My Python Course Notes

Structured Revision for Every Lesson

Siddharth Patel
HTW Berlin

June 12, 2025

Contents

1 If Else – Decision Making in Python

Q1: Greeting by First Name

Write a program that asks the user for their first name. If the user's name is `Marie` or `Otto`, the program should print `Hello Marie!` or `Hello Otto!` respectively. Otherwise, nothing should be printed. Leading and trailing spaces should be ignored. The input and output must match the following examples exactly.

Example 1:
`First name: Marie`
`Hello Marie!`

Example 2:
`First name: Karl`
(No output expected)

Q1 – Solution

```
name = input("First name: ").strip()

if name == "Marie":
    print("Hello Marie!")
elif name == "Otto":
    print("Hello Otto!")
```

Q2: Compare Two Floating-Point Numbers

Write a program that asks the user to input two floating-point numbers, `a` and `b`. Use an `if-else` statement to determine and print the larger number.

Example:
`a = 10`
`b = 11.2`
`The larger number is: 11.2`

Q2 – Solution

```
a = float(input("a = "))
b = float(input("b = "))

if a > b:
    print("The larger number is:", a)
else:
    print("The larger number is:", b)
```

Q3: Compare with a Single if Statement

Solve the previous task again, but this time using only a single `if` statement without `else`. The input and output must match the example exactly.

Example:
a = 10
b = 11.2
The larger number is: 11.2

Q3 – Solution

```
a = float(input("a = "))
b = float(input("b = "))

if b > a:
    print("The larger number is:", b)
if a > b:
    print("The larger number is:", a)
```

Q4: Largest of Three Numbers

Write a program that asks the user to input three floating-point numbers: a, b, and c. Then determine and print the largest number.

Example:
a = 10
b = 11.2
c = -12.31
The largest number is: 11.2

Q4 – Solution

```
a = float(input("a = "))
b = float(input("b = "))
c = float(input("c = "))

if a >= b and a >= c:
    print("The largest number is:", a)
elif b >= a and b >= c:
    print("The largest number is:", b)
else:
    print("The largest number is:", c)
```

Q5: Age-Based Classification (if-elif-else)

Write a program that asks the user for their age and prints a message based on the following rules:

- Under 18: You are a minor.
- Between 18 and 65: You are an adult.
- 66 and older: You are of retirement age.

Example 1:
Please enter your age: 23
You are an adult.

Example 2:
```
Please enter your age: 13
You are a minor.
```

Q5 – Solution

```
age = int(input("Please enter your age: "))

if age < 18:
    print("You are a minor.")
elif age <= 65:
    print("You are an adult.")
else:
    print("You are of retirement age.")
```

Q6: Age-Based Classification with Extended Range

Write a program similar to Q5 but with the following additional condition:

- Under 18: `You are a minor.`
- Between 18 and 65: `You are an adult.`
- Between 66 and 99: `You are of retirement age.`
- 100 or older: `You are in the Methuselah age group.`

Example:
```
Please enter your age: 101
You are in the Methuselah age group.
```

Q6 – Solution

```
age = int(input("Please enter your age: "))

if age < 18:
    print("You are a minor.")
elif age <= 65:
    print("You are an adult.")
elif age <= 99:
    print("You are of retirement age.")
else:
    print("You are in the Methuselah age group.")
```

Q7: Min and Max from Input Without Using min/max

Write a program that asks the user to enter a list of integers separated by commas. Then determine and print the smallest and largest number without using the built-in `min()` or `max()` functions.

Example:
```
Enter numbers: 1, 3, 7, -4, 9, 0, 2
Smallest number: -4
Largest number: 9
```

Q7 – Solution

```
raw_input = input("Enter numbers: ")
numbers = [int(x.strip()) for x in raw_input.split(",")]

smallest = numbers[0]
largest = numbers[0]

for num in numbers[1:]:
    if num < smallest:
        smallest = num
    if num > largest:
        largest = num

print("Smallest number:", smallest)
print("Largest number:", largest)
```

2 Match Statement – Pattern Matching in Python

Q1: Weekday Abbreviation to Full Name

Write a program that asks the user to input a weekday abbreviation (2 letters), such as `Mo`, `Di`, `Mi`, etc. The program should return the full weekday name. If the input is invalid, print `Error`. Use a `match` statement to solve this.

