

# CSI31 Lecture 7

## Topics:

7.3 Multi-way Decisions

7.4 Exception Handling (part of the section)

7.5 Study in Design: Max of Three

## 7.3 Multi-way Decisions

### One-way decisions *if*

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

### Two-way decisions *if-else*

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

## 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>
```

## 7.3 Multi-way Decisions

### Example: Solving quadratic equations

$$ax^2+bx+c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Discriminant  $D = b^2 - 4ac$

if  $D = 0$ , there is only one solution

if  $D < 0$ , there are no real number solutions

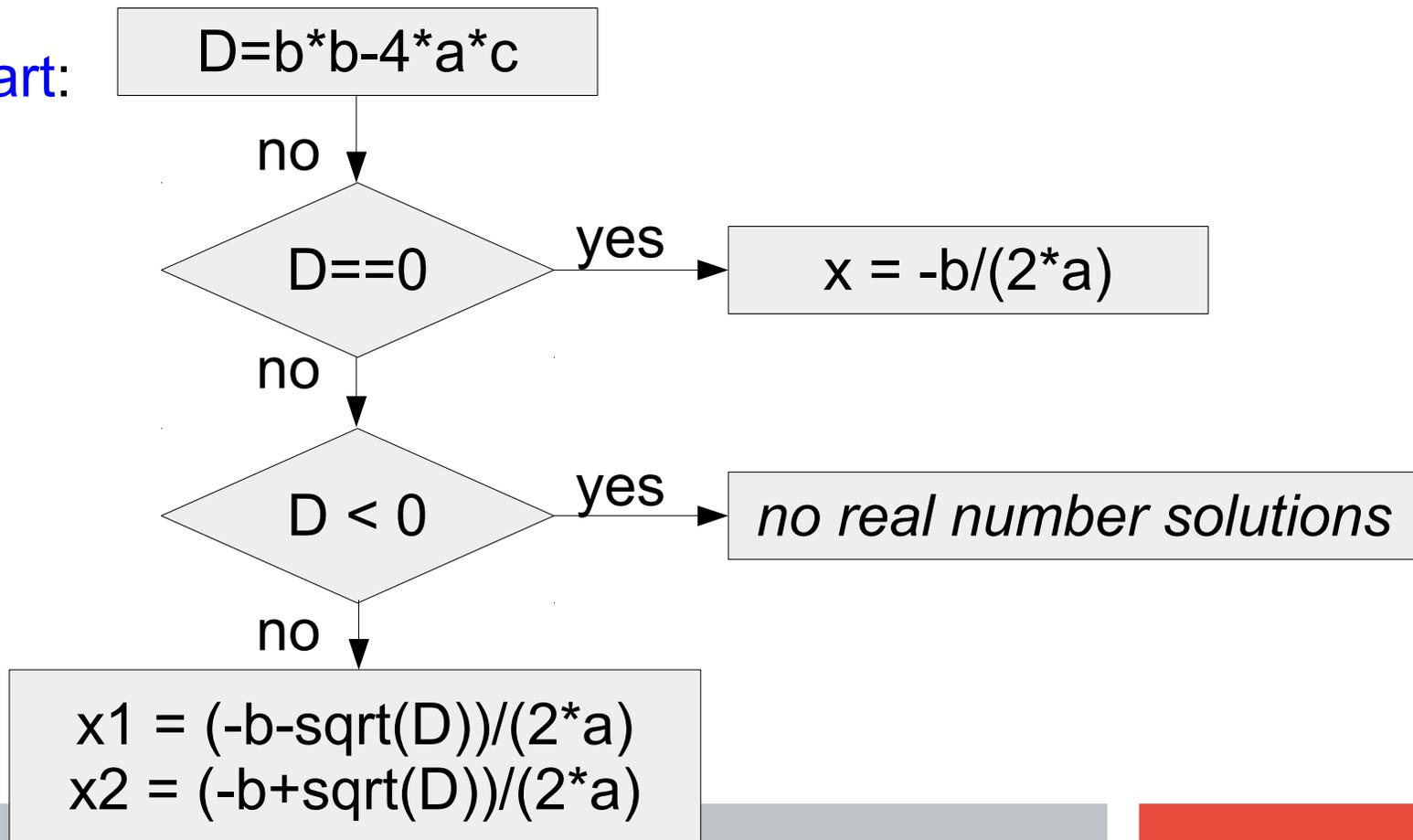
if  $D > 0$ , there are two solutions.

## 7. 3 Multi-way Decisions

### Example: Solving quadratic equations

$$ax^2+bx+c = 0$$

flowchart:



## 7.3 Multi-way Decisions

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
```

see programs: [quadratic-equation.py](#), [quadratic-equation\\_mod.py](#)

## 7.4 Exception Handling

Let's use 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.



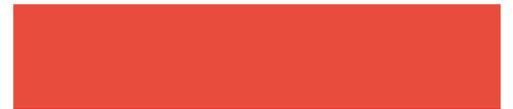
## 7.4 Exception Handling

syntax:

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

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

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



## 7.4 Exception Handling

Consider another program that solves quadratic equation:

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

    try:
        import math
        a = float(input("Enter coefficient a:"))
        b = float(input("Enter coefficient b:"))
        c = float(input("Enter coefficient c:"))
        discrRoot = math.sqrt(b*b-4*a*c)
        root1=(-b+discrRoot)/(2*a)
        root2=(-b-discrRoot)/(2*a)
        print("The roots are:", root1,root2)
    except ValueError:
        print("No real roots")
```

