Control Structure: Multiple Selections

01204111 Computers and Programming

Chalermsak Chatdokmaiprai
Department of Computer Engineering
Kasetsart University
Outline

• *Introduction to multiple selections*
• Nested Conditionals
• Chained Conditionals
• Programming examples
**Review: Basic Selections**

A single **if** statement selects whether or not a code block is to be executed.

A single **if-else** statement selects one of two statements (or blocks) to be executed.

There are only two possible paths of execution in both constructs.
Introduction

- **Multiple selection** selects one of **three or more** paths of execution.

**Example**

How many possible paths of execution?
How to do multiple selections in Python

Multiple selections

- Nested Conditionals
- Chained Conditionals
Outline

• Introduction to multiple selections

• Nested Conditionals

• Chained conditionals

• Programming examples
How *nested conditionals* are possible in Python

Each of these yellow boxes is actually a **single statement**.

```python
if x > y:
    m = x
    z = x+y
else:
    m = y
    z = x-y
```

**so each can be put here:**

```python
if condition:
    statement_1
    statement_2
else:
    statement_3
    statement_4
    statement_5
```

**or here:**

```python
pass
x = y+1
print('hi')
if x > y:
    m = x
```
Example in Python

When an `if` or `if-else` statement is put within another `if` or `if-else` statement, we call it a **nested conditional construct**.

```python
if x > 0:
    i = 2
    if y > 1:
        k = 2
    else:
        if z < 10:
            k = 5
        if y > 5:
            k = x+y
        else:
            k = x-y
```

In Python, indentation is very, very important!
Nested conditionals start as just a single statement

```
if x > 0:
    i = 2
else:
    if y > 1:
        k = 2
    else:
        if z < 10:
            k = 5
        else:
            if y > 5:
                k = x+y
            else:
                k = x-y
```

Recall that a code block follows the line that ends with a colon.
Flow-of-Control Example

# in Python
```python
if x > y:
    m = x
    if z > m:
        m = 1
    else:
        m = y
z = x - y
r = x + z - m
```

Path #1
```
x > y
m = x
z > m
m = 1
```

Path #2
```
x > y
m = y
z = x - y
```

Path #3
```
x > y
m = x
z > m
m = 1
```

T
```
x > y
m = x
f
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
f
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
f
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
f
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
f
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
f
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
f
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
f
z = x - y
```

F
```
x > y
m = y
z = x - y
```

T
```
x > y
m = x
z > m
m = 1
```

F
```
x > y
m = y
z = x - y
```

F
```
x > y
m = y
z = x - y
```
Programming Example on Nested Conditionals
Task: The maximum of three numbers

• Write a program that
  ➢ reads three numbers.
  ➢ computes and prints the maximum of the three.

Sample Run

Enter 1st number: 25.5
Enter 2nd number: 30
Enter 3rd number: 20.2
The max is 30

Sample Run

Enter 1st number: 0
Enter 2nd number: -10
Enter 3rd number: -7
The max is 0

Sample Run

Enter 1st number: 50
Enter 2nd number: 5
Enter 3rd number: 50
The max is 50
The maximum of three numbers - Ideas

❖ What is the maximum of 3 numeric values?

• **Answer:** *The value that is not less than the other two.*

• Therefore, to find the maximum is *to look for a value that is not less than the other two.*

• It’s OK if some of them are equal, or even all of them are equal.
The main routine:

- Reads three numbers.
- Computes the max by calling `max_of_three()`.
- Prints the max.

```python
# --- main --- #
x = float(input("Enter 1st number: "))
y = float(input("Enter 2nd number: "))
z = float(input("Enter 3rd number: "))
max = max_of_three(x,y,z)
print(f"The maximum number is {max}")
```
The function `max_of_three()` – Design

- Now it’s time to write the function `max_of_three()`.
- There are many ways to write it.
- We’ll show a few different ways so as to demonstrate the use of nested conditionals.
The function `max_of_three()` – Version 1

```python
def max_of_three(a, b, c):
    if a >= b and a >= c:  # check a
        return a
    else:  # a is not the max
        if b > c:
            return b
        else:
            return c
```

