CSEN 102– Introduction to Computer Science

Lecture 3:
Algorithmic Problem Solving
Conditional Operations

Prof. Dr. Slim Abdennadher
Dr. Mohammed Salem
slim.abdennadher@guc.edu.eg,
mohammed.salem@guc.edu.eg

German University Cairo, Department of Media Engineering and Technology

13.10.2018 - 18.10.2018

Synopsis

Synopsis

- What is computer science?

Synopsis

- What is computer science?
- What is an algorithm?

Synopsis

- What is computer science?
- What is an algorithm?

Definition (Algorithm)

An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

Synopsis

- What is computer science?
- What is an algorithm?

Definition (Algorithm)

An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

Synopsis

- What is computer science?
- What is an algorithm?

Definition (Algorithm)

An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

- Why python?

Synopsis

- What is computer science?
- What is an algorithm?

Definition (Algorithm)

An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

- Why python?
- What are the necessary elements for sequential algorithms?

Synopsis

- What is computer science?
- What is an algorithm?

Definition (Algorithm)

An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

- Why python?
- What are the necessary elements for sequential algorithms?
  - Input (*e. g.*, "`A = eval(input())`")

Synopsis

- What is computer science?
- What is an algorithm?

Definition (Algorithm)

An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

- Why python?
- What are the necessary elements for sequential algorithms?
  - Input (*e. g.*, "`A = eval(input())`")
  - Output (*e. g.*, "`print(A)`", or "`print("text")`")

Synopsis

- What is computer science?
- What is an algorithm?

Definition (Algorithm)

An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.

- Why python?
- What are the necessary elements for sequential algorithms?
  - Input (*e. g.*, "`A = eval(input())`")
  - Output (*e. g.*, "`print(A)`", or "`print("text")`")
  - Calculation, manipulation (*e. g.*, "`A = B + C`")

Sequential operations

Example (See last lecture)

For a given number of eggs, find out how many dozen eggs we have and how many extra eggs are left over.

Sequential operations

Example (See last lecture)

For a given number of eggs, find out how many dozen eggs we have and how many extra eggs are left over.

```
1  eggs = eval(input())
2  dozens = int(eggs / 12)
3  extras = eggs - (dozens * 12)
4  print("Your number of eggs is ")
5  print(dozens)
6  print(" dozen(s) and ")
7  print(extras)
8  print(" extra(s)")
```

Where the function **int** rounds down the result to an integer. For example **int**(10/3) = 3.

Sequential operations

Example (See last lecture)

For a given number of eggs, find out how many dozen eggs we have and how many extra eggs are left over.

```
1  eggs = eval(input())
2  dozens = int(eggs / 12)
3  extras = eggs - (dozens * 12)
4  print("Your number of eggs is ")
5  print(dozens)
6  print(" dozen(s) and ")
7  print(extras)
8  print(" extra(s)")
```

- Let the input be 27
- What is the output?

Where the function `int` rounds down the result to an integer. For example `int(10/3) = 3`.

How to construct an algorithm

- Identify the input of the algorithm
- Introduce variables for
  - Input
  - (intermediate) results
- Analyze the task into sequential steps
- Provide for detailed output

Objectives

By the end of this lecture, you should be able to:

- Design algorithms using conditional operations

Algorithms: operations

Algorithms can be constructed by the following operations:

- Sequential Operation
- Conditional Operation
- Iterative Operation

Algorithms: operations

Algorithms can be constructed by the following operations:

- Sequential Operation
- Conditional Operation
- Iterative Operation

Conditional operation – idea

Conditional operation – idea

Decision

```
1  nameOfSinger = input()
2  if nameOfSinger == 'Mohamed Mounir':
3      print('I will go home')
4  else:
5      print('I will stay at the GUC')
```

Conditional operation – principle

Conditional operation – principle

- Rationale
  - Determines whether or not a condition is true; and based on whether or not it is true; selects the next step to do

Conditional operation – principle

- Rationale
  - Determines whether or not a condition is true; and based on whether or not it is true; selects the next step to do
- Notation
  - Use the same primitives as before plus the following:

```
1 if condition:
2         # <operations for the then-part>
3 else
4         # <operations for the else-part>
```