```
main()
```

## 7.4 Exception Handling

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 the following in the Python 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

## 7.5 Study in Design: Max of Three

### 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)

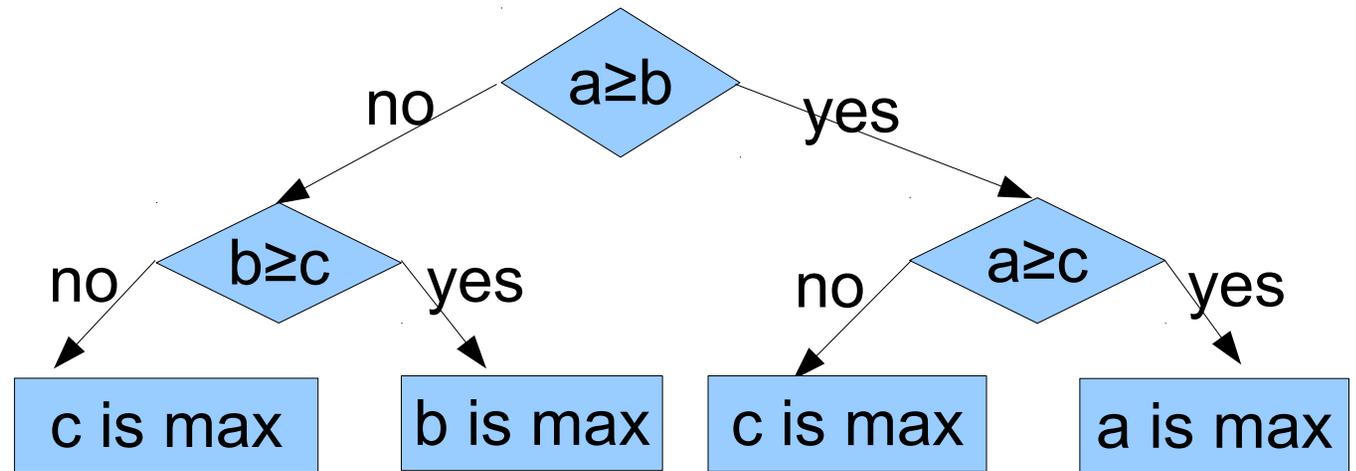
elif b>=a and b>=c:
    print(b, 'is maximum of ', a, b, c)

else:
    print(c, 'is maximum of ', a, b, c)
```

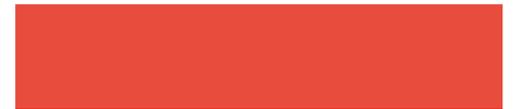
## 7.5 Study in Design: Max of Three

### 2. Decision tree

if  $a \geq b$   
  if  $a \geq c$   
    a is maximum  
  else  
    c is maximum



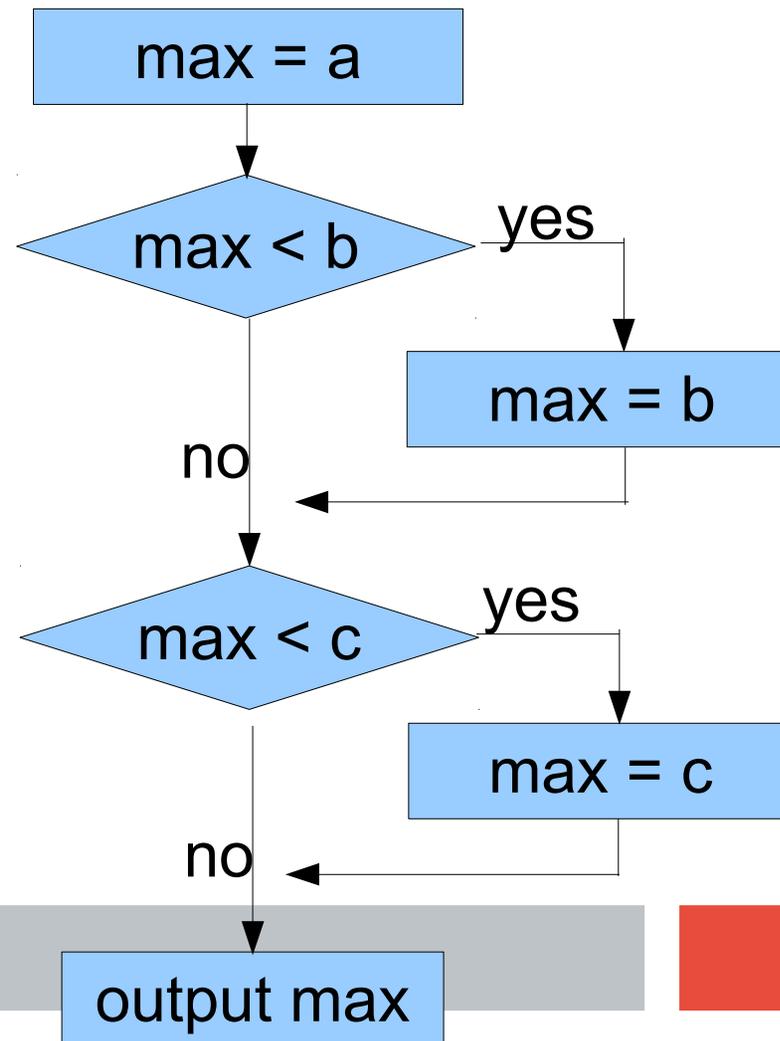
else  
  if  $b \geq c$   
    b is maximum  
  else  
    c is maximum



## 7.5 Study in Design: Max of Three

### 3. Sequential Processing

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



## 7.5 Study in Design: Max of Three

### 4. Use already written by somebody function

Python's function:

```
max(a, b, c)
```

check if we need to use *math* library or it is a *built-in method*.