Algorithm

- **if (a is not less than the other two)**
  - a is the max
- **else**  # a is not the max
  - Compete b with c for the max

Efficiency:
2 or 3 comparisons depending on the inputs
The function `max_of_three()` – Version 2

```python
def max_of_three(a, b, c):
    if a > b:  # so a may be the max
        if a > c:
            return a
        else:
            return c
    else:    # so b may be the max
        if b > c:
            return b
        else:
            return c
```

Algorithm

1. If `a > b`, then `a` may be the max.
   - Compete `a` with `c` for the max.
2. Else, `b` may be the max.
   - Compete `b` with `c` for the max.

Efficiency:
exactly two comparisons in all cases
The function `max_of_three()` – Version 3

```python
def max_of_three(a, b, c):
    max = a
    if b > max:
        max = b
    if c > max:
        max = c;
    return max
```

**Algorithm**

- Let \( \text{max} \) be the value of \( a \)
- If \( b > \text{max} \) then
  - Let \( \text{max} \) be the value of \( b \)
- If \( c > \text{max} \) then
  - Let \( \text{max} \) be the value of \( c \)

This is actually a sequence of two if statements, not a nested if construct.

**Efficiency:**

Exactly two comparisons in all cases

This version can be easily extended to 4 or more numbers.

**How?**
The function **max_of_three()** – **Version 4**

- No **if-** or **if-else** statements used.
- No need to write the function **max_of_three()**.
- Throw away the main routine. **Hmm...?**
- **In fact, no need to write a program at all!**

**Amitta Buddh...???
Nammo Amitta Pythonic Buddha!

```python
>>> max(5, 6)
6
>>> max(5, 6, 4)
6
>>> max(5, 7, 10, 3)
10
>>> max(3, 70, 5, 8, 10, 15, 75, 8, 40)
75
>>> type(max)
<class 'builtin_function_or_method'>
```
Outline

• Introduction to multiple selections
• Nested conditionals
• Chained conditionals
• Programming examples
Chained Conditionals

• What is a chained conditional?

The use of an orderly sequence of \( k \) conditions (\( k \geq 2 \)) to select one of \( k+1 \) code blocks to execute.

❖ It is also informally called the **if-elseif-else** control structure.
Example

Use 3 conditions to select one of the four sets of planned activities

What are the planned activities on Monday, Tuesday, Wednesday, or Thursday?
Chained conditionals in Python implemented by nested conditionals

if cond₁:
    code_block₁
else:
    if cond₂:
        code_block₂
    else:
        if cond₃:
            code_block₃
        else:
            code_block₄

Note that this whole box is actually a single Python statement.

Flow of execution
Chained conditionals in Python implemented by if-elif-else statements

`else:` followed by `if` can be replaced by Python keyword `elif`

```python
if cond_1:
    code_block_1
elif cond_2:
    code_block_2
elif cond_3:
    code_block_3
else:
    code_block_4
```

Then you must re-indent them to become an if-elif-else statement

Then, rearrange the flowchart accordingly
The flow charts of both implementations show that

Both work exactly the same.

Therefore, these two Python constructs work exactly the same too.

```python
if cond_1:
    code_block_1
else:
    if cond_2:
        code_block_2
    else:
        if cond_3:
            code_block_3
        else:
            code_block_4
```

```python
if cond_1:
    code_block_1
elif cond_2:
    code_block_2
elif cond_3:
    code_block_3
else:
    code_block_4
```
**Example:** Check how an integer is divided by 5

- Write a function `divfive()` to check how an integer is divided by 5:

```python
>>> divfive(50)
50 is divisible by 5
>>> divfive(54)
54 is not divisible by 5
the remainder is 4
>>> divfive(53)
53 is not divisible by 5
the remainder is 3
>>> divfive(52)
52 is not divisible by 5
the remainder is 2
>>> divfive(51)
51 is not divisible by 5
the remainder is 1
```
def divfive(d):  # version 1
    rem = d % 5
    if rem == 1:
        print(d, 'is not divisible by 5')
        print('the remainder is 1')
    elif rem == 2:
        print(d, 'is not divisible by 5')
        print('the remainder is 2')
    elif rem == 3:
        print(d, 'is not divisible by 5')
        print('the remainder is 3')
    elif rem == 4:
        print(d, 'is not divisible by 5')
        print('the remainder is 4')
    else:
        print(d, 'is divisible by 5')

