MODuLE-1: Topic Variable ,condition, loop and funtion

Variable: In Python, a variable is a named location in memory that is used to store data. You can think of it as a container that holds a value.

Key point about Variable

- **No declaration:**

  Unlike some other languages, Python does not require you to declare a variable before using it. You simply assign a value to a name, and the variable is created.
- **Dynamic typing:**

  Python is dynamically typed, which means that the type of a variable is determined at runtime. You don't need to explicitly specify the type when you create a variable.
- **Assignment operator:**

  The = operator is used to assign a value to a variable.
- **Naming rules:**
  Variable names must start with a letter or an underscore (_), and can contain letters, numbers, and underscores. They are case-sensitive.

SYNTAX: variable=value:

```
EX:  x = 10     # Assigns the integer value 10 to the variable 'x'
name = "sathi"    # Assigns the string value "sathi" to the variable 'name'
print(x)    # Output: 10
print(name)     # Output: sathi
```

CONDITION STATEMENT: Conditional statements in Python are used to execute certain blocks of code based on specific conditions. These statements help control the flow of a program, making it behave differently in different situations.

If Conditional Statement: It is the simplest form of a conditional statement. It executes a block of code if the given condition is true.

```
Syntax: if condition:
              statement
```

```
Eg:   age = 20
      if age >= 18:
       print("Eligible to vote.")
```

Output: Eligible to vote.

If else Conditional Statements: It allows us to specify a block of code that will execute if the condition(s) associated with an if or elif statement evaluates to False. Else block provides a way to handle all other cases that don't meet the specified conditions.

```
Syntax: if cond:
            Statement
          else:
             statement
```

ex: age = 10

```
if age <= 12:
   print("Travel for free.")
else:
 print("Pay for ticket.")
```

elif Statement: elif statement in Python stands for "else if." It allows us to check multiple conditions , providing a way to execute different blocks of code based on which condition is true. Using elif statements makes our code more readable and efficient by eliminating the need for multiple nested if statements.

Example: age = 25

```
if age <= 12:
   print("Child.")
elif age <= 19:
   print("Teenager.")
elif age <= 35:
   print("Young adult.")
else:
print("Adult.")
```

Nested if..else Conditional Statements: **Nested if..else means an if-else statement inside another if statement. We can use nested if statements to check conditions within conditions.**

Example:

```
age = 70
is_member = True

if age >= 60:
   if is_member:
      print("30% senior discount!")
   else:
      print("20% senior discount.")
else:
   print("Not eligible for a senior discount.")
```

Ternary Conditional Statement : **It is a compact way to write an if-else condition in a single line. It’s sometimes called a "conditional expression.**

Example: # Assign a value based on a condition

```
age = 20
s = "Adult" if age >= 18 else "Minor"
print(s)
```

Match-Case Statement in Python: **match-case statement is Python's version of a switch-case found in other languages. It allows us to match a variable's value against a set of patterns**

EXAMPLE : **number = 2**

```
match number:
   case 1:
      print("One")
   case 2 | 3:
      print("Two or Three")
   case _:
      print("Other number")
```

Loop IN Python: **Loops in Python are used to repeat actions efficiently. The main types are For loops (counting through items) and While loops (based on conditions). Additionally, Nested Loops allow looping within loops for more complex tasks.**

1. For Loop

Definition: A for loop is used to iterate over a sequence (such as a list, tuple, or string) and execute a block of code for each item.

Syntax:

```
for variable in iterable:
    # code to execute
```

Example:

```
fruits = ['apple', 'banana', 'cherry']
for fruit in fruits:
    print(fruit)
```

2. While Loop

Definition: A while loop is used to execute a block of code repeatedly as long as a certain condition is true.

Syntax:

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

Example:

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

3. Nested Loop

Definition: A nested loop is a loop within another loop.

Syntax:

```
for variable1 in iterable1:
    for variable2 in iterable2:
        # code to execute
```

Example:

```
colors = ['red', 'green', 'blue']
shapes = ['circle', 'square', 'triangle']
for color in colors:
    for shape in shapes:
        print(f"{color} {shape}")
```

4. Break and Continue

Definition: Break and continue statements are used to control the flow of loops.

- Break: Exits the loop entirely.
- Continue: Skips the current iteration and moves on to the next one.

Syntax:

```
for variable in iterable:
    if condition:
        break
    elif condition:
        continue
    # code to execute
```

Example:

```
for i in range(10):
    if i == 5:
        break
    elif i == 3:
        continue
    print(i)
```

Function in Python: **a function is a block of code that can be executed multiple times from different parts of a program. It's a way to group a set of statements together to perform a specific task.**

Key characteristics of Python functions:

1. Reusability: Functions can be called multiple times from different parts of a program.

2. Modularity: Functions help break down a large program into smaller, manageable chunks.

3. Abstraction: Functions can hide complex implementation details and provide a simple interface.

Basic syntax of a Python function:

```
def function_name(parameters):
    # function body
    # return statement (optional)
```

Example of a simple Python function:

```
def greet(name):
    print(f"Hello, {name}!")
greet("John")
# Output: Hello, John!
```

Function arguments and parameters:

1.Positional arguments: Passed to the function in the order they are defined.
2. Keyword arguments: Passed to the function using the parameter name.
3. Default arguments: Parameters with default values, which can be overridden when calling the function.