IMPERATIVE PROGRAMMING PARADIGM

Mrs. S. Niveditha
Assistant Professor (Sr. G)
Department of Computer Science and Engineering SRMIST, Vadapalani Campus

Topics

- Program state, instructions to change the program state

Combining Algorithms and Data Structures ■ Imperative Vs Declarative Programming

■ Other Languages: PHP, Ruby, Perl, Swift

■ Demo: Imperative Programming in Python

INTRODUCTION

■ In a computer program, a variable stores the data. The contents of these locations at any givenpointin the program's execution are called the program's state. Imperative programming is characterized by programming with state and commands which modify the state.

- The first imperative programming languages were

  machinelanguages.

Machine Language

- Each instruction performs a very specific task, such as a load, a jump,or an ALU operation on a unit of data in a CPU register or memory. For example:
  - rs, rt, and rd indicate register operands – shamt gives a shift amount
  - the address or immediate fields contain an operand directly

Assembly Code

- 10110000 01100001

- Equivalent Assembly code
- B0 61
  - B0 - ‘Move a copy of the following valueinto AL’ (AL is a register)
  - 61 is a hexadecimal representation of the value01100001 – 97
- Intel assembly language
  - MOV AL, 61h; Load AL with 97 decimal (61 hex)

Other Languages

- FORTRAN(FORmula TRANslation) was the first high level language to gain wide acceptance. It was designed for scientific applications and featured an algebraic notation, types, subprograms, and formatted input/output.

- COBOL (COmmon Business Oriented Language) was designed at the initiative of the U. S. Department of Defence in 1959 and implemented in 1960 to meet the need for business data processing applications.
- ALGOL 60 (ALGorithmic Oriented Language) was designed in 1960 by an international committee for use in scientific problem solving

Evolutionary developments

- Control statements
- Recursion
- Formatted IO
- Dynamic storage allocation
- Linked structures
- File manipulation
- Record
- Block structure
- Subprogram

OVERVIEW

- In imperative programming, a name may be assigned to a value and later reassigned to another value.
- The collection of names and the associated values and the location of control in the program constitute the state.
- The state is a logical model of storage whichis an association between memory locations and values.
- A program in execution generates a sequence of states.
- The transition from one state to the next is determined by assignment operations and sequencing commands.