This version is to show that you can have as many elif-clauses as you need.
def divfive(d):    # version 2
    rem = d % 5
    if rem == 0:
        print(d, 'is divisible by 5')
    elif rem == 1:
        print(d, 'is not divisible by 5')
        print('the remainder is 1')
    elif rem == 2:
        print(d, 'is not divisible by 5')
        print('the remainder is 2')
    elif rem == 3:
        print(d, 'is not divisible by 5')
        print('the remainder is 3')
    elif rem == 4:
        print(d, 'is not divisible by 5')
        print('the remainder is 4')

This version is to show that you can have no else-clause at all if you don't need it.
def divfive(d): # version 3
    rem = d % 5
    if rem == 0:
        print(d, 'is divisible by 5')
    else:
        print(d, 'is not divisible by 5')
    print('the remainder is', rem)

This version is to show that you can have no `elif`-clauses at all if you don't need them.

This becomes an ordinary `if-else` statement.

You should convince yourself that all these three versions produce exactly the same result.
More Programming Example on Chained Conditionals
Write a function `bmi_and_status()` that

- receives `weight` (in kg) and `height` (in meters) as parameters
- computes the body-mass index (BMI) and returns the BMI and weight status.
BMI and Weight Status - Idea

• Given the weight (in kilograms) and the height (in meters) of a person, the Body-Mass Index (BMI) of the person can be computed by the formula:

\[
\text{BMI} = \frac{\text{weight}}{\text{(height)} \times \text{(height)}}
\]

• The Weight Status of a person is categorized by the BMI as follows:

<table>
<thead>
<tr>
<th>BMI</th>
<th>Weight Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMI &lt; 18.5</td>
<td>Underweight</td>
</tr>
<tr>
<td>18.5 ≤ BMI &lt; 25.0</td>
<td>Normal</td>
</tr>
<tr>
<td>25.0 ≤ BMI &lt; 30.0</td>
<td>Overweight</td>
</tr>
<tr>
<td>BMI ≥ 30.0</td>
<td>Obese</td>
</tr>
</tbody>
</table>
BMI and Weight Status – Algorithm

<table>
<thead>
<tr>
<th>BMI</th>
<th>Weight Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMI &lt; 18.5</td>
<td>Underweight</td>
</tr>
<tr>
<td>18.5 ≤ BMI &lt; 25.0</td>
<td>Normal</td>
</tr>
<tr>
<td>25.0 ≤ BMI &lt; 30.0</td>
<td>Overweight</td>
</tr>
<tr>
<td>BMI ≥ 30.0</td>
<td>Obese</td>
</tr>
</tbody>
</table>

Here, it's certain that BMI ≥ 18.5
Here, it's certain that BMI ≥ 25.0
Here, it's certain that BMI ≥ 30.0

Compute BMI

BMI < 18.5
BMI < 25.0
BMI < 30.0

wstatus = "underweight"

wstatus = "normal"

wstatus = "overweight"

wstatus = "obese"
def bmi_and_status(weight, height):
    bmi = weight/(height*height)
    if bmi < 18.5:
        wstatus = "underweight"
    elif bmi < 25.0:
        wstatus = "normal"
    elif bmi < 30.0:
        wstatus = "overweight"
    else:
        wstatus = "obese"
    return bmi, wstatus
### BMI and Weight Status — another equivalent algorithm

<table>
<thead>
<tr>
<th>BMI</th>
<th>Weight Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMI &lt; 18.5</td>
<td>Underweight</td>
</tr>
<tr>
<td>18.5 ≤ BMI &lt; 25.0</td>
<td>Normal</td>
</tr>
<tr>
<td>25.0 ≤ BMI &lt; 30.0</td>
<td>Overweight</td>
</tr>
<tr>
<td>BMI ≥ 30.0</td>
<td>Obese</td>
</tr>
</tbody>
</table>