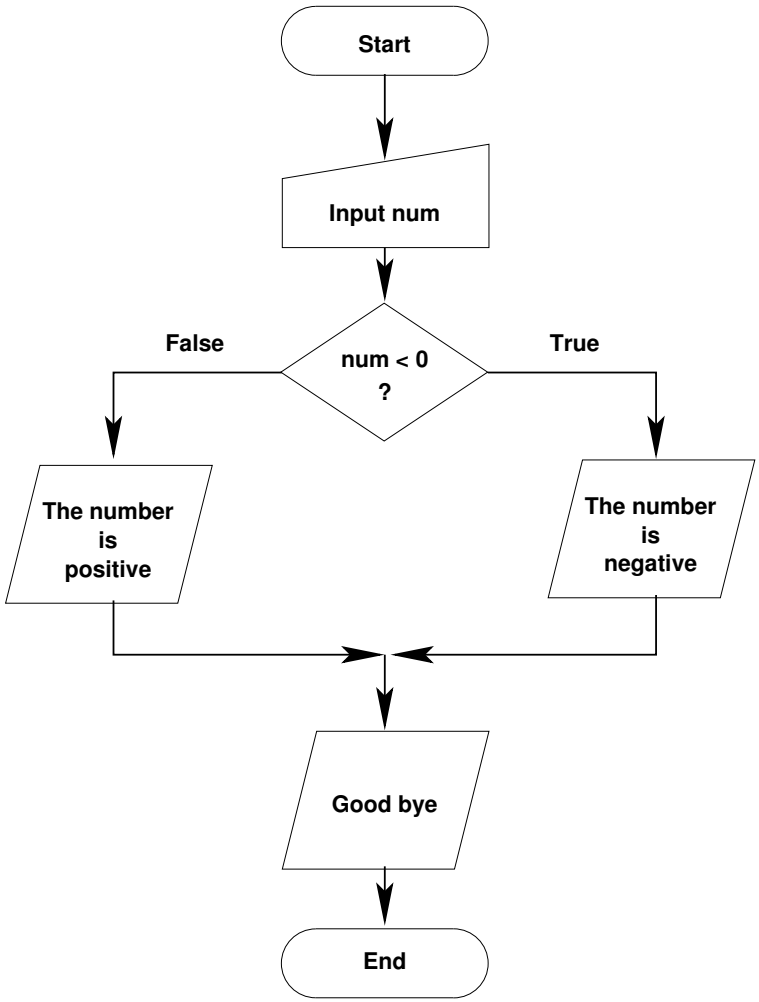
Conditional operation – principle

- Rationale
  - Determines whether or not a condition is true; and based on whether or not it is true; selects the next step to do
- Notation
  - Use the same primitives as before plus the following:

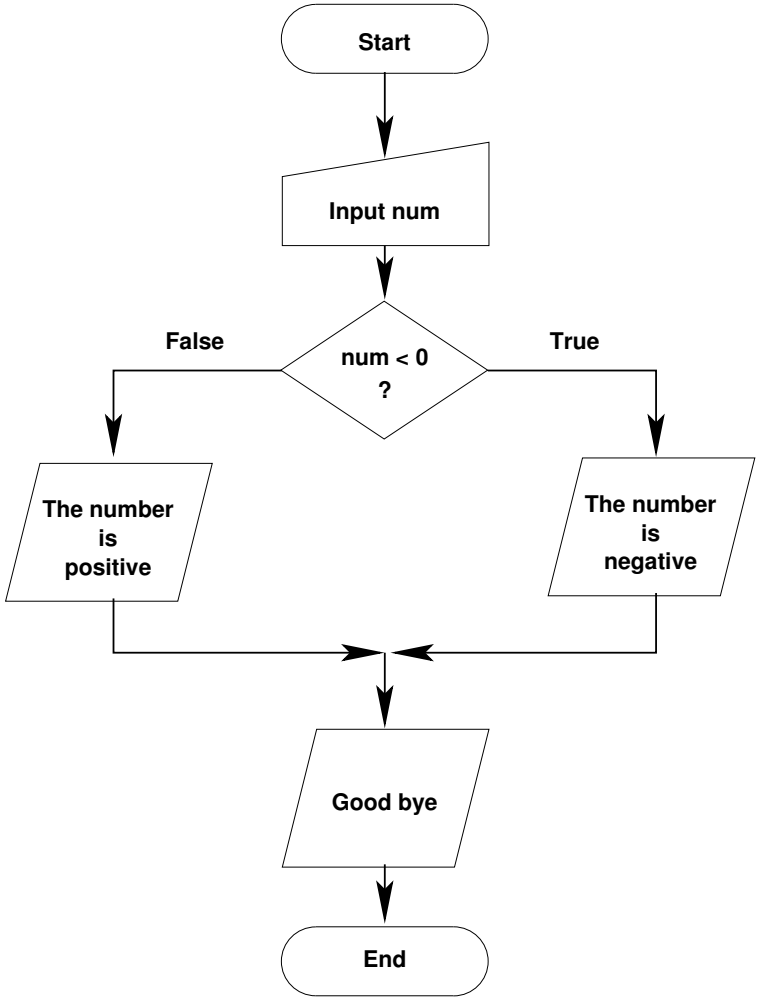
```
1  if condition:
2          # <operations for the then-part>
3  else
4          # <operations for the else-part>
```

- Execution
  - Evaluate *<condition>* expression to see whether it is true or false.
  - If true, then execute operations in **if**-part
  - Otherwise, execute operations in **else**-part

Conditional operation – diagram



Conditional operation – diagram




```
1  num = eval(input())
2  if num < 0:
3      print("The number is negative")
4  else:
5      print("The number is positive")
6
7  print("Good-bye for now")
```

Conditional operation – examples

Example 1:

Write an algorithm to compute the absolute value of a given number.

