THE UNIVERSITY OF WESTERN AUSTRALIA

Lecture 8
Multi-way Decisions

Objectives of this Lecture

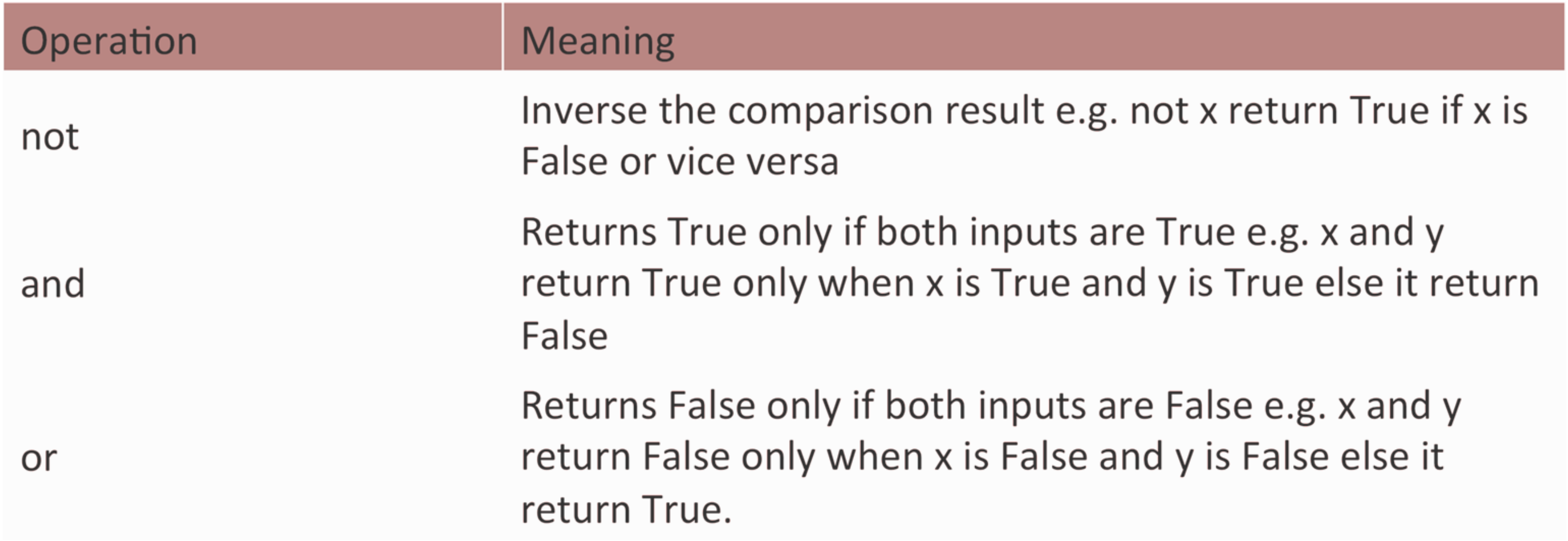
- A little revision
- To understand the conditional (decision) statement
  - *Nested if --- elif*

Revision: comparison operators

Python	Mathematics	Meaning
<	<	Less than
<=	≤	Less than or equal to
==	=	Equal to
>=	≥	Greater than or equal to
>	>	Greater than
!=	≠	Not equal to

Note ==

Revision: logical/boolean operators



Operation	Meaning
not	Inverse the comparison result e.g. not x return True if x is False or vice versa
and	Returns True only if both inputs are True e.g. x and y return True only when x is True and y is True else it return False
or	Returns False only if both inputs are False e.g. x and y return False only when x is False and y is False else it return True.