Highlights on

- Assignment,
- goto commands

- structured programming

Command

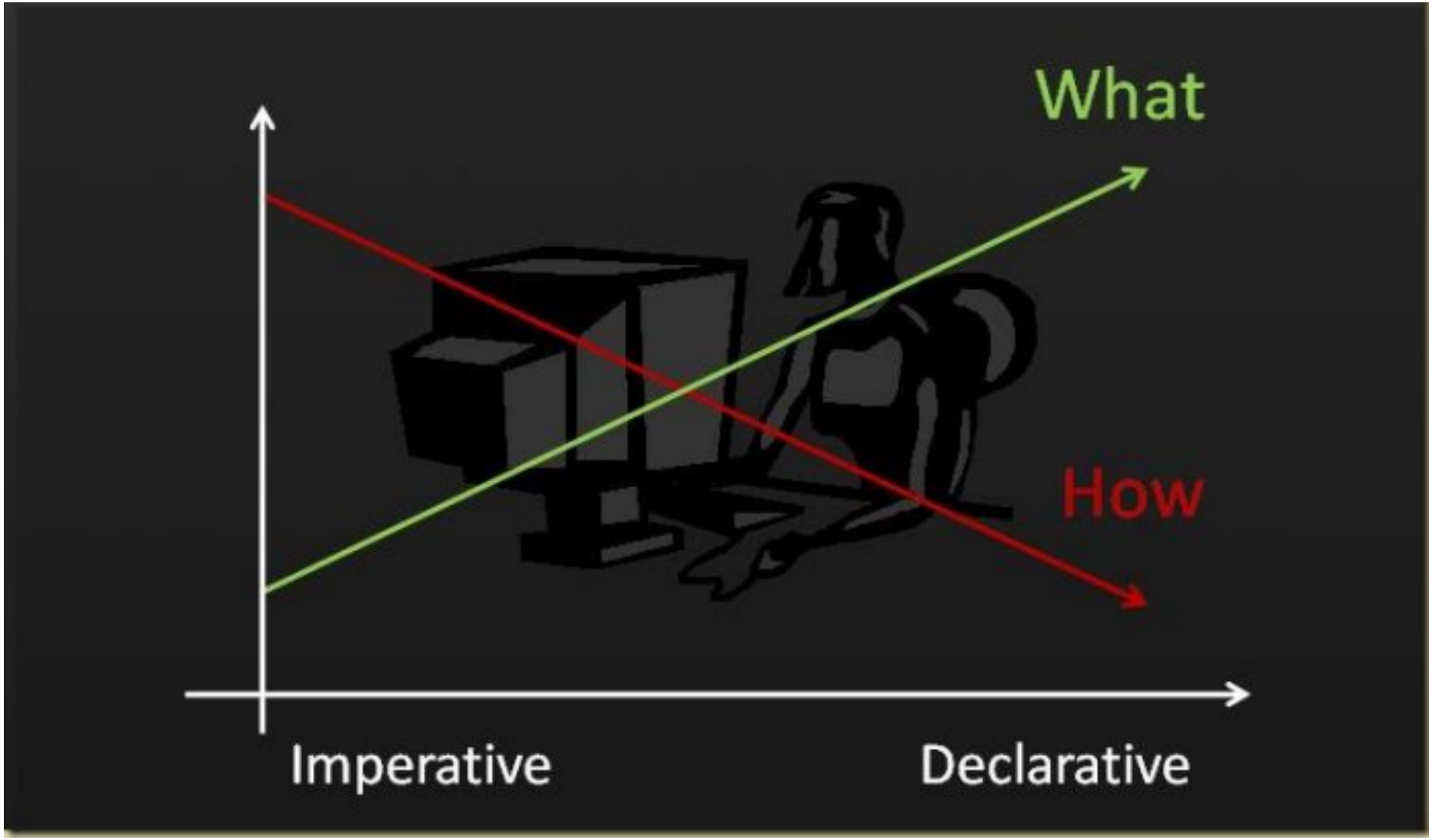
- Statement
- Procedure
- Control-flow

- Imperative language

Assertions

- Axiomatic semantics
- State
- Variables
- Instructions
  - Control structures

Declarative Vs Imperative



Declarative Vs Imperative

Declarative

```
# Declarative
small_nums = [x for x in range(20) if x < 5]
```

#

Imperative

```
# Imperative
small = []
for i in range(20):
    if i < 5:
        small.append(i)
```

```
0
1
2
3
4
```

DEMO

An algorithm to add two numbers entered by user

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum. sum←num1+num2

Step 5: Display sum

Step 6: Stop

Addition two numbers entered by user

```
num1 = 0
num2 = 0
sum = 0
num1 = input("Enter the First number ")
num2 = input("Enter the Second number ")
sum = int(num1) + int(num2)
print("\nSum:", sum)
```

```
Enter the First number 10

Enter the Second number 20

Sum: 30
```

An Algorithm to Get n number, print the same and find Sum of n numbers

Step 1: Start

Step 2: Declare variable sum = 0.

Step 3: Get the value of limit “n”.

Step 4: If limit is reached, goto Step 7 else goto Step 5

Step 5: Get the number from user and add it to sum Step

6: Goto Step 4

Step 7: If limit is reached, goto Step 9 else goto Step 8

Step 8: Print the numbers

Step 9: Goto Step 7

Step 9: Display sum

Step 10: Stop

```
sum = 0
num=[]
n = input("Enter the Total number of values ")
num = [ int(input("Enter value ")) for i in range(int(n))]
for i in range(int(n)):
    sum = sum + num[i]
print("\nYou have entered")
for i in range(int(n)):
    print(num[i])
print("..and the sum is", sum)
```

```
Enter the Total number of values 5

Enter value 55

Enter value 62

Enter value 12

Enter value 34

Enter value 20

You have entered
55
62
12
34
20
..and the sum is 183
```

```
Dict = {1: 'Song A', 2: 'Song B', 3: 'Song C'}

n=input("Select the number to play your favorite song ")
# accessing a element using key
print("You are listening to ")
print(Dict[int(n)])
```

```
Select the number to play your favorite song 3
You are listening to
Song C
```