TechSaksham Training - Day 4 Notes

🔹 Problem-Solving with Loops

Palindrome Check (Without Built-in Functions & Type Casting)

- Use a **while loop** to check if a number is a palindrome.
- Reverse the number mathematically instead of converting it to a string.

Armstrong Number Check

- Find the **length of the number** using a count variable.
- Raise each digit to the power of the length and sum them.
- If the sum equals the original number, it is an Armstrong number.

Prime Number Verification

- Take input from the user.
- Loop through numbers from **2 to num-1** and count divisors.
- If no divisors are found, it is a prime number; otherwise, it is not.

Generating Prime Numbers (1 to 100)

- Use a **while loop** and iterate over numbers from 1 to 100.
- Skip numbers less than 2.
- Check for prime numbers and store them in a **list**.

Separate Prime & Non-Prime Numbers

- Loop through numbers **1 to 100**.
- Store **prime numbers** in one list and **non-prime numbers** in another.

Printing Even Numbers (1 to 1000) in Two Lines

```
print([num for num in range(2, 1001, 2)])
```

🔹 Game Development: Word Simulator

- Objective: Rearrange letters of a randomly chosen word to form a valid match.

- **Implementation Steps:**
    1. Use two lists: `original_words` and `rearranged_words`.
    2. Select a **random word** from `original_words`.
    3. Ask the user to input a rearranged version.
    4. Validate input against `rearranged_words`.
    5. Award points for correct matches.
    6. Display final score.

🔹 Functions in Python

Function Basics

- A function is a **block of reusable code**.
- Defined using `def` and called to execute.
- Function names are **case-sensitive** and should follow naming conventions.

Types of Functions

1. **User-Defined Functions**

Simple Function:

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

-

Function with Parameters:

```
 def info(name, age):
    print(f"Name: {name}, Age: {age}")
info("Tanisha", 20)
```

-

Keyword Arguments:

```
 def details(name, age, city):
    print(f"{name} is {age} years old from {city}.")
details(age=20, city="Solapur", name="Tanisha")
```

-

Default Parameter Function:

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

- ○

2. **Arbitrary Functions**

Variable Length Arguments (`*args`):

```
 def add_numbers(*nums):
    return sum(nums)
print(add_numbers(1, 2, 3, 4, 5))
```

- ○

Keyword Variable Length (`**kwargs`):

```
 def student_data(**details):
    print(details)
student_data(name="Tanisha", age=20, city="Solapur")
```

- ○

3. **Lambda Function**

Anonymous function for simple operations:

```
 square = lambda x: x * x
print(square(5))
```

- ○

4. **Recursive Function**

Calls itself to solve a smaller instance:

```
 def factorial(n):
    return 1 if n == 0 else n * factorial(n-1)
print(factorial(5))
```

- ○

5. **Generator Function**

Uses `yield` instead of `return`:

```
 def generate_numbers():
    yield 1
    yield 2
gen = generate_numbers()
print(next(gen))
```

- ○

🔹 Advanced Problem-Solving

Pattern Printing

```
   a
  bbb
 ccccc
ddddddd
```

- **Constraints:**
  - Use **while loop**.
  - Do not use any **data type conversion**.
  - Hint: Use **ASCII values**.

Building a Restaurant Ordering System

- Ask for **customer's name**.
- Allow selection of a **food category**.
- Display **available food items with prices**.
- If selecting **curry, prompt for bread selection**.
- Calculate and display the **total bill**.
- Implement **PIN-based payment**.
- Confirm order placement.

Additional Coding Challenges

✅ Function to add any number of values. ✅ Function to calculate length **without built-in function**. ✅ Convert string to **uppercase, lowercase, capitalize, and title case**. ✅ Print numbers **1 to 10 without loops**. ✅ Print multiplication **tables (1-10) without loops**. ✅ Square each number in a list **without creating a new list**.

💡 Summary:

Day 4 focused on **loop-based problem-solving, game development, functions, pattern printing, and advanced problem statements**. Looking forward to **applying Python in real-world applications!** 🚀