n-inputs-and-outputs-and-functions

November 4, 2023

0.1 Python input and output examples:

1. Input: Accepting user's name and greeting them.

```
name = input("Enter your name: ")
print(f"Hello, {name}!")
```

2. Input: Taking two numbers and adding them.

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
sum_result = num1 + num2
print(f"The sum is: {sum_result}")
```

3. Input: Getting the user's age and determining if they are old enough to vote.

```
age = int(input("Enter your age: "))
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not old enough to vote.")
```

4. Input: Taking a string and counting its length.

```
text = input("Enter a string: ")
length = len(text)
print(f"The string has {length} characters.")
```

5. Input: Accepting a number and checking if it's even or odd.

```
number = int(input("Enter a number: "))
if number % 2 == 0:
    print("The number is even.")
else:
    print("The number is odd.")
```

6. Input: Taking a temperature in Celsius and converting it to Fahrenheit.

```
celsius = float(input("Enter temperature in Celsius: "))
fahrenheit = (celsius * 9/5) + 32
print(f"The temperature in Fahrenheit is: {fahrenheit}")
```

7. Input: Receiving a list of numbers and finding their sum.

```python
numbers = input("Enter a list of numbers separated by spaces: ").split()
numbers = [float(num) for num in numbers]
total = sum(numbers)
print(f"The sum of the numbers is: {total}")
```

8. Input: Taking a radius and calculating the area of a circle.

```python
radius = float(input("Enter the radius of the circle: "))
area = 3.14159 * radius**2
print(f"The area of the circle is: {area:.2f}")
```

9. Input: Accepting a sentence and counting the number of words.

```python
sentence = input("Enter a sentence: ")
words = sentence.split()
word_count = len(words)
print(f"The sentence has {word_count} words.")
```

10. Input: Taking a list of names and sorting them alphabetically.

```python
names = input("Enter a list of names separated by commas: ").split(',')
sorted_names = sorted(names)
print(f"Sorted names: {', '.join(sorted_names)}")
```

11. Input: Accepting a list of integers and finding the maximum and minimum values.

```python
numbers = input("Enter a list of integers separated by spaces: ").split()
numbers = [int(num) for num in numbers]
max_value = max(numbers)
min_value = min(numbers)
print(f"Maximum value: {max_value}, Minimum value: {min_value}")
```

12. Input: Taking a sentence and counting the occurrences of a specific word.

```python
sentence = input("Enter a sentence: ")
target_word = input("Enter a word to count: ")
word_count = sentence.lower().count(target_word.lower())
print(f"The word '{target_word}' appears {word_count} times in the sentence.")
```

13. Input: Accepting a number and checking if it's prime.

```python
num = int(input("Enter a number: "))

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

if is_prime(num):
    print(f"{num} is a prime number.")
```

```
else:
    print(f"{num} is not a prime number.")
```

14. Input: Taking a string and reversing it.

```
text = input("Enter a string: ")
reversed_text = text[::-1]
print(f"Reversed string: {reversed_text}")
```

15. Input: Accepting a list of grades and calculating the average.

```
grades = input("Enter a list of grades separated by spaces: ").split()
grades = [float(grade) for grade in grades]
average = sum(grades) / len(grades)
print(f"The average grade is: {average:.2f}")
```

16. Input: Taking a number and generating its multiplication table.

```
num = int(input("Enter a number for its multiplication table: "))
for i in range(1, 11):
    product = num * i
    print(f"{num} x {i} = {product}")
```

0.2 Python Function examples:

1. **Hello World Function:**

```
def say_hello():
    print("Hello, World!")

say_hello()
```

2. **Function with Parameters:**

```
def greet(name):
    print(f"Hello, {name}!")

greet("Alice")
```

3. **Function with Default Parameter:**

```
def greet(name="Guest"):
    print(f"Hello, {name}!")

greet()
```

4. **Function with Return Value:**

```
def add(a, b):
    return a + b

result = add(3, 5)
print(result)
```

5. **Function with Multiple Return Values (Tuple):**

```python
def get_name_and_age():
    return "Alice", 30

name, age = get_name_and_age()
print(f"Name: {name}, Age: {age}")
```

6. **Recursive Function (Factorial):**

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

result = factorial(5)
print(result)
```

7. **Lambda Function:**

```python
double = lambda x: x * 2
print(double(5))
```

8. **Function with *args (Arbitrary Number of Arguments):**

```python
def sum_numbers(*args):
    total = 0
    for num in args:
        total += num
    return total

result = sum_numbers(1, 2, 3, 4, 5)
print(result)
```

9. **Function with** kwargs (Arbitrary Keyword Arguments):**

```python
def print_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_info(name="Alice", age=30, city="New York")
```

10. **Higher-Order Function (Function as an Argument):**

```python
def apply(func, x):
    return func(x)

def square(x):
    return x * x

result = apply(square, 5)
print(result)
```