Logical operators are used to combine comparisons

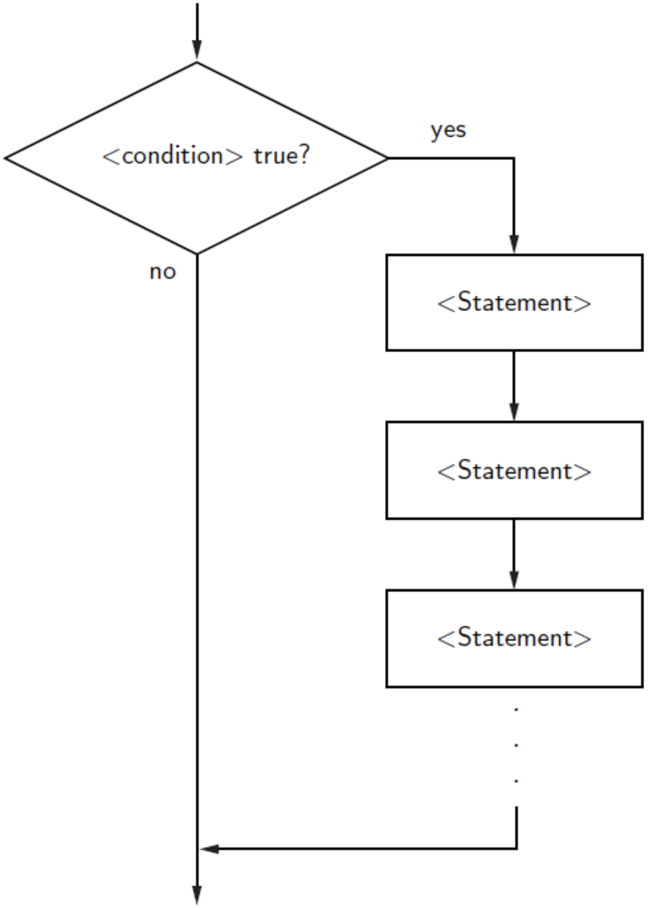
Revision: simple *if* statements

if <condition> :

<statements to execute if condition is True>

The condition is a Boolean expression

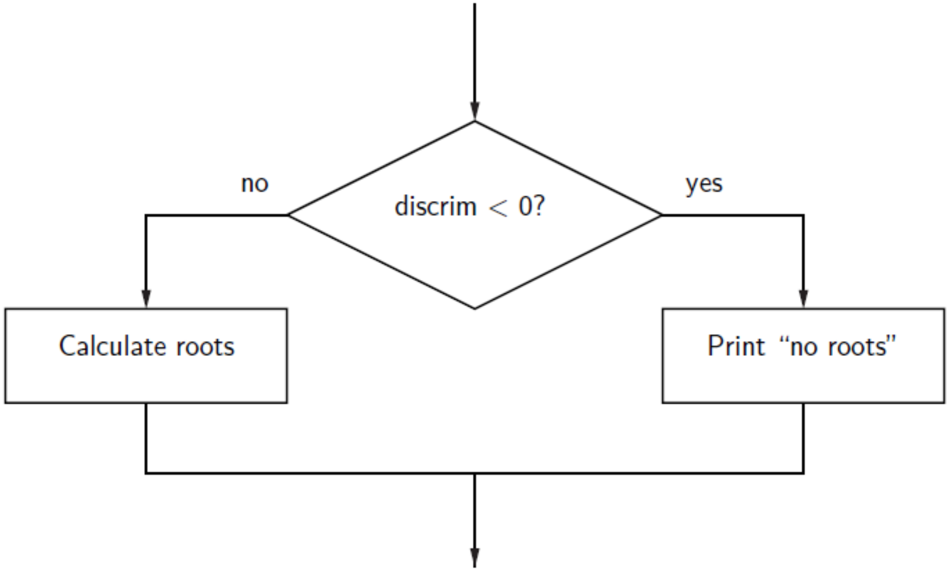
i.e., evaluates to values `True` or `False`



Revision: Two-Way Decisions

This is called an `if-else` statement:

```
if <condition>:
    <statements>
else:
    <statements>
```



Revision: Two-Way Decisions

```
# quadratic3.py
#    A program that computes the real roots of a quadratic equation.
#    Illustrates use of a two-way decision

import math

def main():
    print "This program finds the real solutions to a quadratic\n"
    a = float(input("Enter coefficient a: "))
    b = float(input("Enter coefficient b: "))
    c = float(input("Enter coefficient c: "))
    discrim = b * b - 4 * a * c
    if discrim < 0:
        print("\nThe equation has no real roots!")
    else:
        discRoot = (b * b - 4 * a * c)**(1/2)
        root1 = (-b + discRoot) / (2 * a)
        root2 = (-b - discRoot) / (2 * a)
        print ("\nThe solutions are:", root1, root2 )
```

Revision: Two-Way Decisions

```
This program finds the real solutions to a quadratic

Enter coefficient a: 1
Enter coefficient b: 1
Enter coefficient c: 2

The equation has no real roots!
>>>
This program finds the real solutions to a quadratic

Enter coefficient a: 2
Enter coefficient b: 5
Enter coefficient c: 2

The solutions are: -0.5 -2.0
```

Two-Way Decisions

The newest program is great, but it still has some quirks!

```
This program finds the real solutions to a quadratic


Enter coefficient a: 1

Enter coefficient b: 2

Enter coefficient c: 1


The solutions are: -1.0 -1.0
```

Program looks broken, when it isn't

Two-Way Decisions

- While correct, this program output might be confusing for some people.
  - *It looks like it has mistakenly printed the same number twice!*
- A single root occurs when the discriminant is exactly 0, and then the root is *–b/2a*.
- It looks like we need a three-way decision!

