TECHNIK NEST

INNOVATIVE MINDS, NESTING SUCCESS

Name: Syed Ali Mosavi

Intern ID: TN/IN02/PY/028

Email ID: alimonster105@gmail.com

Task week: 01

Internship Domain: Python Development

Instructor Name: Mr. Hassan Ali

Tasks – Intro & Install

Task 01:

Install Python & print version.

Step-by-Step Instructions

I downloaded Python from https://www.python.org/downloads/.

1. I ran the installer, checked **"Add Python to PATH"**, and clicked **Install Now**.
2. I opened PyCharm and created a new file named version.py in my project.
3. I used **import sys** to access system functions and **sys.version** to retrieve the current Python version.

Code Snippet:

```
Task.py        task version.py
import sys  # imports system functions
print("Python version:", sys.version)  # prints the version
```

Output Snippet:

```
Run    task version
C:\Users\Ali\Desktop\Internship\.venv\Scripts\python.exe "C:\Users\Ali\Desktop\Internship\task version.py"
Python version: 3.13.4 (tags/v3.13.4:8a526ec, Jun  3 2025, 17:46:04) [MSC v.1943 64 bit (AMD64)]

Process finished with exit code 0
```

Learning and Challenges

- I learned how to install Python so the python command works everywhere.

- I discovered that import sys gives access to system functions and sys.version shows the exact Python version.

Task 02:

Run hello script printing your name.

Step-by-Step Instructions

1. I opened PyCharm and created a new file named hello.py in my project.
2. I used print ("Hello, Ali Mosavi!") to display my name.

Code Snippet:

```
Task.py    task version.py    hello task.py
1    print("Hello, Ali Mosavi!")
```

Output Snippet:

```
Run    hello task
C:\Users\Ali\Desktop\Internship\.venv\Scripts\python.exe "C:\Users\Ali\Desktop\Internship\hello task.py"
Hello, Ali Mosavi!

Process finished with exit code 0
```

Learning and Challenges

- I learned how a simple print() statement shows text in the console.

Tasks – Syntax & Indentation

Task 03:

Fix badly-indented code.

Step-by-Step Instructions

1. Strip all leading whitespace from every line so nothing is indented.
2. Find each block header (lines ending with :), such as def greet(): or for i in range(3):.
3. Indent the body of each block by exactly 4 spaces under its header.
4. Add the missing call to greet() so the function actually runs.
5. Use spaces only, never tabs—tabs vary by editor and can trigger errors.

Code Snippet (Wrong):

```
def greet():
print("Hello, world!")
  print("Welcome to Python!")

for i in range(3):
 print(i)
```

Code Snippet (Fixed):

```
def greet():  1 usage
    print("Hello, world!")
    print("Welcome to Python!")

greet()  # call the function for prints the greetings

for i in range(3):
    print(i)
```

Output Snippet:

```
Run    badly indented
C:\Users\Ali\Desktop\Internship\.venv\Scripts\python.exe "C:\Users\Ali\Desktop\Internship\badly indented.py"
Hello, world!
Welcome to Python!
0
1
2

Process finished with exit code 0
```

Learning and Challenges

- Function calls: Defining a function (def) doesn't run it—you must call it (greet()).
- Why 4 spaces? It follows Python's official style guide (PEP 8) for clear, consistent indentation.
- Why not tabs? Tabs can display inconsistently across editors; spaces avoid that problem.

<u>Task 04:</u>

Add comments explaining each step.

Step-by-Step Instructions

1. Identify each logical part of the code (function definition, function call, loop).
2. Write a brief comment above the function to explain its purpose.
3. Add inline comments next to each print to describe what is displayed.
4. Comment the loop header to clarify what it does.

Code Snippet:

```python
# Define a function to print two greetings
def greet():  1 usage
    print("Hello, world!")       # print the first greeting
    print("Welcome to Python!")  # print the second greeting

greet()  # call the function for greetings appear

# Loop from 0 to 2 and print each number
for i in range(3):
    print(i)  # print current loop index
```

Output Snippet:

```
C:\Users\Ali\Desktop\Internship\.venv\Scripts\python.exe C:\Users\Ali\Desktop\Internship\commenting.py
Hello, world!
Welcome to Python!
0
1
2

Process finished with exit code 0
```

Learning and Challenges

- Adding comments makes it clear what each block and line does when someone reads the code.
- Writing concise comments helps reinforce your own understanding of each step.

Tasks – Variables & Types

Task 05:

Collect user profile & print typed summary.

Step-by-Step Instructions

1. I created a new file named profile.py in PyCharm.
2. I used input() to ask for first name, last name, age, city, and profession.
3. I combined the first and last names into a single full_name variable.
4. I printed a formatted summary using f-strings to display all the collected details.

