ITI 1120
Lab #3

Branches

1

Lab. Objective

- Boolean Expressions
- AND versus OR
- Complex Conditions
- Branch Instructions
- Exercises

2

Boolean Expressions

- Returns `true` or `false`
- Translation from software model to Python :

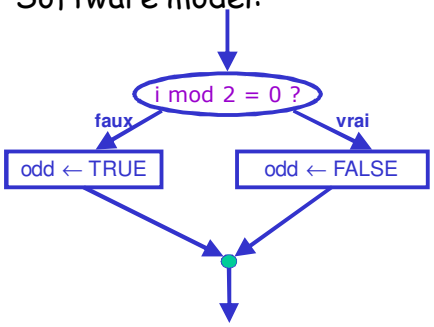
Software Model	Phyton
←	`=` *(not a Boolean expression)*
AND	`and`
OR	`or`
NOT	`not`
A = B	`A == B`
A ≤ B	`A <= B`
A ≥ B	`A >= B`
A ≠ B	`A != B`

3

Boolean Expressions, Example 1

- Derive an algorithm that returns TRUE if an integer I is odd; it should return FALSE otherwise.

Software model:



Phyton:

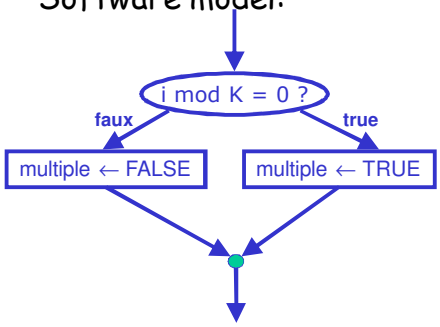
```
# i need a value
i = 5

if (i % 2 == 0):
   odd = False
else:
   odd = True

print (odd)
```

4

Boolean Expressions, Example 2

- Derive an algorithm that returns TRUE if an integer I is a multiple of the positive integer K; and return FALSE otherwise.

Software model:



Python:

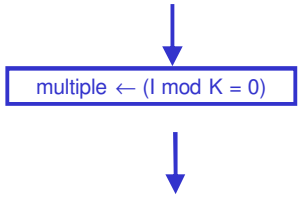
```
# i and k have values…

if i % k == 0:
   multiple = True;
else:
   multiple = False

print (multiple)
```

5

Boolean Expressions, Example 2

- Other approach…

Software model:



Phyton:

```
# i and k have values…

multiple = (i % k == 0)
```

6

AND and OR

- Used to combine conditions
- Use parentheses to make sure that the complex expressions are well represented.
- Wherever you find a test in our pseudocode you can use any boolean expression
- What are the values of the following expressions?

((room = STE0131) OR (room = STE0130)) AND (lab = ITI1120)

It depends…

(I am at home) OR (I am at school) TRUE

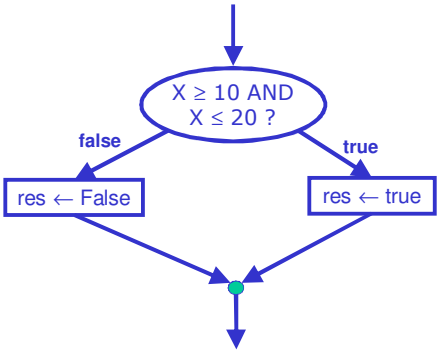
(I am at home) AND (I am at school) FALSE

7

Boolean Expressions, Example 3

- Derive an algorithm that returns True if x is between 10 and 20 (inclusively); or returns False otherwise.

Software model:



Python:

```
# x has a value...

if x >= 10 and x <= 20 :

   res = True

else:

   res = False
```

8

AND versus OR

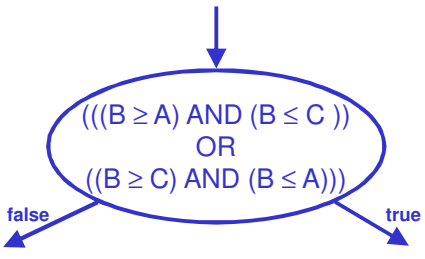
- In the previous page:
  - The Python boolean expression:
    `x >= 10 and x <= 20` was used to détermine if `x` is between 10 and 20.
- How about if we use `or` instead of `and`
  - Assume `x` is 7.
  - If we had `x >= 10 or x <= 20`:
    `x <= 20` is true and thus the whole expression is true; and yet `x` is not between 10 and 20.

9

Boolean expressions, Example 4

- Derive an algorithm that returns True if B is between A and C (but, we do not know if A is bigger than C or the inverse).

Software model:



Python:

```
if  ( b >= a and b <= c ) or
     ( b >= c and b <= a ):

   # b is between a and c
else:

   # b is outside
```

10

Exercises – Some hints

- Develop first algorithm then translate (convert) them in Python

11

Exercise 1

- Derive a Boolean expression that evaluates if an age is between 18 and 55 inclusively.
  - Think about a problem resolution algorithm with one parameter (DATA), an age and a Boolean result – true if age is inside the data set.
  - Convert your algorithm in Python.
  - Your program must ask the user for an age, calculate the Boolean value and print "Tansaction accepted" if the value is true (good age) otherwise "Transaction refused".

12

Exercise 2

- As the activity director at Dow's Lake in Ottawa, you have to recommend appropriate activities to tourists according to temperatures:

temp $\geq$ 80.0:	Swimming
60.0 $\leq$ temp < 80.0:	Soccer
40.0 $\leq$ temp < 60.0:	Volleyball
temp < 40.0:	Skying

- Develop a problem resoltion algorithm with one DATA, the temperature, and with a RESULT, an activity number: 1 (Swimming), 2 (Soccer), 3 (Volleyball), or 4 (Skying).
- Convert the algorithm in Python.
- The program must request the user for a temperature, use the algorithm to get an activity number and display the activity (the name not number).

13

Exercise 3

- Develop a programm that determines if an integer is divisible by 2 and by 3, divisible by 2 or by 3, or is not divisible by neither 2 nor 3.
  - The algorithm, `isDivisible`, analyses the integer and returns an integer that indicates the analysis result: 1 (divisible by 2 and by 3), 2 (divisible by 2 or by 3), 0 (not divisible by neither 2 nor by 3).
  - Convert your votre algorithm in Python.
  - The program must ask the user for an integer, derive the above value and print the result.

14

Exercise 4

- Develop a program that derives the number of real roots in a quadratic equation:

  $ax^2 + bx + c = 0$ (a, b, and c are real constant)

- Derive a problem resolution algorithm from 3 coefficients (DATA) that determines the number of real roots as the result.
- Convert the algorithm in Python.
- The program asks the user for coefficients a, b, and c, derive the number of roots and prints the result (number of roots).

15

Exercise 4 - suite

- Hints for the algorithm:
  - DATA: a, b, and c
  - Remember how to derive the roots (x_1 and x_2)

    roots $= \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ **where** $\Delta = b^2 - 4ac$ (discriminant)

  - The discriminant value determines the number of real roots in the equation:
    - Smaller than 0 – no real roots
    - Equal to 0 – one real root (duplicated)
    - Larger than 0 – two distinct real roots
  - The algorithm provides a RESULT, the number of real roots.

16

Exercise 4

- Test your program with the following values for the coefficients:
  - a = 1.23456789
  - b = 2.4691356
  - c = 1.23456789
  - The appropriate response should be 1 root (note that the discriminant is 0 when a = c = $\frac{1}{2}$ b, try with a=1.3, b=2.6, c=1.3)
  - But it is possible (and probable) that your program do not provide you with the good response
    - The solution does not provide the good response.
  - Can you explain why?

17