§3.1-3.8: Selection (if) and Repetition (while)

- *Quiz ch2 today*

18 Sep 2006
CMPT14x
Dr. Sean Ho
Trinity Western University

TRINITY
WESTERN
UNIVERSITY

Review of §2.7-2.10

- Expressions, operators, operands
  - Binary arithmetic: + - * / % // **
  - Comparison: == < > <= => != <> is, is not
  - Boolean: and or not (shortcut semantics)
- Type conversions
- Precedence rules
- Formatted output
  - %d, %f, %s

Quiz ch2: 10 minutes, 20 pts

- Name the five software control/flow abstractions
- Evaluate the following Python expressions:
  - **3.0 >= 1 and 3.0 <= 10**
  - **True and (3 <> 5.7)**
  - **not False or (12 % 0)**
  - **3 + 32 // 5.0**
- Show the output of this Python code:
  - **print “I have %04d %s.” % (23.7, “apples”)**
- Assume that the variable numApples has integer type. Write a line of pseudocode that would work in a dynamically typed language like Python but would fail in a statically typed language like C.

Quiz ch2: answers #1-2

- Name the five software control/flow abstractions [5]
  - Sequence (;)
  - Selection (if)
  - Repetition (loops: while, for)
  - Composition (subroutine/function)
  - Parallelism
- Evaluate the following Python expressions: [8]

3.0 >= 1 and 3.0 <= 10	True
True and (3 <> 5.7)	True
not False or (12 % 0)	True
3 + 32 // 5.0	9.0

Quiz ch2: answers #3-4

- Show the output of this Python code: [4]
  - print “I have %04d %s.” % (23.7, “apples”)
  - **I have 0023 apples.**
- Assume that the variable numApples has integer type. Write a line of pseudocode that would work in a dynamically typed language like Python but would fail in a statically typed language like C. [3]
  - **numApples = 5.0**
  - **numApples = “Hello World!”**
  - **numApples = False**
  - **etc.**

What's on for today (§3.1-3.8)

- Selection: if, if..else.., if..elif..else
- Loops: while
- Sentinel variables
- Loop counters
- Using mathematical closed forms instead of loops

Chapter 3: Program Structure

- Five basic program structure/flow abstractions:
  - Sequence (newline)
  - Selection (if … elif … else)
  - Repetition/loops (while, for)
  - Composition (subroutines)
  - Parallelism
- This chapter mostly covers the first three program structure abstractions

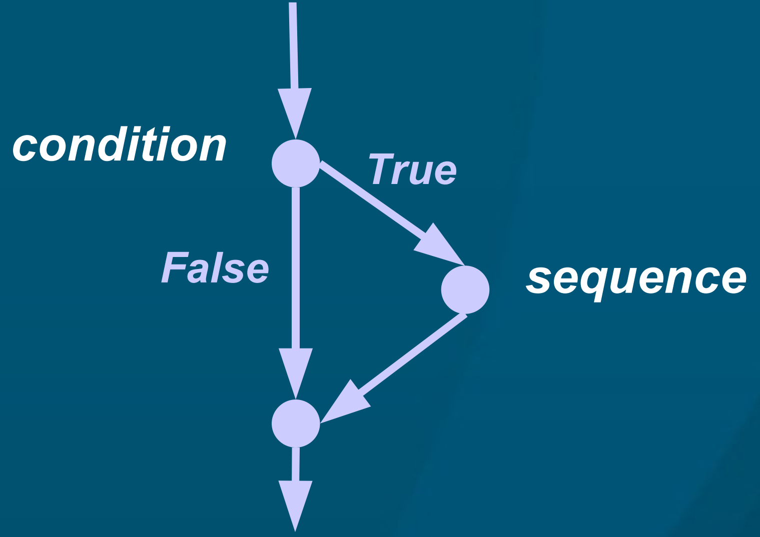
Statement sequences

- A sequence of statements is executed in order:
  - Successive statements are not executed until the preceding statement is completed

```
print “Running really_slow_function() ...”
really_slow_function()
print “done!”
```

- Separate statements are on separate lines
  - Whitespace and newlines matter in Python
  - In most other languages, semicolon (;) separates statements, and newlines don't matter

Simple selection: if



if *condition* :

statement sequence

- Indentation (tab) indicates what's part of the statement sequence
- Condition is a Boolean expression evaluating to either True or False
- Conditional execution: if condition evaluates to False, then the statement sequence is skipped over and not executed