Example 1:
`Input: Mo`
`Monday`

Example 2:
`Input: Mon`
`Error`

Q1 – Solution

```
day = input("Input: ").strip()

match day:
    case "Mo":
        print("Monday")
    case "Di":
        print("Tuesday")
    case "Mi":
        print("Wednesday")
    case "Do":
        print("Thursday")
    case "Fr":
        print("Friday")
    case "Sa":
        print("Saturday")
    case "So":
        print("Sunday")
    case _:
        print("Error")
```

Q2: Month Abbreviation to Season

Write a program that asks the user to enter a three-letter month abbreviation (e.g., `Jan`, `Feb`). The input should be case-insensitive. The program should return the corresponding season:

- Spring: Mar–May
- Summer: Jun–Aug
- Autumn: Sep–Nov
- Winter: Dec–Feb

Invalid input should return `Error`. Use a `match` statement with as few `case` blocks as possible (5 total).

Example:
`Input: Feb`
`Winter`

Q2 – Solution

```
month = input("Input: ").strip().lower()

match month:
    case "mar" | "apr" | "may":
        print("Spring")
    case "jun" | "jul" | "aug":
        print("Summer")
    case "sep" | "oct" | "nov":
        print("Autumn")
    case "dec" | "jan" | "feb":
        print("Winter")
    case _:
        print("Error")
```

Q3: Rewrite if-elif-else Using match

Given the following code:

```
        val = int(input('Value: '))
        if val >= 0 and val < 4:
        print('Small number')
        elif val < 6:
        print('Medium number')
        elif val == 7 or val == 9:
        print('Special number')
        else:
        print('Some other number')
```

Rewrite the program using a `match` statement instead of `if-elif-else` to achieve the same logic and output.

Q3 – Solution

```python
val = int(input("Value: "))

match val:
    case 0 | 1 | 2 | 3:
        print("Small number")
    case 4 | 5:
        print("Medium number")
    case 7 | 9:
        print("Special number")
    case _:
        print("Some other number")
```

3 The for Loop – Counting, Iteration, and Containers

Q1: Count from 1 to a User-Defined Value

Write a program that asks the user to input a number. Then use a `for` loop to print all numbers from 1 to the entered value.

Example:

```
Last count value: 4
1
2
3
4
```

Q1 – Solution

```python
n = int(input("Last count value: "))
for i in range(1, n + 1):
    print(i)
```

Q2: Count Down to 1

Repeat the previous task, but count down from the input value to 1, using a step of -1.

Example:

```
Last count value: 4
4
3
2
1
```

Q2 – Solution

```
n = int(input("Last count value: "))
for i in range(n, 0, -1):
    print(i)
```

Q3: Factorial with a for Loop

Write a program that calculates the factorial of a user-given number n. Use a for loop starting at 1.

Example:
Please enter n: 4
n! = 24

Q3 – Solution

```
n = int(input("Please enter n: "))
factorial = 1
for i in range(1, n + 1):
    factorial *= i
print("n! =", factorial)
```

Q4: Count Occurrences of a Character in a String

Write a program that asks the user for a string and counts how many times the letter e appears using a for loop.

Example:
Input: Das Leben ist schoen!
Number of e: 3

Q4 – Solution

```
text = input("Input: ")
count = 0
for char in text:
    if char == 'e':
        count += 1
print("Number of e:", count)
```

Q5: Sum All Elements in a List

Given the list:

`intlist = [1, 4, 3, 7, -5, 3, 5]`

Write a program that uses a for loop to sum all elements and print the result.

Output:
Sum: 18

Q5 – Solution

```
intlist = [1, 4, 3, 7, -5, 3, 5]
total = 0
for number in intlist:
    total += number
print("Sum:", total)
```

4 Functions – Parameters, Return Values, and Reusability

Q1: Average of Three Numbers

Write a function that accepts three floating-point numbers as parameters and returns their average. Add a main program that asks the user to enter three values and then calls the function and displays the result.

Example:
a = 12
b = 18
c = 6
Average = 12

Q1 – Solution

```
def average(a, b, c):
    return (a + b + c) / 3

a = float(input("a = "))
b = float(input("b = "))
c = float(input("c = "))

result = average(a, b, c)
print("Average =", int(result))
```

Q2: Factorial Function

Write a function that receives an integer n as a parameter and returns n!. Add a main program that gets input from the user and prints the result.

Example:
n = 4
n! = 24

Q2 – Solution

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

```
6
7 n = int(input("n = "))
8 print("n! =", factorial(n))
```

Q3: Area and Perimeter of Rectangle

Write a function `rect(a, b)` that receives two sides of a rectangle and returns the area and perimeter. Add a main program that takes inputs and prints both results.

