CMPT 120
Intro to CS & Programming I

WEEK 7 (Feb. 24-28)

— *Jérémie O. Lumbroso* —

Lecture 18:
`if` statement clarifications and `while` loops (continued)

http://www.sfu.ca/~jlumbros/Courses/CMPT120/

Some common mistakes made with `if` statements

CLARIFICATIONS ON IF STATEMENTS

`if` statement: clarification 1

- Recall that several conditions can be tested for using the `elif` for "else if …"
- This is different from using several `if` one after the other

Is there a difference?

```
age = int(raw_input("Age? "))
print "you may drink",
if age < 12:
  print "water or milk"
elif 12 <= age < 18:
  print "coffee"
else:
  print "alcohol"
```

```
age = int(raw_input("Age? "))
print "you may drink",
if age < 12:
  print "water or milk"
if 12 <= age < 18:
  print "coffee"
else:
  print "alcohol"
```

A No difference

B There is a difference: left program is wrong when `age < 11`

C There is a difference: right program is wrong when `age < 11`

D Both programs are wrong
(who drinks water?? alcohol should be breastfed)

There is a difference!

```
age = int(raw_input("Age? "))
print "you may drink",
if age < 12:
  print "water or milk"
elif 12 <= age < 18:
  print "coffee"
else:
  print "alcohol"
```

```
>>> ======= RESTART =======
>>>
Age? 11
you may drink water or milk
```

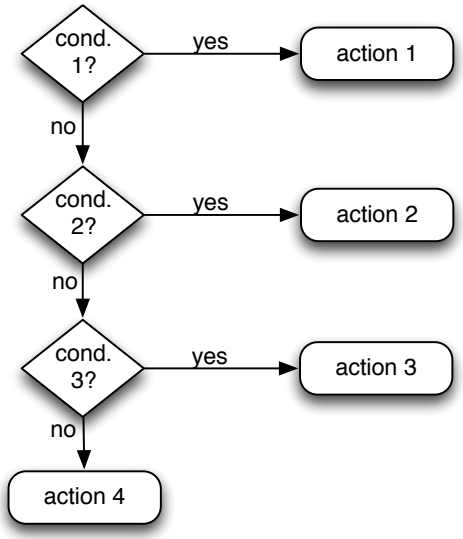
```
age = int(raw_input("Age? "))
print "you may drink",
if age < 12:
  print "water or milk"
if 12 <= age < 18:
  print "coffee"
else:
  print "alcohol"
```

```
>>> ======= RESTART =======
>>>
Age? 11
you may drink water or milk
alcohol
```

Comparison

if-elif-elif-else

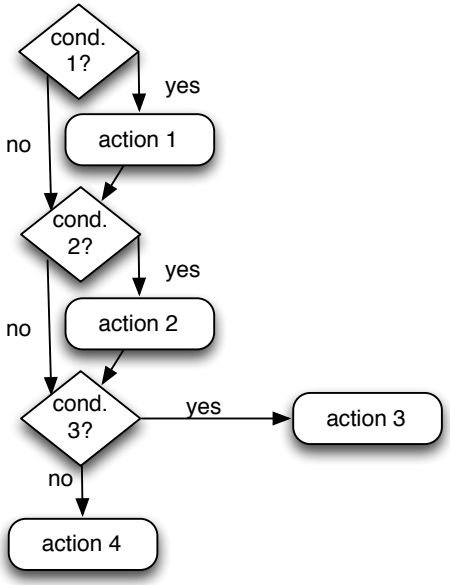
```
if <cond 1>:
  <action 1>
elif <cond 2>:
  <action 2>
elif <cond 3>:
  <action 3>
else:
  <action 4>
```



- Only one action is executed
- **Conditions tried in order**

if-if-if-else

```
if <cond 1>:
  <action 1>
if <cond 2>:
  <action 2>
if <cond 3>:
  <action 3>
else:
  <action 4>
```



- With if/if/…, possibly as many actions as if blocks

`if` statement: clarification 2

```
def remove_parentheses(s):
  new_s = ""
  inside_parentheses = False
  for ch in s:
    if ch == "(":
      inside_parentheses = True
    elif inside_parentheses == False:
      new_s = new_s + ch
    elif ch == ")":
      inside_parentheses = False
  return new_s
```

(Actual solution given by a student)

