

CSEN 102– Introduction to Computer Science

Lecture 2: Python and Sequential Algorithms

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

06.10.2018 - 11.10.2018

Synopsis

Synopsis

- What is computer science?

Synopsis

What is **computer science**?

What is an **algorithm**?

Synopsis

What is **computer science**?

What is an **algorithm**?

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**?

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**.

What is a **computing agent**?

Synopsis

What is **computer science**?

What is an **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**.

What is a **computing agent**?

Representing algorithms

What language to use?

Expressive

Clear, precise and unambiguous

For example, we could use:

Natural Languages (e. g., English)

Formal Programming Languages (e.g. Java, C++)

Something close?

Representing algorithms: Natural languages

Example

Given is a natural number n . Compute the sum of numbers from 1 to n .

Representing algorithms: Natural languages

Example

Given is a natural number n . Compute the sum of numbers from 1 to n .

Representation with Natural Language

Initially, set the value of the variable **result** to **0** and the value of the variable **i** to **1**. When these initializations have been completed, begin looping until the value of the variable **i** becomes greater than **n**. First, add **i** to **result**. Then add **1** to **i** and begin the loop all over again.

Representing algorithms: Natural languages

Example

Given is a natural number n . Compute the sum of numbers from 1 to n .

Representation with Natural Language

Initially, set the value of the variable **result** to **0** and the value of the variable **i** to **1**. When these initializations have been completed, begin looping until the value of the variable **i** becomes greater than **n**. First, add **i** to **result**. Then add **1** to **i** and begin the loop all over again.

Disadvantages:

- too verbose
- unstructured
- too rich in interpretation (ambiguous)
- imprecise

Representing algorithms: Formal programming language

Example

Given is a natural number n . Compute the sum of numbers from 1 to n .

Representing algorithms: Formal programming language

Example

Given is a natural number n . Compute the sum of numbers from 1 to n .

Representation with Formal Programming Language (Java)

```
public class Sum {
    public static void main(String[] args)
    {
        int result = 0;
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            result = result + i;
            i = i + 1;}
        System.outp.println(result);}}

```

Representing algorithms: Formal programming language

Disadvantages:

Too many implementation details to worry about

Too rigid syntax

Representing algorithms: Script Programming Language

A Less strict Formal Programming Language:

) JavaScript, Python

We will use Python as it has:

English like constructs (or other natural language) but is still a programming language.

Python-code: Input/output and computation

Python-code: Input/output and computation

Input operations: allow the computing agent to receive from the outside world data values to use in subsequent computations.

General Format:

```
<variable> = eval ( input () )
```

Python-code: Input/output and computation

Input operations: allow the computing agent to receive from the outside world data values to use in subsequent computations.

General Format:

```
<variable> = eval ( input () )
```

Output operations: allow the computing agent to communicate results of the computations to the outside world.

General Format:

```
print ( <variable> )  
print ( "text" )
```

Python-code: Input/output and computation

Input operations: allow the computing agent to receive from the outside world data values to use in subsequent computations.

General Format:

```
<variable> = eval ( input () )
```

Output operations: allow the computing agent to communicate results of the computations to the outside world.

General Format:

```
print ( <variable> )  
print ( "text" )
```

Computation: performs a computation and stores the result.

General Format:

```
<variable> = <expression>
```

Python: What kind of operations do we need?

Example

Given the radius of circle, determine the area.

Python: What kind of operations do we need?

Example

Given the radius of circle, determine the area.

Decide on names for the objects in the problem and use them consistently (e. g., **radius** , **area**). We call them **variables**

Python: What kind of operations do we need?

Example

Given the radius of circle, determine the area.

Decide on names for the objects in the problem and use them consistently (e. g., **radius** , **area**). We call them **variables**

Use the following primitive operations:

Python: What kind of operations do we need?

Example

Given the radius of circle, determine the area.

Decide on names for the objects in the problem and use them consistently (e. g., `radius` , `area`). We call them **variables**

Use the following primitive operations:

Get a value (input) e. g., `radius = eval (input ())`

Python: What kind of operations do we need?

Example

Given the radius of circle, determine the area.

Decide on names for the objects in the problem and use them consistently (e. g., `radius` , `area`). We call them **variables**

Use the following primitive operations:

Get a value (input) e. g., `radius = eval (input ())`

print a value or message (output) e. g.,

`print (area) , or`

`print ("Hello")`

Python: What kind of operations do we need?

Example

Given the radius of circle, determine the area.

Decide on names for the objects in the problem and use them consistently (e. g., `radius` , `area`). We call them **variables**

Use the following primitive operations:

Get a value (input) e. g., `radius = eval (input ())`

print a value or message (output) e. g.,

`print (area)` , or

`print ("Hello")`

Set the value of an object (e. g., `Pi = 3.14`)

) Performs a computation and stores the result.

Python: What kind of operations do we need?

Example

Given the radius of circle, determine the area.

Decide on names for the objects in the problem and use them consistently (e. g., `radius` , `area`). We call them **variables**

Use the following primitive operations:

Get a value (input) e. g., `radius = eval (input ())`

print a value or message (output) e. g.,

`print (area)` , or

`print ("Hello")`

Set the value of an object (e. g., `Pi = 3.14`)

) Performs a computation and stores the result.

Arithmetic operations: e.g. `+` , `*` , `...`

Python: Example

Example

Given the radius of circle, determine the area and circumference.

Names for the objects:

Input: **radius**

Outputs: **area** , **circumference**

Python: Example

Example

Given the radius of circle, determine the area and circumference.

