user defined function

```
def add(x,y)
  return x+y
# usage
result = add(2,3)
print(result)
```

User input

User input in Python allows interaction with the program by capturing information entered by the user during runtime.

```
# Taking user input in Python.
num = input("Enter a number:")
print(num)
# type checking of a user input variable
type(num)
# Output
<class 'str'>
```

User Input & Type Casting

Type casting involves converting this input into a specific data type for accurate processing

```
# Taking user input in Python.
num = input("Enter a number.")
print(num)
# type casting
num = float(num)
# type checking of a user input variable.
type(num)
# Output
<class 'float'>
```

Conditional Statement

```
# Checking if a number is even or odd.
def check_even_odd(number):
    if number % 2 == 0:
        return 'Even'
    else:
        return 'Odd'
# Example usage:
result = check_even_odd(7)
print(result)

# Exa Determining the sign of a number
def check_number_sign(number):
    if number > 0:
        return 'Positive'
    elif number < 0:
        return 'Negative'
    else:
        return 'Zero'

# Example usage:
result = check_number_sign(-5)
print(result)
```

While Loop

① A loop is a fundamental programming concept that allows to repeat a certain block of code.

② Loops help in automating repetitive tasks and improving code efficiency.

The "while" loop is a control flow statement that repeatedly executes a block of code as long as a specified condition is true.

A 'while True' loop creates an infinite loop since the condition always true.

Exit Mechanism: "break" statement.

Example

```
while True:
    user_input = input("Enter something (type 'quit' to exit): ")
    if user_input.lower() == "quit":
        print("Exiting the loop.")
        break  ## Break out of the loop when the user enters "quit"
    print(f"You entered: {user_input}")
```