Conditional operation – examples

Example 1:

Write an algorithm to compute the absolute value of a given number.

```
1  Number = int(input())
2
3  if (Number >= 0):
4          Value = Number
5  else:
6          Value = (-1) * Number
7
8  print(Value)
```

Conditional operation – examples

Example 2:

Give the user a choice of seeing the area or the circumference of a circle given its radius.

Conditional operation – examples

Example 2:

Give the user a choice of seeing the area or the circumference of a circle given its radius.

```
1  radius = eval(input())
2  response = input("Type A for area or C for circumference")
3  if (response == "A"):
4   area = (radius * radius * 3.14)
5   print(area)
6  else:
7   circumference = (2 * radius * 3.14)
8   print(circumference)
```

Conditional Operation – examples

Example 3:

Write an algorithm to convert Euro (EUR, €) to Egyptian Pound (EGP, £E) and Egyptian Pound to Euro. The inputs to your algorithm are the following:

- Amount of money to be converted
- Conversion Type (*i. e.*, 1 for EUR to EGP and 2 for EGP to EUR)
- Exchange Rate (*i. e.*, the EGP equivalent for 1 EUR)

Conditional Operation – examples

Example 3:

Write an algorithm to convert Euro (EUR, €) to Egyptian Pound (EGP, £E) and Egyptian Pound to Euro. The inputs to your algorithm are the following:

- Amount of money to be converted
- Conversion Type (*i. e.*, 1 for EUR to EGP and 2 for EGP to EUR)
- Exchange Rate (*i. e.*, the EGP equivalent for 1 EUR)

```
1 amount, type, rate = eval(input()), eval(input()), eval(inp
2 if type == 1:
3     amount = amount * rate
4 else:
5     amount = amount / rate
6 print(amount)
```

Compounded conditions

Conditions may be compounded using AND, OR and NOT.

- E1 or E2: true if at least one of them is true; false otherwise.
- E1 and E2: true if both are true; false otherwise.
- not E: true if E is false and false if E is true.

Compounded conditions

Conditions may be compounded using AND, OR and NOT.

- E1 or E2: true if at least one of them is true; false otherwise.
- E1 and E2: true if both are true; false otherwise.
- not E: true if E is false and false if E is true.

Find the sum of three positive numbers

```
1  A, B, C = int(input()), int(input()), int(input())
2  if (A > 0) and (B > 0) and (C > 0):
3    Sum = (A+B+C)
4    print(Sum)
```

Conditional algorithms with more than two choices

Nested if-statement

Conditional algorithms with more than two choices

Nested if-statement

```
1  if first_condition:
2      # <do first thing>
3  else:
4      if second_condition:
5          # <do second thing>
6      else:
7          # <do something else>
```

Nested if-statement – examples

Example 1

Algorithm to find the largest of three numbers.

Nested if-statement – examples

Example 1

Algorithm to find the largest of three numbers.

```
1  A, B, C = eval(input()), eval(input()), eval(input())
2  if A >= B:
3      if A >= C:
4          print(A)
5      else:
6          print(C)
7  else:
8      if B >= C:
9          print(B)
10     else:
11         print(C)
```

Nested if-statement – examples

Example 2

Write an algorithm that reads each student's marks, print either a grade or an error message. Students marks in a class are graded on the following policy:

- A: 85-100
- B: 74-85
- C: 60-74
- D: 50-60
- F: $<$50

Nested if-statement – examples

Solution with if and else

```
1  Mark = eval(input())
2  if(Mark >=0):
3      if (Mark >100):
4          print("invalid␣mark")
5      else:
6          if (Mark <50):
7              print("grade is F")
8          else:
9              if (Mark <60):
10                 print("grade is D")
11             else:
12                 if (Mark <74):
13                     print("grade is C")
14                 else:
15                     if (Mark <85):
16                         print("grade is B")
17                     else:
18                         print("grade is A")
```

Nested if-statement – examples

Solution with if and elif

```
1  Mark = eval(input())
2
3  if(Mark >=0):
4      if (Mark >100):
5          grade = "invalid mark"
6      elif (Mark <50):
7          grade = "grade is F"
8      elif (Mark <60):
9          grade = "grade is D"
10     elif (Mark <74):
11         grade = "grade is C"
12     elif (Mark <85):
13         grade = "grade is B"
14     else:
15       grade = "grade is A"
16
17 print(grade)
```

Nested if-statement – examples

Example 3

Given an employee's eligible medical expenses for a calendar year, write an algorithm which computes the amount of reimbursement from group medical insurance.

- The insurance does not cover the first 100 LE of medical expenses.
- It pays 90% of the remaining amount in the first 2000 LE of expenses and 100% of any additional expenses.

Nested if-statement – examples

```
1  LL = 100
2  UL = 2000
3  Expense = int(input())
4
5  if (Expense < LL):
6          Refund = 0
7  else:
8          if (Expense < UL):
9                  Refund = 0.9 * (Expense-LL)
10         else:
11                 Refund = 0.90 * (UL-LL) + (Expense - UL)
12
13 print(Refund)
```