Names for the objects:

Input: **radius**

Outputs: **area** , **circumference**

Algorithm in Python:

```
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)
```

Python: Variables

A **variable** is a named storage

A value can be stored into it, overwriting the previous value

Its value can be copied

Python: Variables

A **variable** is a named storage

A value can be stored into it, overwriting the previous value

Its value can be copied

Example

```
A = 3
```

The variable **A** holds the value three after its execution

```
A = A + 1
```

Same as: add 1 to the value of **A** (**A** is now 4)

Algorithms: operations

Algorithms can be constructed by the following operations:

Sequential Operation

Conditional Operation

Iterative Operation

Sequential operations

Each step is performed in the order in which it is written

Example (1)

Write an algorithm to find the result of a division operation for the given two numbers x and y

Sequential operations

Each step is performed in the order in which it is written

Example (1)

Write an algorithm to find the result of a division operation for the given two numbers x and y

```
1 x = int ( input () )
2 y = int ( input () )
3 quotient = x/y
4 print (quotient)
```

Sequential operations

Each step is performed in the order in which it is written

Example (1)

Write an algorithm to find the result of a division operation for the given two numbers x and y

```
1 x = int ( input () )
2 y = int ( input () )
3 quotient = x/y
4 print (quotient)
```

Let $x=5$ and $y=2$

Sequential operations

Each step is performed in the order in which it is written

Example (1)

Write an algorithm to find the result of a division operation for the given two numbers x and y

```
1 x = int (input ())
2 y = int (input ())
3 quotient = x/y
4 print (quotient)
```

Let $x=5$ and $y=2$

quotient = 2.5

Sequential operations

Example (2)

Algorithm for finding the average of three numbers.

Sequential operations

Example (2)

Algorithm for finding the average of three numbers.

```
1 A = int (input ())
2 B = int (input ())
3 C = int (input ())
4 Sum = A + B + C
5 Average = Sum/3
6 print (Average)
```

Sequential operations

Example (2)

Algorithm for finding the average of three numbers.

```
1 A = int (input ())
2 B = int (input ())
3 C = int (input ())
4 Sum = A + B + C
5 Average = Sum/3
6 print (Average)
```

Let $A=12$, $B=10$, and $C=8$

Sequential operations

Example (2)

Algorithm for finding the average of three numbers.

```
1 A = int ( input () )
2 B = int ( input () )
3 C = int ( input () )
4 Sum = A + B + C
5 Average = Sum/3
6 print (Average)
```

Let $A=12$, $B=10$, and $C=8$

$Sum = 30$

Sequential operations

Example (2)

Algorithm for finding the average of three numbers.

```
1 A = int (input ())
2 B = int (input ())
3 C = int (input ())
4 Sum = A + B + C
5 Average = Sum/3
6 print (Average)
```

Let $A=12$, $B=10$, and $C=8$

$Sum = 30$

$Average = 10$

Sequential operations

Example (3)

The distance between two points $(x_1; y_1)$ and $(x_2; y_2)$ can be calculated using the following equation:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Write an algorithm that calculates the value of d .

Sequential operations

Example (3)

The distance between two points $(x_1; y_1)$ and $(x_2; y_2)$ can be calculated using the following equation:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Write an algorithm that calculates the value of d .

```
1 import math
2 x1, y1 = eval (input ()), eval (input ())
3 x2, y2 = eval (input ()), eval (input ())
4 d = math.sqrt((x2-x1) * (x2-x1) + (y2-y1) * (y2-y1))
5 print (d)
```

Sequential Operations

Example (4)

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 (4)

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` .

Modulus Operator

Example (4)

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

Modulus Operator

Example (4)

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
3 d = eggs//12
4 extra = eggs%12
5 print ("Your number of eggs is ",d, "dozen(s) and ")
6 print (extra, "extra(s) ")
```

Sequential operations

Example (5)

Given a two-digit number, nd the sum of its digits.

Sequential operations

Example (5)

Given a two-digit number, `nd` the sum of its digits.

```
1 num = eval (input ())
2 tens = int (num / 10)
3 ones = num - (tens * 10)
4 s = (tens + ones)
5 print (s)
```

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

Sequential operations

Example (5)

Given a two-digit number, `nd` the sum of its digits.

```
1 num = eval ( input () )
2 tens = int ( num / 10 )
3 ones = num - ( tens * 10 )
4 s = ( tens + ones )
5 print ( s )
```

Let `num=45`

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

Sequential operations

Example (5)

Given a two-digit number, nd the sum of its digits.

```
1 num = eval ( input () )
2 tens = int ( num / 10 )
3 ones = num - ( tens * 10 )
4 s = ( tens + ones )
5 print ( s )
```

Let $num=45$

$s = 9$

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

Modulus Operator

Example (5)

Given a two-digit number, find the sum of its digits.

Modulus Operator

Example (5)

Given a two-digit number, nd the sum of its digits.

```
1 num = eval ( input () )
2 a= num % 10
3 b= num // 10
4 s = a + b
5
6 print (s)
```

Modulus Operator

Example (5)

Given a two-digit number, nd the sum of its digits.

```
1 num = eval ( input () )
2 a= num % 10
3 b= num // 10
4 s = a + b
5
6 print (s)
```

Let num=45

Modulus Operator

Example (5)

Given a two-digit number, nd the sum of its digits.

```
1 num = eval ( input () )
2 a= num % 10
3 b= num // 10
4 s = a + b
5
6 print (s)
```

Let num=45

s = 9

Sequential operations

Example (6)

Write an algorithm that given two Strings returns their sum (concatenation)

Sequential operations

Example (6)

Write an algorithm that given two Strings returns their sum (concatenation)

```
1 x = input () # "Hello"
2 y = input () # "_CSIS104!"
3 z = x + y
4 print (z)     # Prints "Hello_CSIS104!"
```