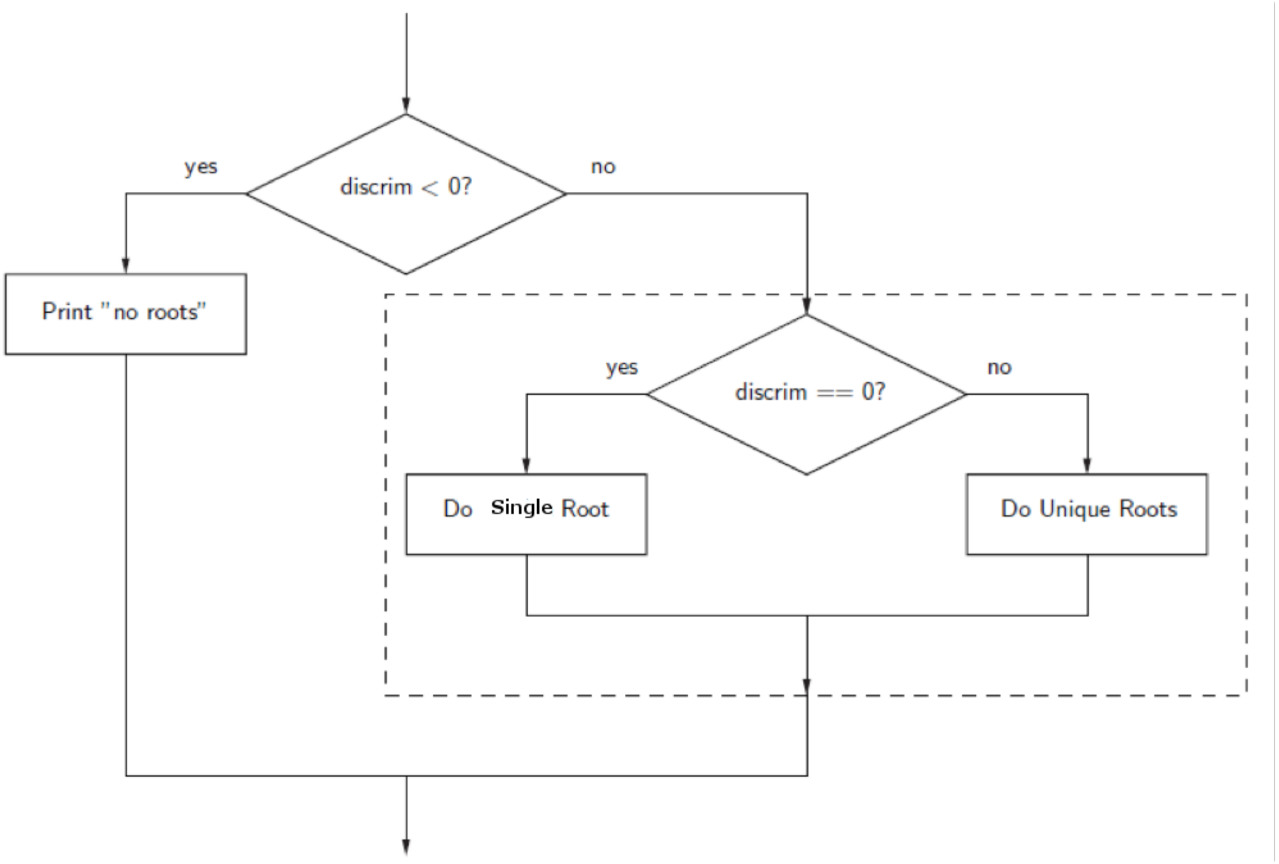
Two-Way Decisions

- Check the value of discrim
  when < 0: handle the case of no real roots
  when = 0: handle the case of a single root
  when > 0: handle the case of two distinct roots
- We can do this with two if-else statements, one inside the other.
- Putting one compound statement inside of another is called *nesting*.

Two-Way Decisions

```
if discrim < 0:
    print("Equation has no real roots")
else:
    if discrim == 0:
        root = -b / (2 * a)
        print("There is a single root at", root)
    else:
        # Do stuff for two roots
```

Nested Two-Way Decisions



Multi-Way Decisions

- Imagine if we needed to make a five-way decision using nesting. The if-else statements would be nested four levels deep!
- There is a construct in Python that achieves this, combining an `else` followed immediately by an `if` into a single `elif`.

