

CSI31 Lecture 9

Topics:

- 7.3 Multi-way Decisions
- 7.4 Exception Handling (part of the section)
- 7.5 Study in Design: Max of Three

HW# 7 (due date: Wednesday, October 22nd):

Write three programs that find the maximum of three numbers. The user will input the numbers from the keyboard. Use decision tree method, sequential processing and Python's built-in function *max*.

7. 3 Multi-way Decisions

One-way decisions *if*

```
if <condition>:  
    body
```

Two-way decisions *if-else*

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

Multi-way decisions *if-elif-else*:

```
if <condition1>:  
    <case 1 statements>  
elif <condition2>:  
    <case 2 statements>  
elif <condition3>:  
    <case 3 statements>  
  
elif <conditionn>:  
    <case n statements>  
else:  
    <default statements>
```

Examples (quadratic equation), pseudocodes:

With *if*:

```
if (discr == 0):  
    find one root  
  
if (discr < 0):  
    no real roots  
  
if (discr > 0):  
    find two roots
```

With *if-else*:

```
if (discr == 0):  
    find one root  
  
else:  
    if (discr < 0):  
        no real roots  
  
    else:  
        find two roots
```

With *if-elif-else*:

```
if (discr == 0):  
    find one root  
  
elif (discr < 0):  
    no real roots  
  
else:  
    find two roots
```

7.4 Exception Handling

Using the same example: solving a quadratic equation

we checked whether the radicand is less than zero **before** the call to sqrt function.

Sometimes the programs become too crowded with decisions to check for special cases that the main algorithm for handling the run-of-the-mill cases seems completely lost. Programming language designers have come up with mechanisms for *exception handling* that helps to solve this design problem.

Idea: we can write code that catches and deals with errors that arise when the program is running, rather than explicitly checking that each step in the algorithm was successful.

«Do these steps and if there is a problem, handle it this way»

syntax:

```
try:  
    <body>  
except <ErrorType>  
    <handler>
```

← What to do in case if something failed in <body>

Another program that solves quadratic equation:

```
def main():
    print "This program solves the quadratic equation. ..."

    try:
        import math
        a,b,c = input("Please, input the coefficients ...:")
        discrRoot = math.sqrt(b*b-4*a*c)
        root1=(-b+discrRoot)/(2*a)
        root2=(-b-discrRoot)/(2*a)
        print "The roots of the quadratic equation are:", root1,root2
    except ValueError:
        print "No real roots"

main()
```

Please, note that `ValueError` is the name of the error that arises when the program tries to extract a square root of a negative number

(type in the following in the interactive window:

```
>>> import math
```

```
>>> math.sqrt(-10)
```

see what's the error name)

See the more sophisticated program in `quadratic-s.py`

7.5 Study in Design: Max of Three

Let's write a program that finds the maximum of three numbers (a,b,c):
There are more than one way of finding the maximum:

1. Compare each to all
2. Decision tree
3. Sequential processing
4. Use already written by somebody function

1. Compare each to all

idea:

If $a \geq b$ and $a \geq c$ then a is maximum

If $b \geq a$ and $b \geq c$ then b is maximum

If $c \geq a$ and $c \geq b$ then c is maximum

```
...  
if a>=b and a>=c:  
    print a, 'is maximum of ', a, b, c  
elif b>=a and b>=c:  
    print b, 'is maximum of ', a, b, c  
else:  
    print c, 'is maximum of ', a, b, c
```