Example:
`a = 4`
`b = 3`
`Area = 12`
`Perimeter = 14`

Q3 – Solution

```
1  def rect(a, b):
2      area = a * b
3      perimeter = 2 * (a + b)
4      return area, perimeter
5
6  a = int(input("a = "))
7  b = int(input("b = "))
8
9  area, perimeter = rect(a, b)
10 print("Area =", area)
11 print("Perimeter =", perimeter)
```

Q4: Count Vowels Using Function

Write a function that accepts a string `s` and a letter `letter`, and returns how often that letter appears (case-insensitive). In the main program, prompt for a string and print the number of occurrences of `a, e, i, o, u`.

Q4 – Solution

```
1  def count_letter(s, letter):
2      s = s.lower()
3      letter = letter.lower()
4      count = 0
5      for char in s:
6          if char == letter:
7              count += 1
8      return count
9
10 s = input("Input: ")
11 for vowel in "aeiou":
12     print(f"{vowel}: {count_letter(s, vowel)}")
```

Example:
Input: Die Summe der Kathetenquadrate ist Gleich dem Hypotenusenquadrat
a: 5
e: 10
i: 3
o: 1
u: 4

Q5: Custom `join()` Implementation

Write a function `myJoin(lst, sep)` that replicates the behavior of `join()` without actually using it. It should combine list elements into a single string, separated by `sep`.

Test Code:

```
    mylist = ['Wir', 'müssen', 'uns', 'Sisyphos', 'als', 'einen', 'glücklichen', 'Menschen',
    ↪  'vorstellen']
    mystring = myJoin(mylist, '-')
    print(mystring)
```

Expected Output:
`Wir-müssen-uns-Sisyphos-als-einen-glücklichen-Menschen-vorstellen`

Q5 – Solution

```
def myJoin(lst, sep):
    result = ""
    for i in range(len(lst)):
        result += lst[i]
        if i != len(lst) - 1:
            result += sep
    return result

mylist = ['Wir', 'müssen', 'uns', 'Sisyphos', 'als', 'einen', 'glücklichen', 'Menschen',
↪  'vorstellen']
mystring = myJoin(mylist, '-')
print(mystring)
```

Q6: Recursive Sum Function

Write a recursive function `total(n)` that returns the sum from 1 to `n` without using loops. Include a main program to prompt the user and show the result.

Example:
n = 4
Sum = 10

Q6 – Solution

```
def total(n):
    if n <= 1:
```

```
        return n
    return n + total(n - 1)

n = int(input("n = "))
print("Sum =", total(n))
```

5 The while Loop – Manual Iteration and Termination Conditions

Q1: Rewrite a for-loop as a while-loop (0 to 4)

Given the following for-loop:

```
for i in range(5):
print(i)
```

Rewrite it using a while-loop to produce the same output:

Expected Output:

```
0
1
2
3
4
```

Q1 – Solution

```
i = 0
while i < 5:
    print(i)
    i += 1
```

Q2: Rewrite a for-loop with start, stop, step

Given the following for-loop:

```
for i in range(5, 13, 2):
print(i)
```

Rewrite it using a while-loop to produce the same output:

Expected Output:

```
5
7
9
11
```

Q2 – Solution

```
i = 5
while i < 13:
    print(i)
    i += 2
```

Q3: Capitalize User Input Until "Ende"

Write a program that reads user input, converts it to uppercase, and prints it. The loop should end when the user enters ende (case-insensitive). Use a `while`-loop.

Example:

```
Input: hallo welt
Output: HALLO WELT
Input: Noch ein Test
Output: NOCH EIN TEST
Input: Ende
```

Q3 – Solution

```
while True:
    text = input("Input: ").strip()
    if text.lower() == "ende":
        break
    print("Output:", text.upper())
```

Q4: Sum Until Empty Input

Write a program that asks the user to input integers, one at a time, inside a `while`-loop. When the user enters an empty line, stop and display the sum. Ignore extra spaces.

Example:

```
Input: 1
Input:  55
Input: 7
Input:
Sum: 63
```

Q4 – Solution

```
total = 0

while True:
    line = input("Input: ").strip()
    if line == "":
        break
    total += int(line)
```

```
8
9 print("Sum:", total)
```

Q5: Sum from Multi-Value Inputs with Nested Loops

Write a program that allows the user to input multiple integers in a single line (space-separated). Continue until an empty line is entered, then print the total sum. Use only `while`-loops.

Example:

```
Input: 1 3 5
Input:  7 8
Input: 4
Input:
Sum: 28
```

Q5 – Solution

```
1  total = 0
2
3  while True:
4      line = input("Input: ").strip()
5      if line == "":
6          break
7      for num in line.split():
8          total += int(num)
9
10 print("Sum:", total)
```