Multi-Way Decisions

```
if <condition1>:
   <statements>
elif <condition2>:
   <case2 statements>
elif <condition3>:
   <case3 statements>
…
else:
   <default statements>
```

Multi-Way Decisions

- Python evaluates each condition in turn looking for the first one that evaluates to `True`. If a true condition is found, the statements indented under that condition are executed, and control passes to the next statement after the entire `if-elif-else`.
- If none are `True`, the statements under `else` are performed.
- The final `else` is optional. If there is no `else`, it's possible no indented block would be executed.

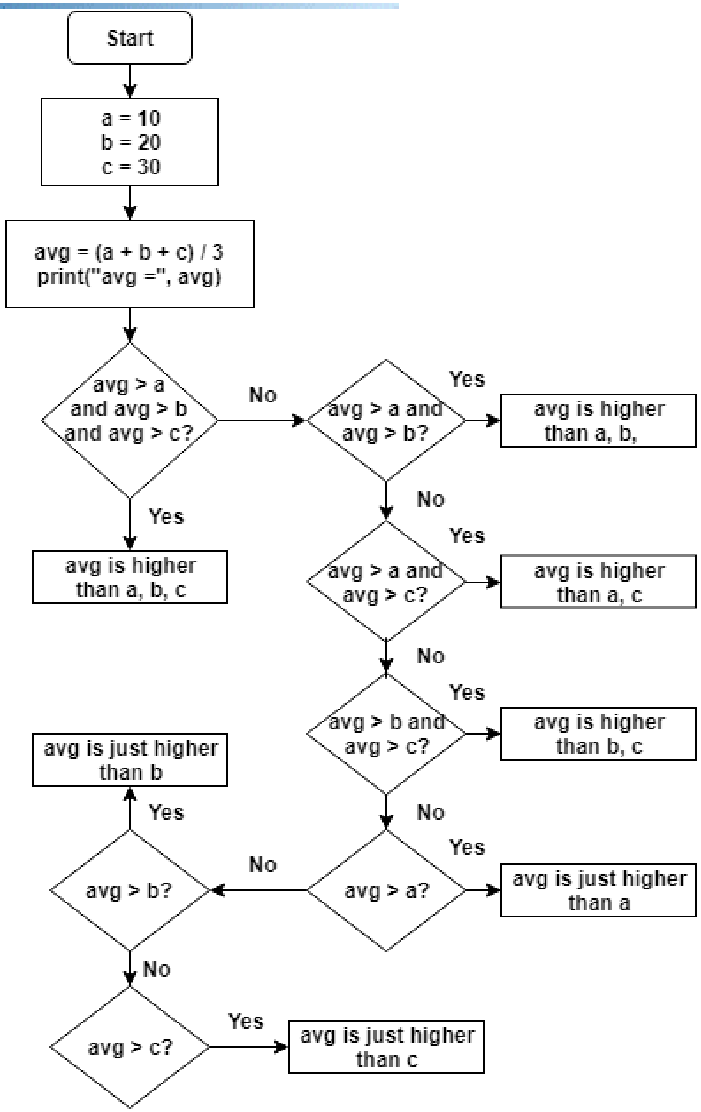
Three-Way Decisions

```
# quadratic4.py
import math
def main():
    print("This program finds the real solutions to a
   quadratic\n")

    a = float(input("Enter coefficient a: "))
    b = float(input("Enter coefficient b: "))
    c = float(input("Enter coefficient c: "))
    discrim = b * b - 4 * a * c
    if discrim < 0:
        print("\nThe equation has no real roots!")
    elif discrim == 0:
        root = -b / (2 * a)
        print("\nThere is a single root at", root)
    else:
        discRoot = (discrim)**(1/2)
        root1 = (-b + discRoot) / (2 * a)
        root2 = (-b - discRoot) / (2 * a)
        print("\nThe solutions are:", root1, root2 )
```

Nested Two-way Decisions

- What is the purpose of this algorithm?



Anti-bugging

- In the quadratic program we used decision structures to avoid taking the square root of a negative number, thus avoiding complex numbers which may introduce error later.
- This is true for many programs: decision structures are used to protect against rare but possible errors.
- Some authors describe this as anti-bugging; before processing some data have tests to ensure procedures will be safe.
- You are expected to do the same for your projects.

Lecture Summary

- We learned about multi-way decision making in computer program
- We learned about the use of `if-elif-else` decision statement
- We learned about *anti-bugging*