Code Snippet:

profile.py × hello task.py badly indented.py

```python
print("----- USER REGISTRATION -----")
first_name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
age = input("Enter your age: ")
city = input("Enter your city: ")
profession = input("Enter your profession: ")

full_name = first_name + " " + last_name

print("\n----- REGISTRATION SUMMARY -----")
print(f"Welcome {full_name}!")
print(f"Age: {age}")
print(f"City: {city}")
print(f"Profession: {profession}")
print("Registration completed successfully.")
```

Output Snippet:

profile.py × hello task.py badly indented.py

```python
print("----- USER REGISTRATION -----")
first_name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
age = input("Enter your age: ")
city = input("Enter your city: ")
profession = input("Enter your profession: ")

full_name = first_name + " " + last_name

print("\n----- REGISTRATION SUMMARY -----")
print(f"Welcome {full_name}!")
print(f"Age: {age}")
print(f"City: {city}")
print(f"Profession: {profession}")
print("Registration completed successfully.")
```

Learning and Challenges

- I learned how to gather multiple inputs from the user using input().
- I practiced combining strings and formatting output neatly with f-strings.

Task 06:

Swap two variables without temp var.

Step-by-Step Instructions

1. I read two values into variables a and b.
2. I printed their values before swapping.
3. I used tuple unpacking (a, b = b, a) to swap them without a temporary variable.
4. I printed their values after swapping.

Code Snippet:

```
task version.py    hello task.py    badly indented.py    commenting.py    swaping var without temp.py

a = input("Enter value for a: ")
b = input("Enter value for b: ")
print("Before swap:", a, b)
a, b = b, a
print("After swap:", a, b)
```

Output Snippet:

```
Run    swaping var without temp

C:\Users\Ali\Desktop\Internship\.venv\Scripts\python.exe "C:\Users\Ali\Desktop\Internship\swaping var without temp.py"
Enter value for a: 1
Enter value for b: 2
Before swap: 1 2
After swap: 2 1

Process finished with exit code 0
```

Learning and Challenges

- I learned that Python's tuple unpacking lets me swap values in one line.
- This avoids the need for a temporary variable and makes the code cleaner.

Tasks – Casting & I/O

Task 07:

Read three numbers: output avg.

Step-by-Step Instructions

- I created a new file named three num average.py.
- I used input () three times to read numbers and converted them to floats.
- I calculated the average by summing the three values and dividing by 3.
- I printed the result formatted to two decimal places.

Code Snippet:

three num avg.py × profile.py hello task.py badly indented.py

```python
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))
avg = (num1 + num2 + num3) / 3
print(f"Average: {avg:.2f}")
```

Output Snippet:

Run three num avg ×

```
Enter first number: 1
Enter second number: 2
Enter third number: 3
Average: 2.00

Process finished with exit code 0
```

Learning and Challenges

- I learned how to convert inputs to floats and perform arithmetic operations.
- I practiced formatting numeric output using f-strings for better readability.

Task 08:

Convert minutes to hours + minutes.

Step-by-Step Instructions

1. I created a new file named hour to min .py.
2. I used input() to read the total number of minutes and converted it to an integer.
3. I calculated the hours by dividing by 60 (using //) and the remaining minutes with modulus (%).
4. I printed the result in the form "X hour(s) and Y minute(s)."

Code Snippet:

three num avg.py | hour to min.py | profile.py

```python
total_minutes = int(input("Enter total minutes: "))
hours = total_minutes // 60
minutes = total_minutes % 60
print(f"{hours} hour(s) and {minutes} minute(s)")
```

Output Snippet:

Run | hour to min

```
C:\Users\Ali\Desktop\Internship\.venv\
Enter total minutes: 155
2 hour(s) and 35 minute(s)

Process finished with exit code 0
```

Learning and Challenges

- I learned to use **integer division** (`//`) to get whole hours and **modulus** (`%`) to find leftover minutes.

- I practiced converting user input to integers and formatting the result clearly.

Tasks – Operators

Task 09:

BMI calc from user input.

Step-by-Step Instructions

1. I created a new file named bmi calc.py.
2. I used input () to read weight in kilograms and height in meters, converting both to floats.
3. I calculated BMI by dividing the weight by the square of the height.
4. I printed the BMI rounded to two decimal places.

Code Snippet:

three num avg.py | hour to min.py | BMI calc.py

```python
weight = float(input("Enter weight (kg): "))
height = float(input("Enter height (m): "))
bmi = weight / (height ** 2)
print(f"Your BMI: {bmi:.2f}")
```

Output Snippet:

```
Run    BMI calc
C:\Users\Ali\Desktop\Internship\.venv\Scripts\python.exe "C:\Users\Ali\Desktop\Internship\BMI calc.py"
Enter weight (kg): 150
Enter height (m): 1.83
Your BMI: 44.79

Process finished with exit code 0
```

Learning and Challenges

- I learned to convert input strings into floats for mathematical calculations.
- I practiced using the exponent operator (**) and formatting numeric output with f-strings.