#### Compute BMI

- **BMI >= 30.0**
  - wstatus = "obese"
- **BMI >= 25.0**
  - wstatus = "overweight"
- **BMI >= 18.5**
  - wstatus = "normal"
- **BMI < 18.5**
  - wstatus = "underweight"
def bmi_and_status(weight, height):
    bmi = weight/(height*height)

    if bmi >= 30.0:
        wstatus = "obese"
    elif bmi >= 25.0:
        wstatus = "overweight"
    elif bmi >= 18.5:
        wstatus = "normal"
    else:
        wstatus = "underweight"

    return bmi, wstatus
Next: Write a main routine to test it

```python
def bmi_and_status(weight, height): # version 1
    bmi = weight/(height*height)

    if bmi < 18.5:
        wstatus = "underweight"
    elif bmi < 25.0:
        wstatus = "normal"
    elif bmi < 30.0:
        wstatus = "overweight"
    else:
        wstatus = "obese"

    return bmi, wstatus

# ---- main routine ---- #
weight = float(input("Enter your weight (in kilograms): "))
height = float(input("Enter your height (in meters): "))
bmi, status = bmi_and_status(weight, height)
print(f"BMI is {bmi:.2f}, weight status: {status}")
```
Test the program, thoroughly

Enter your weight (in kilograms): 70
Enter your height (in meters): 2
BMI is 17.50, weight status: underweight

Enter your weight (in kilograms): 80
Enter your height (in meters): 1.8
BMI is 24.69, weight status: normal

Enter your weight (in kilograms): 90
Enter your height (in meters): 1.8
BMI is 27.78, weight status: overweight

Enter your weight (in kilograms): 100
Enter your height (in meters): 1.8
BMI is 30.86, weight status: obese

Enter your weight (in kilograms): 74
Enter your height (in meters): 2
BMI is 18.50, weight status: normal

Enter your weight (in kilograms): 100
Enter your height (in meters): 2
BMI is 25.00, weight status: overweight

Enter your weight (in kilograms): 120
Enter your height (in meters): 2
BMI is 30.00, weight status: obese

Also try some inputs that hit all the three boundary cases 18.5, 25.0, 30.0

Try input values that yield all possible outputs
The End
Conclusion

• A **basic selection** control structure uses a single **if-** or **if-else** statement to select one of two paths of execution.

• A **multiple selection** control structure selects one of three or more paths of execution.

• To do a multiple selection in Python, we may use **nested conditionals** or **chained conditionals**.

• We've got a **nested conditional** when we put one or more **if** or **if-else** statements in a code block within another **if** or **if-else statement**. This naturally gives rise to many different paths of execution.

• A **chained conditional** is the use of an orderly sequence of k conditions, $k \geq 2$, to select one of $k+1$ code blocks to execute. In Python, a chained conditional can be conveniently implemented by an **if-elif-else** statement.
References

• **if-elif-else** statement in Python:
  ◦ [https://docs.python.org/3/reference/compound_stmts.html#the-if-statement](https://docs.python.org/3/reference/compound_stmts.html#the-if-statement)
  ◦ [https://docs.python.org/3/tutorial/controlflow.html#if-statements](https://docs.python.org/3/tutorial/controlflow.html#if-statements)

• Good tutorials for multiple selections:
  ◦ [http://interactivepython.org/runestone/static/thinkcspy/Selection/Nestedconditionals.html](http://interactivepython.org/runestone/static/thinkcspy/Selection/Nestedconditionals.html)
  ◦ [http://interactivepython.org/runestone/static/thinkcspy/Selection/Chainedconditionals.html](http://interactivepython.org/runestone/static/thinkcspy/Selection/Chainedconditionals.html)
  ◦ [https://www.programiz.com/python-programming/if-elif-else](https://www.programiz.com/python-programming/if-elif-else)
Major Revision History

• August 2016 – Chalermsak Chatdokmaiprai
  ◦ originally created for C#

• July 31, 2017 – Chalermsak Chatdokmaiprai
  ◦ adapted and enhanced for Python

Constructive comments or error reports on this set of slides would be welcome and highly appreciated. Please contact Chalermsak.c@ku.ac.th