>>>> Day 4:

Point 1 — Using math.sqrt() in Python

- The `math.sqrt()` function is used to find the **square root** of a number.
- It requires importing the `math` module.

Syntax:

```
import math
math.sqrt(number)
```

-
- Returns a **floating-point value**.

Example:

```
import math
print(math.sqrt(25))  # Output: 5.0
```

-
- It raises a `ValueError` for negative numbers (for which `cmath.sqrt()` can be used).

Point 2 — Converting Decimal to Binary, Octal, and Hexadecimal

- Decimal numbers can be converted using **built-in functions**: `bin()`, `oct()`, and `hex()`.
- Each returns a string with prefixes: `0b`, `0o`, `0x`.
- Slicing `[2:]` removes these prefixes.

Example (using functions):

```
decimal = int(input("Enter a number: "))
print("Binary:", bin(decimal)[2:])
print("Octal:", oct(decimal)[2:])
print("Hexadecimal:", hex(decimal)[2:])
```

-

Logical conversion (without functions) can be done using repeated **division and remainder** logic:

```
while n > 0:
    remainder = n % base
    result = str(remainder) + result
    n = n // base
```

-
- The same logic applies for **binary (base 2), octal (base 8), and hexadecimal (base 16 using digits 0–F)**.

Point 3 — Variable Declaration and Initialization in Python

- Python uses **dynamic typing**: you don't need to declare the data type of a variable.
- A variable is created when a value is assigned using `=`.

Example:

```
x = 10        # int
name = "Adarsh"  # str
pi = 3.14     # float
is_valid = True  # bool
```

-
- Python automatically infers the type based on the assigned value.
- **Using `input()`**:

- All user inputs are stored as **strings (`str`)**.

```
data = input("Enter something: ")
print(type(data))  # <class 'str'>
```

-

To convert input to other data types:

```
num = int(input("Enter an integer: "))
price = float(input("Enter price: "))
```

-

Operation	Example	Result Type	Notes
Assignment	`x = 10`	`int`	Auto-detected
Assignment	`x = "10"`	`str`	Quotes → string
Input	`x = input()`	`str`	Always string
Type Cast	`int(input())`	`int`	Convert manually

Point 4 — While Loop in Python

- The `while` loop executes a block of code **repeatedly** as long as a condition is `True`.

Syntax:

```
while condition:
    # code block to execute
```

-
- The **condition is checked before each iteration**. If `False`, loop stops.
- Often used when the **number of iterations is not fixed**.

Example 1 — Counting from 1 to 5:

```
i = 1
```

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

Output:

```
1
2
3
4
5
```

Example 2 — Factorial using while loop:

```
n = int(input("Enter a number: "))
factorial = 1
i = 1
while i <= n:
    factorial *= i
    i += 1
print(f"The factorial of {n} is {factorial}")
```