Task 10:

Simple interest calc..

Step-by-Step Instructions

1. I created a new file named interest calc.py.
2. I used input() to read the principal amount (P), rate of interest (R), and time in years (T), converting each to a float.
3. I calculated simple interest using the formula SI = (P * R * T) / 100.
4. I printed the result formatted to two decimal places.

Code Snippet:

```
profile.py    hello task.py    interest calc.py    b
p = float(input("Enter principal amount: "))
r = float(input("Enter rate of interest (%): "))
t = float(input("Enter time in years: "))
si = (p * r * t) / 100
print(f"Simple Interest: {si:.2f}")
```

Output Snippet:

```
Run    interest calc
C:\Users\Ali\Desktop\Internship\.venv\Scripts\python.exe "C:\Users\Ali\Desktop\Internship\interest calc.py"
Enter principal amount: 100000
Enter rate of interest (%): 5
Enter time in years: 2.5
Simple Interest: 12500.00

Process finished with exit code 0
```

Learning and Challenges

- I learned how to apply a financial formula in code by converting inputs to floats and performing arithmetic.
- I practiced formatting numeric output with f-strings for clear, two-decimal display.

Tasks – Strings

Task 11:

Username builder from full name.

Step-by-Step Instructions

1. I created a file named username builder.py.
2. I read the user's full name with input().
3. I removed all spaces and converted the name to lowercase.
4. I imported random and generated a two-digit number.
5. I concatenated the processed name and number into the final username and printed it.

Code Snippet:

```python
import random

full_name = input("Enter your full name: ")
clean_name = full_name.replace(" ", "").lower()
number = random.randint(10, 99)
username = clean_name + str(number)
print(f"Generated username: {username}")
```

Output Snippet:

```
C:\Users\Ali\Desktop\Internship\.venv\
Enter your full name: syed ali
Generated username: syedali23

Process finished with exit code 0
```

Learning and Challenges

- I learned to strip spaces with replace() and convert strings to lowercase.
- I practiced combining string manipulation with random.randint() to ensure unique usernames.

Task 12:

Vowel/consonant counter.

Step-by-Step Instructions

- I created a file named vowel consonant counter.py.
- I used input() to read a text string from the user.
- I defined a set of vowels and iterated over each character to count vowels.

- I counted consonants by checking alphabetic characters that weren't in the vowel set.
- I printed both counts.

Code Snippet:

```
interest calc.py    Vowel consonant counter.py    username builder.py    badly indented.py
text = input("Enter a text: ")
vowels = set("aeiouAEIOU")
vowel_count = sum(1 for ch in text if ch in vowels)
consonant_count = sum(1 for ch in text if ch.isalpha() and ch not in vowels)
print(f"Vowels: {vowel_count}, Consonants: {consonant_count}")
```

Output Snippet:

```
Run    Vowel consonant counter
C:\Users\Ali\Desktop\Internship\.venv
Enter a text: syed ali mosavi
Vowels: 6, Consonants: 7

Process finished with exit code 0
```

Learning and Challenges

- I learned how to use generator expressions to count specific characters efficiently.
- I practiced distinguishing vowels from consonants and handling non-letter characters correctly.

Tasks – Conditionals

Task 13:

Grade calculator.

Step-by-Step Instructions

- I created a file named grade calc.py.
- I read the user's score with input() and converted it to a float.
- I used if, elif, and else to assign a grade (A, B, C, or F) based on the score.
- I printed the resulting grade.

Code Snippet:

interest calc.py | Vowel consonant counter.py | Grade calc.py

```python
score = float(input("Enter score (0-100): "))
if score >= 90:
    grade = "A"
elif score >= 75:
    grade = "B"
elif score >= 50:
    grade = "C"
else:
    grade = "F"
print(f"Grade: {grade}")
```

Output Snippet:

Run | Grade calc

```
C:\Users\Ali\Desktop\Internship\.venv\
Enter score (0-100): 68
Grade: C

Process finished with exit code 0
```

Learning and Challenges

- I learned how to use conditional branching to handle multiple ranges.
- I practiced converting user input to a number and mapping it to a category.

Task 14:

Password strength classifier.

Step-by-Step Instructions

1. I created a file named password strength.py.
2. I read the password input from the user.
3. I checked for length ≥ 8, lowercase letters, uppercase letters, digits, and special characters.
4. I counted how many of these criteria were met.
5. I classified strength as “Weak,” “Medium,” or “Strong” based on the count.

