1. Create a function that calculates the factorial of a given positive integer based on user input.

```
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers."

    elif n == 0 or n == 1:
        return 1

    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

num = int(input("Enter a positive integer: "))

print(f"Factorial of {num} : {factorial(num)}")
```

2. Create a function that takes a number as an argument and returns True if it's prime, False otherwise.

```
def prime(num):
    if num <= 1:
        return False

    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False

    return True

number = int(input("Enter a number: "))

print(f"{number} is prime : {prime(number)}")
```

3. Write a function that takes a list of numbers and returns a new list with only the even numbers.

```
def filter_even_numbers(x):
    return [num for num in x if num % 2 == 0]

numbers = [0,1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print("Even numbers:", filter_even_numbers(numbers))
```

4. Write a function that counts the number of vowels in a string.

```
def count_vowels(x):
    vowels = 'aeiouAEIOU'
    return sum(1 for char in x if char in vowels)

text = input("Enter a string: ")

print(f"Number of vowels: {count_vowels(text)}")
```

5. Write a function student_grades() that takes a dictionary of students and their grades (each grade is a list of scores). It should return a dictionary with each student's name and their final grade (average of all scores, rounded to two decimal places). The function should also classify the students' final grades as "Excellent" (90 and above), "Good" (80–89), "Average" (70–79), and "Poor" (below 70).

```
def student_grades(students_grades):

    final_grades = {}
    for student, grades in students_grades.items():
        average_grade = round(sum(grades) / len(grades), 2)
        if average_grade >= 90:
            classification = "Excellent"

        elif 80 <= average_grade < 90:
            classification = "Good"

        elif 70 <= average_grade < 80:
            classification = "Average"

        else:
            classification = "Poor"

        final_grades[student] = {'final_grade': average_grade, 'classification': classification}

    return final_grades

students = {

    "Alice": [85, 90, 88],

    "Bob": [78, 74, 80],

    "Charlie": [95, 92, 97],

    "David": [60, 65, 68]

}

print(student_grades(students))
```

6. Write two functions:

> **is_prime(n) that checks if a number is prime.**

> **generate_primes(limit) that generates a list of all prime numbers up to limit.**

```
def is_prime(n):
    if n <= 1:
        return False

    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False

    return True

def generate_primes(limit):
    primes = []

    for num in range(2, limit + 1):
        if is_prime(num):
            primes.append(num)

    return primes

print(is_prime(7))

print(generate_primes(20))
```

7. Write a function longest_consecutive(nums) that takes a list of integers and returns the length of the longest consecutive elements sequence.

```
def longest_consecutive(nums):
    if not nums:
        return 0

    nums_set = set(nums)

    longest_streak = 0

    for num in nums_set:
        if num - 1 not in nums_set:  # start of a sequence
            current_num = num
            current_streak = 1
            while current_num + 1 in nums_set:
                current_num += 1
                current_streak += 1

            longest_streak = max(longest_streak, current_streak)
    return longest_streak


print(longest_consecutive([100, 4, 200, 1, 3, 2]))
```

8. Write a function called "find_longest_word" that takes a sentence as a string and returns the longest word in the sentence. The function should ignore punctuation and consider only alphabetic characters.

```
import string

def find_longest_word(sentence):
    words = sentence.translate(str.maketrans('', '', string.punctuation)).split()
    longest_word = max(words, key=len)
    return longest_word

user_sentence = input("Enter a sentence: ")

print("The longest word is:", find_longest_word(user_sentence))
```

9. Write a function called "reverse_string" that takes a string as an argument and returns the reverse of that string. Implement the function using a loop and without using built-in string reversal functions.

```
def reverse_string(x):
    reversed_str = ' '
    for char in x:
        reversed_str = char + reversed_str
    return reversed_str

user_input = input("Enter a string to reverse: ")

print("Reversed string:", reverse_string(user_input))
```

10. Create a function called "find_common_elements" that takes two lists as arguments and returns a new list containing the common elements between the two lists.

```
def find_common_elements(list1, list2):
    return list(set(list1) & set(list2))

print(find_common_elements([1, 2, 3], [2, 3, 4]))
```

11. Develop a function called "capitalize_words" that takes a sentence as a string and returns the sentence with each word capitalized.

```
def capitalize_words(sentence):
    return ' '.join(word.capitalize() for word in sentence.split())

user_input = input("Enter a string : ")

print(capitalize_words(user_input))
```

12. Write a function called "calculate_power" that takes two numbers as arguments, a base and an exponent, and returns the result of raising the base to the exponent.

```
def calculate_power(base, exponent):
  return base ** exponent

print(calculate_power(2, 3))
```

13. write a program using each built-in functions:

> type()

```
def built_in_functions_example():
  num_list = [4, 7, -2, 9, 1]
  value = -5.7
  print("Type:", type(num_list))

built_in_functions_example()
```

> max()

```
def built_in_functions_example():
  num_list = [4, 7, -2, 9, 1]
  value = -5.7
  print("Max value:", max(num_list))

built_in_functions_example()
```

> min()

```
def built_in_functions_example():
  num_list = [4, 7, -2, 9, 1]
  value = -5.7
  print print("Min value:", min(num_list))

built_in_functions_example()
```

> abs()

```
def built_in_functions_example():
  num_list = [4, 7, -2, 9, 1]
  value = -5.7
  print("Absolute value:", abs(value))

built_in_functions_example()
```

> round()

```
def built_in_functions_example():
    num_list = [4, 7, -2, 9, 1]
    value = -5.7
    print("Rounded value:", round(value))

built_in_functions_example()
```

> sorted()

```
def built_in_functions_example():
    num_list = [4, 7, -2, 9, 1]
    value = -5.7
    print("Sorted list:", sorted(num_list))

built_in_functions_example()
```

14. Write a Python function called square_numbers that takes a list of numbers as input and returns a new list containing the square of each number.

```
def square_numbers(numbers):
    return [num ** 2 for num in numbers]

print(square_numbers([1, 2, 3, 4]))
```

15. Write a Python function called format_info that takes the following information as keyword arguments: name, age, city, country. The function should format the information into a string and return it.

```
def format_info(**x):
    return f"{x.get('name', '')}, aged {x.get('age', '')}, lives in {x.get('city', '')}, {x.get('country', '')}."

print(format_info(name="Alice", age=25, city="New York", country="USA"))
```