- `inside_parentheses` is a flag to **remember** if we are inside a group of parentheses
  - set to `True` when see `(`
  - set to `False` when see `)`
- when not inside a group of parentheses, then add the character `ch` normally to the string `new_s`
- when inside a group of parentheses don't add the character

`if` statement: clarification 2

```
def remove_parentheses(s):
  new_s = ""
  inside_parentheses = False
  for ch in s:
    if ch == "(":
      inside_parentheses = True
    elif inside_parentheses == False:
      new_s = new_s + ch
    elif ch == ")":
      inside_parentheses = False
  return new_s
```

`s = ")hello("` should return `"hello"`; does it?

- **A** No, it returns `")("`
- **B** No, it returns `")hello"`
- **C** No, it returns `"hello("`
- **D** Yes!

The `elif ch == ")"` branch is not executed for the `")"` because at that point in the execution, `inside_parentheses == False`, and the second condition bypasses the third condition.

Order Matters

```
def remove_parentheses(s):
  new_s = ""
  inside_parentheses = False
  for ch in s:
    if ch == "(":
      inside_parentheses = True
    elif inside_parentheses == False:
      new_s = new_s + ch
    elif ch == ")":
      inside_parentheses = False
  return new_s
```

Fix!

```
def remove_parentheses(s):
  new_s = ""
  inside_parentheses = False
  for ch in s:
    if ch == "(":
      inside_parentheses = True
    elif ch == ")":
      inside_parentheses = False
    elif inside_parentheses == False:
      new_s = new_s + ch
  return new_s
```

- **The order of conditions matters!**
- Simple fix: swap the two last `elif` blocks so that the more **general/inclusive** condition is last

More examples of `while` loops, and their difference with `for` loops

WHILE LOOPS (CONT'D)

while loops syntax

```
while <condition>:
    <actions>
```

- Every iteration, the loop executes `<actions>`
- The loop goes on as long as `<condition>` is verified
- The `<actions>` block **must** contain something to make the condition evolve, or else there is a risk a of **loop being infinite**

Which are infinite?

```
i = 0
while i < 4:
 print i
 i = i + 1
```

A Stops

B Infinite

```
i = 1
while i < 9:
 print i
```

A Stops

B Infinite

```
i = 4
while i > 5:
 print i
 i = i + 1
```

A Stops

B Infinite

```
i = 10
while i > 2:
 print i
 i = i + 1
```

A Stops

B Infinite

```
i = 1
while i < 9:
 print i
 # i = i + 1
```

A Stops

B Infinite

```
i = 4
while i < 7:
 print i
 i = i - 1
```

A Stops

B Infinite

Incremental while loops are infinite if

- there is no increment (2 & 5)
- the increment won't make condition change (4 & 6)

What about this one?

```
while 2*i < 4:
 print i
 i = i + 1
```

A Stops

B Other

Unlike the for loop, the increment variable must be defined, or else `NameError: name 'i' is not defined`

Incremental while loops

for loop

```
for i in range(5):
    print i

print "immediately after for"
print " i is: ", i
```

- A 3
- B 4
- C 5
- D Error

while loop

```
i = 0
while i < 5:
    print i
    i = i + 1
print "immediately after while"
print " i is: ", i
```

- A 3
- B 4
- C 5
- D Error

Changing index in a `for` loop?

- What happens when we change the index of the increment variable in a for loop?

```
for i in range(10):
  i = i - 1
  print i
```

A. Infinite loop because we keep decreasing `i` when it changes

B. Will not change the execution, but will print different values

C. There will be an error because Python does not like it

D. Why are you asking me questions on stuff I don't know? You really suck, man! This better not be graded, or else…

Changing index in a `for` loop?

- What if we are iterating over a string?

```
s = "hello"
for ch in s:
  s = s.upper()
print s
```

A It will print `hello`

B It will print `HELLO`

- Changing the variable in a for loop does not affect the execution of the for loop
- Changing the variable in a for loop does not modify the object being iterated over (in this case, the string `s`)

Pacing and Understanding

How well did you understand today?

- **A** Too easy, this lecture is way below my abilities
- **B** Everything went at a good pace, and I am fine
- **C** Too fast, but I will catch up on my own
- **D** Too fast, and I need you to slow down
- **E** I really do not think I can handle this