Code Snippet:

interest calc.py | Vowel consonant counter.py | password strg.py | Grad

```python
password = input("Enter a password: ")

length_ok = len(password) >= 8
has_lower = any(c.islower() for c in password)
has_upper = any(c.isupper() for c in password)
has_digit = any(c.isdigit() for c in password)
has_special = any(not c.isalnum() for c in password)

score = sum([length_ok, has_lower, has_upper, has_digit, has_special])

if score <= 2:
    strength = "Weak"
elif score == 3 or score == 4:
    strength = "Medium"
else:
    strength = "Strong"

print(f"Password strength: {strength}")
```

Output Snippet:

```
Run    password strg
C:\Users\Ali\Desktop\Internship\.venv\
Enter a password: alimosavi105
Password strength: Medium

Process finished with exit code 0
```

Learning and Challenges

- I learned how to use random.randint() to append a simple numeric suffix.
- I combined string processing with randomness to produce more unique usernames.

Tasks – Loops

Task 15:

Multiplication table.

Step-by-Step Instructions

1. I created a new file named multiplication table.py.
2. I used input () to read an integer n from the user.
3. I set up a for loop from 1 to 10 to multiply n by each loop index.
4. I printed each line in the format “n x i = result”.

Code Snippet:

```
interest calc.py    multiplication table.py    Vowel
n = int(input("Enter a number for table: "))
for i in range(1, 11):
    print(f"{n} x {i} = {n * i}")
```

Output Snippet:

```
Run    multiplication table ×
C:\Users\Ali\Desktop\Internship\.venv
Enter a number for table: 6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60

Process finished with exit code 0
```

Learning and Challenges

- I learned how to use for loops to repeat an action a fixed number of times.
- I practiced formatting output with f-strings to present results clearly.

Task 16:

Sum numbers divisible by 3.

Step-by-Step Instructions

1. I created a file named sum divisible by 3.py.
2. I read an integer **N** from the user.
3. I used a generator expression to sum all numbers from 1 to **N** that are divisible by 3.
4. I printed the total sum.

Code Snippet:

```
interest calc.py    multiplication table.py    sum divisible by 3.py
N = int(input("Enter N: "))
total = sum(i for i in range(1, N + 1) if i % 3 == 0)
print(f"Sum of multiples of 3 up to {N}: {total}")
```

Output Snippet:

```
Run    sum divisible by 3
C:\Users\Ali\Desktop\Internship\.venv\
Enter N: 99
Sum of multiples of 3 up to 99: 1683

Process finished with exit code 0
```


Learning and Challenges

- I learned to combine range(), if filtering, and sum() in a single, efficient expression.
- I practiced using the modulus operator (%) to identify divisible numbers.

Tasks – Weekly Challenge

Task 17:

CLI Unit Converter: length, weight, temperature menus + loops & conditionals.

Step-by-Step Instructions

1. I created a file named unit converter.py.
2. I wrote a while True loop to display a main menu with options for length, weight, temperature, and exit.
3. I used if/Elif/else to handle the user's choice.
4. Inside each branch, I read the input value, performed the conversion, and printed the result.
5. I included an exit option to break the loop and end the program.

Code Snippet:

interest calc.py | Unit Converter.py | multiplication table.py

```python
def length_menu():  1 usage
    m = float(input("Enter length in meters: "))
    print(f"{m} m = {m*100:.2f} cm")

def weight_menu():  1 usage
    kg = float(input("Enter weight in kilograms: "))
    print(f"{kg} kg = {kg*1000:.2f} g")

def temp_menu():  1 usage
    c = float(input("Enter temperature in °C: "))
    f = (c * 9/5) + 32
    print(f"{c} °C = {f:.2f} °F")

while True:
    print("\nUnit Converter")
    print("1) Length  2) Weight  3) Temperature  4) Exit")
    choice = input("Select option: ")
    if choice == '1':
        length_menu()
    elif choice == '2':
        weight_menu()
    elif choice == '3':
        temp_menu()
    elif choice == '4':
        print("done")
        break
    else:
        print("Invalid choice, try again.")
```

Output Snippet:

```
Run    Unit Converter

C:\Users\Ali\Desktop\Internship\.venv\Scripts\python.exe

Unit Converter
1) Length  2) Weight  3) Temperature  4) Exit
Select option: 1
Enter length in meters: 10
10.0 m = 1000.00 cm

Unit Converter
1) Length  2) Weight  3) Temperature  4) Exit
Select option: 4
done

Process finished with exit code 0
```

Learning and Challenges

1. I learned to build a text-based menu using loops and conditionals.
2. I practiced structuring code into functions and handling user input for different conversion tasks.