2. Decision tree

if $a \geq b$

 if $a \geq c$

 a is maximum

 else

 c is maximum

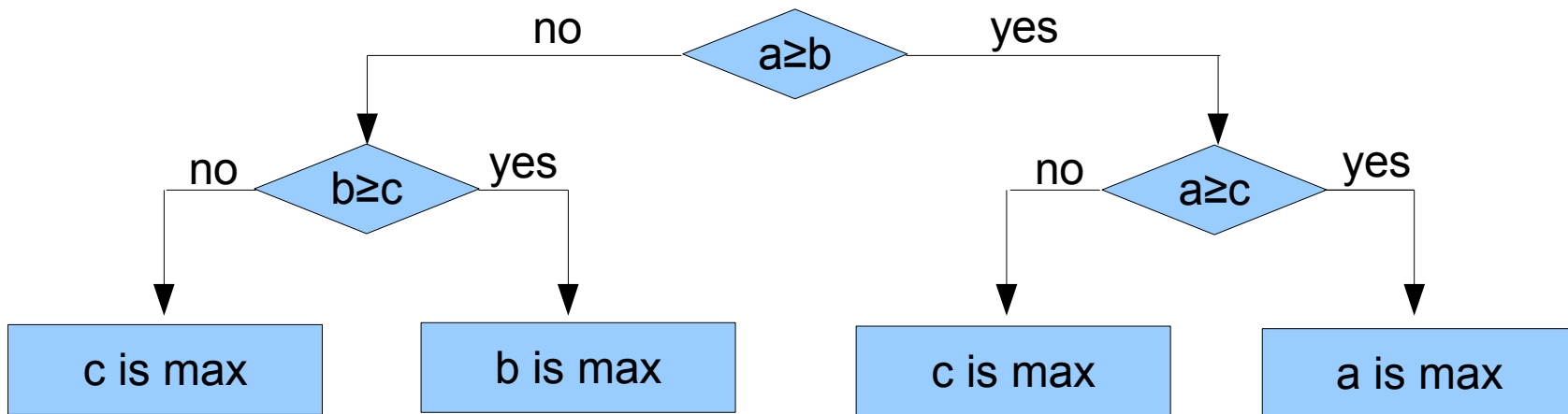
else

 if $b \geq c$

 b is maximum

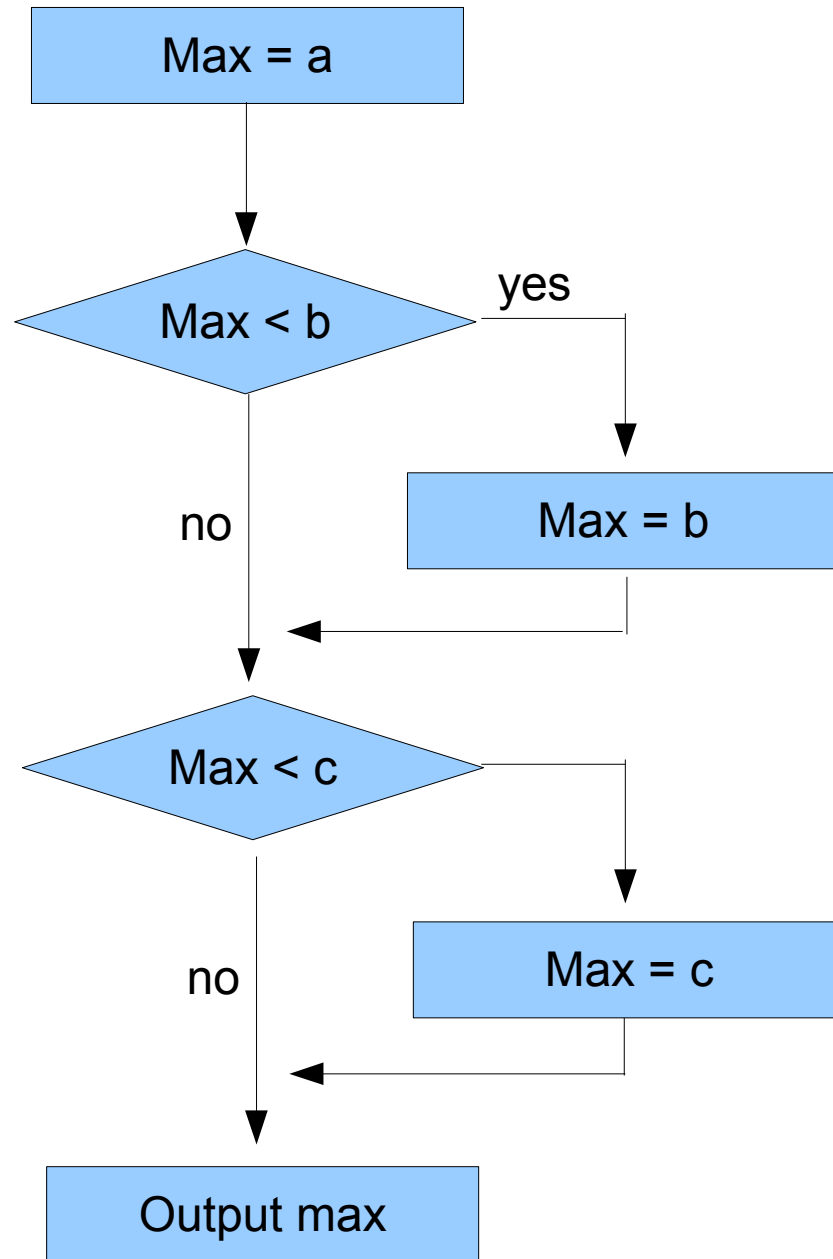
 else

 c is maximum



3. Sequential Processing

```
max=a
if max < b
    max=b
if max < c
    max = c
```



4. Use already written by somebody function

Python's function:

`max(a,b,c)`

See if we need to use *math* library