1. 1. Implement all list operations

```
my_list = [1, 2, 3, 4, 5]
my_list.append(6)          # Add an item
my_list.insert(0, 0)       # Insert at index
my_list.remove(3)          # Remove an item
my_list.pop()              # Remove last item
my_list.reverse()          # Reverse the list
sorted_list = sorted(my_list) # Sort the list
print("List:", my_list)
print("Sorted List:", sorted_list)
```

2. 2. Implement all dictionary operations

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
my_dict['d'] = 4           # Add new key-value pair
my_dict.update({'e': 5})   # Update dictionary
del my_dict['a']           # Delete key-value pair
keys = my_dict.keys()      # Get all keys
values = my_dict.values()  # Get all values
print("Dictionary:", my_dict)
print("Keys:", keys)
print("Values:", values)
```

3. 3. Implement all set operations

```
set1 = {1, 2, 3}

set2 = {3, 4, 5}

union = set1 | set2        # Union

intersection = set1 & set2  # Intersection

difference = set1 - set2    # Difference

set1.add(6)                # Add element

set1.remove(2)             # Remove element

print("Union:", union)

print("Intersection:", intersection)

print("Difference:", difference)
```

4. 4. Implement all tuple operations

```
my_tuple = (1, 2, 3, 4, 5)

first_element = my_tuple[0]  # Access element

slice_tuple = my_tuple[1:4]  # Slicing

length = len(my_tuple)       # Get length

print("Tuple:", my_tuple)

print("Sliced Tuple:", slice_tuple)

print("Length:", length)
```

5. 5. Implement all string operations

```
my_str = "Hello, Python"

upper = my_str.upper()        # Convert to uppercase

lower = my_str.lower()        # Convert to lowercase
```

```
split_str = my_str.split()    # Split into list
join_str = ' '.join(split_str) # Join list into string
print("Upper:", upper)
print("Lower:", lower)
print("Split:", split_str)
print("Joined:", join_str)
```

6. 6. Display Armstrong numbers in the range from 1 to 1000

```
for num in range(1, 1001):
    order = len(str(num))
    sum_of_powers = sum(int(digit)**order for digit in str(num))
    if num == sum_of_powers:
        print(num, end=" ")
```

7. 7. Display prime numbers in the range from 1 to 100

```
for num in range(2, 101):
    if all(num % i != 0 for i in range(2, int(num**0.5) + 1)):
        print(num, end=" ")
```

8. 8. Print the nth Fibonacci number (n given by user)

```
n = int(input("Enter n: "))
a, b = 0, 1
```

```
for _ in range(n):
    a, b = b, a + b
print("Nth Fibonacci number:", a)
```

9. 9. Perform matrix addition and multiplication (user inputs M and N)

```
import numpy as np
M, N = map(int, input("Enter rows and columns M N: ").split())
matrix1 = np.random.randint(1, 10, (M, N))
matrix2 = np.random.randint(1, 10, (M, N))
matrix_addition = matrix1 + matrix2
print("Matrix Addition:\n", matrix_addition)
matrix_multiplication = matrix1 * matrix2
print("Matrix Multiplication:\n", matrix_multiplication)
```

10. 10. Implement linear search on 20 random generated numbers

```
import random
nums = random.sample(range(100), 20)
target = int(input("Enter number to search: "))
found = False
for i, num in enumerate(nums):
    if num == target:
        found = True
        print(f"Found {target} at index {i}")
        break
```

```
if not found:
    print("Number not found")
```

11. 11. Implement binary search for strings

```
words = sorted(["apple", "banana", "cherry", "date"])
target = "banana"
low, high = 0, len(words) - 1
while low <= high:
    mid = (low + high) // 2
    if words[mid] == target:
        print("Found at index", mid)
        break
    elif words[mid] < target:
        low = mid + 1
    else:
        high = mid - 1
```

12. 12. Perform all operations of random functions

```
import random
print(random.random())
print(random.randint(1, 10))
print(random.choice(['a', 'b', 'c']))
```

13. 13. Perform all math operations

```
import math
print(math.sqrt(16))
print(math.sin(math.radians(90)))
print(math.factorial(5))
```

14. 14. Create user-defined functions with different arguments

```
def greet(name, message="Hello"):
    print(f"{message}, {name}")
greet("Alice")
greet("Bob", "Welcome")
```

15. 15. Create packages and import modules for real application

```
from my_package.module import function_name
function_name()
```

16. 16. Perform File manipulations

```
with open("file.txt", "w") as f:
    f.write("Hello World")
with open("file.txt", "r") as f:
    print(f.read())
```

17. 17. Handle user-defined exceptions

```
class CustomError(Exception):
    pass
try:
    raise CustomError("An error occurred")
except CustomError as e:
    print(e)
```

18. 18. Handle multiple exceptions

```
try:
    num = int(input("Enter number: "))
    result = 10 / num
except ValueError:
    print("Invalid input")
except ZeroDivisionError:
    print("Cannot divide by zero")
```

19. 19. Create command-line arguments for binary search

```
import sys
target = sys.argv[1]
sorted_list = ["apple", "banana", "cherry"]
```

20. 20. Find substring in a string using command-line arguments

```
import sys

main_string = sys.argv[1]

substring = sys.argv[2]

if substring in main_string:

    print(f"{substring} found in {main_string}")

else:

    print(f"{substring} not found in {main_string}")
```