11. **Function with a Docstring:**

```python
def greet(name):
    """
    This function greets the person passed in as a parameter.
    """
    print(f"Hello, {name}!")

greet("Bob")
```

12. **Function with a Global Variable:**

```python
global_variable = 10

def modify_global():
    global global_variable
    global_variable = 20

print(global_variable)
modify_global()
print(global_variable)
```

13. **Anonymous Function (Lambda) with Filter:**

```python
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)
```

14. **Anonymous Function (Lambda) with Map:**

```python
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x ** 2, numbers))
print(squared_numbers)
```

15. **Function as an Object:**

```python
def square(x):
    return x * x

def cube(x):
    return x * x * x

# Using functions as objects
functions = [square, cube]

for function in functions:
    print(function(5))
```

16. **Function with Decorator:**

```python
def my_decorator(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

@my_decorator
def say_hello():
    print("Hello!")

say_hello()
```

17. **Function with Recursion (Fibonacci Sequence):**

```python
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

for i in range(10):
    print(fibonacci(i))
```

18. **Function with a Generator (Yield):**

```python
def generate_fibonacci_sequence(n):
    a, b = 0, 1
    for _ in range(n):
        yield a
        a, b = b, a + b

for number in generate_fibonacci_sequence(10):
    print(number)
```

19. **Function with Default Argument Value and Non-default Argument:**

```python
def greet(name, greeting="Hello"):
    print(f"{greeting}, {name}!")

greet("Alice")  # Uses the default greeting
greet("Bob", "Hi")  # Uses a custom greeting
```

20. **Function with a Try-Except Block:**

```python
def safe_divide(x, y):
    try:
        result = x / y
        return result
    except ZeroDivisionError:
        return "Division by zero is not allowed."

print(safe_divide(10, 2))
print(safe_divide(5, 0))
```

21. **Function with a List as a Parameter:**

```python
def print_list_elements(numbers):
    for number in numbers:
        print(number)

num_list = [1, 2, 3, 4, 5]
print_list_elements(num_list)
```

22. **Function to Find the Maximum Element in a List:**

```python
def find_max(numbers):
    if len(numbers) == 0:
        return None
    max_num = numbers[0]
    for number in numbers:
        if number > max_num:
            max_num = number
    return max_num

num_list = [10, 5, 20, 8, 15]
max_value = find_max(num_list)
print(f"The maximum value is: {max_value}")
```

23. **Function to Check for Palindrome:**

```python
def is_palindrome(word):
    return word == word[::-1]

print(is_palindrome("racecar"))
print(is_palindrome("python"))
```

24. **Function to Calculate the Area of a Circle:**

```python
import math

def circle_area(radius):
    return math.pi * radius ** 2

print(circle_area(5))
```

25. **Function to Filter Even Numbers from a List using List Comprehension:**

```python
def filter_even(numbers):
    return [num for num in numbers if num % 2 == 0]

num_list = [1, 2, 3, 4, 5, 6]
even_numbers = filter_even(num_list)
print(even_numbers)
```

26. **Function to Count the Occurrences of a Character in a String:**

```python
def count_char_occurrences(string, char):
    count = 0
    for c in string:
        if c == char:
            count += 1
    return count

text = "Hello, world!"
char_to_count = "o"
count = count_char_occurrences(text, char_to_count)
print(f"The character '{char_to_count}' appears {count} times.")
```

27. **Function to Calculate Factorial with Memoization:**

```
# Using memoization to optimize factorial calculation
memo = {}

def factorial(n):
    if n in memo:
        return memo[n]
    if n == 0:
        return 1
    else:
        result = n * factorial(n - 1)
        memo[n] = result
        return result

result = factorial(10)
print(result)
```

28. **Function with Multiple Decorators:**

```
def uppercase_decorator(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return result.upper()
    return wrapper

def greeting(name):
    return f"Hello, {name}!"

@uppercase_decorator
def polite_greeting(name):
    return greeting(name)

print(polite_greeting("Alice"))
```

29. **Function with a List of Dictionaries:**

```
def find_person_by_name(people, name):
    for person in people:
        if person['name'] == name:
            return person
    return None
```

```python
people = [
    {'name': 'Alice', 'age': 30},
    {'name': 'Bob', 'age': 25},
    {'name': 'Charlie', 'age': 35}
]

result = find_person_by_name(people, 'Bob')
if result:
    print(f"Found: {result['name']}, Age: {result['age']}")
else:
    print("Person not found.")
```

30. **Function to Generate a List of Primes:**

```python
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

def generate_primes(limit):
    primes = [2, 3, 5]
    for num in range(7, limit + 1, 2):
        if is_prime(num):
            primes.append(num)
    return primes

prime_list = generate_primes(50)
print(prime_list)
```