by `y`.

- The `try...except` block handles potential exceptions:
  - `ZeroDivisionError`: Catches the error when dividing by zero.
  - `TypeError`: Catches the error when the input is not a number.
  - `Exception as e`: Catches any other unexpected errors and prints the error message.

4. Using External Modules (NumPy)

Problem: Install the NumPy module using pip (provide the command for installation). Write a Python script to generate a 2×3 matrix filled with random numbers between 0 and 10 using NumPy, and then compute the transpose of the matrix.

Installation:

To install NumPy, use the following command in your terminal or command prompt:

```
pip install numpy
```

Code:

```
import numpy as np

# Generate a 2x3 matrix with random numbers between 0 and 10
matrix = np.random.randint(0, 11, size=(2, 3))
print("Original Matrix:")
print(matrix)

# Compute the transpose of the matrix
transpose = matrix.T
print("\nTranspose of the Matrix:")
print(transpose)
```

Explanation:

- We import the NumPy library as `np`.
- `np.random.randint(0, 11, size=(2, 3))` generates a 2×3 matrix with random integers between 0 (inclusive) and 11 (exclusive).
- `matrix.T` computes the transpose of the matrix.
- We print the original matrix and its transpose.

5. Lambda Function with Filter

Problem: Use a lambda function to filter out odd numbers from a list ranging from 1 to 50. Use the `filter()` function and print the result.

```
# Solution:
numbers = range(1, 51)
```

```
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)
```

Explanation:

- We create a range of numbers from 1 to 50 using `range(1, 51)`.
- We define a lambda function `lambda x: x % 2 == 0` that takes a number `x` and returns `True` if it's even (i.e., the remainder when divided by 2 is 0) and `False` otherwise.
- We use the `filter()` function to apply the lambda function to each number in the `numbers` range. `filter()` returns an iterator containing only the elements for which the lambda function returned `True`. We convert it to a list using `list()`.
- Finally, we print the list of even numbers.

⁂