Example using if

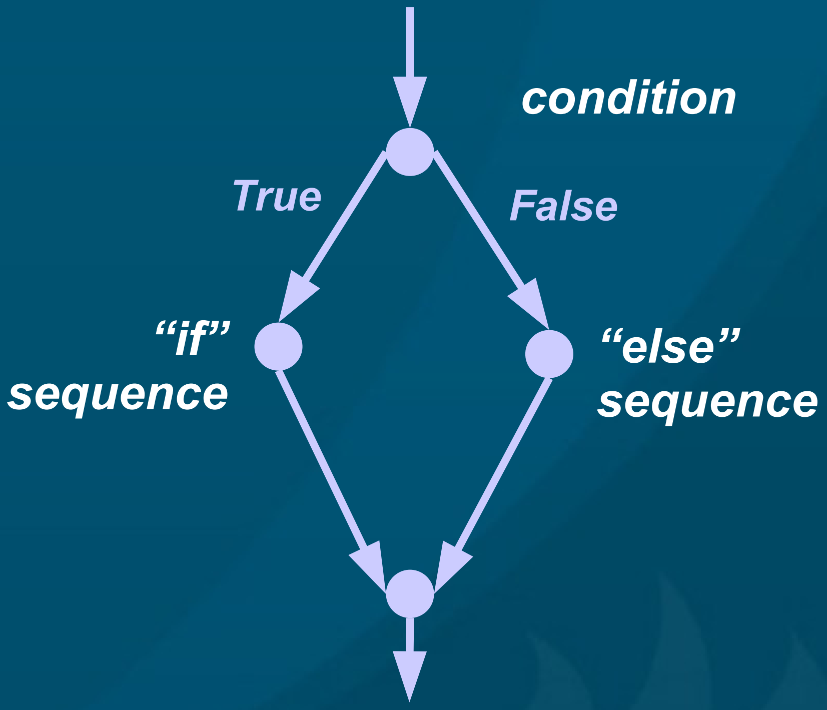
```
if numApples > 12:
        print “Okay, that's waay too many apples!”
print “Let's eat some apples!”
```

- Observe indentation (it matters in Python!)
- Parentheses () not needed around the condition
  - But if the condition is complex, parentheses may be useful to clarify precedence:
    - if (numApples > 5) and (numApples < 12)

Branching: if ... else ...

```
if condition :
        statement sequence
else :
        statement sequence
```

- Only one of the two statement sequences is executed



Example using if ... else ...

```
if numFriends > 0:
        applesPerFriend = numApples / numFriends
else:
        print “Awww, you need some friends!”
```

- Would the division work if numFriends == 0?
- Will this code generate an error if numFriends == 0?

Branching: if … elif … else …

if *condition* :

statement sequence

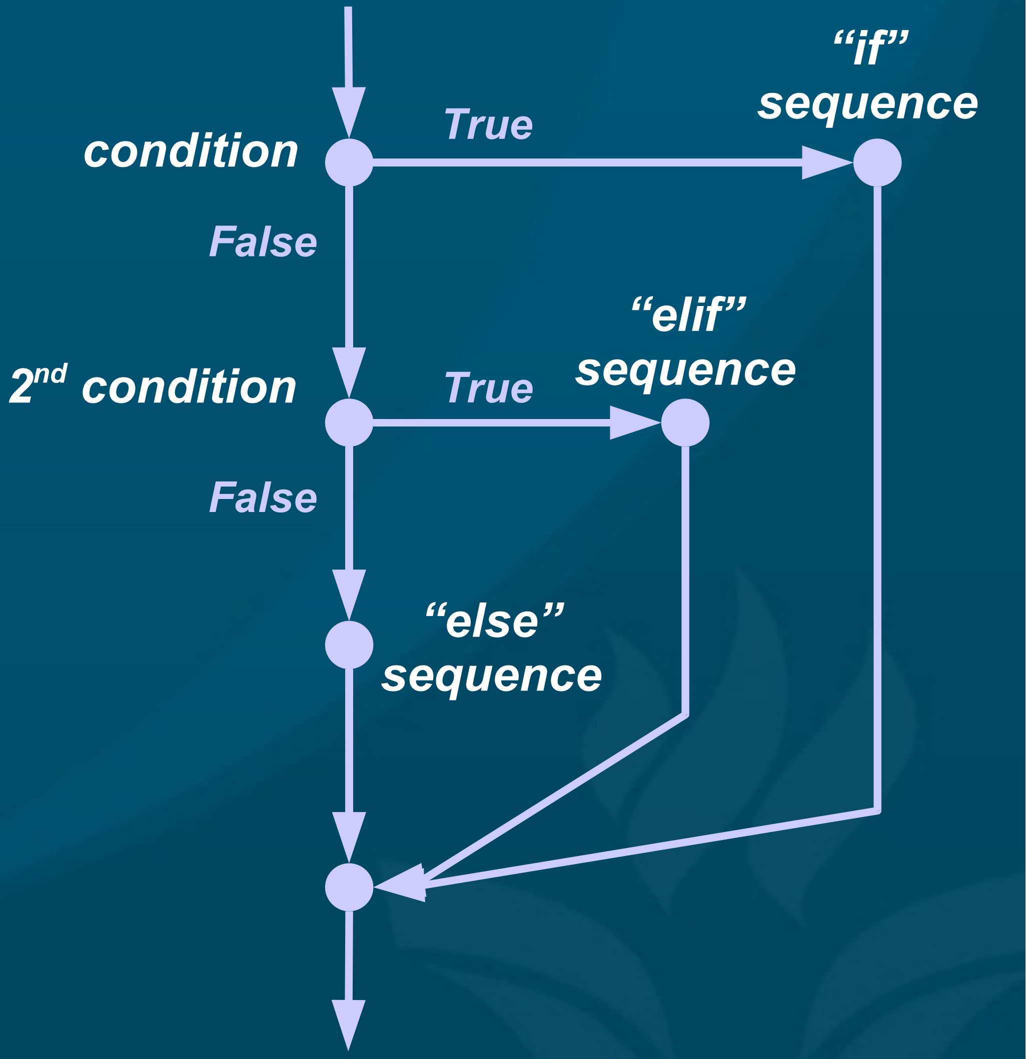
elif 2nd condition :

statement sequence

else :

statement sequence

- Only one of the statement sequences is executed



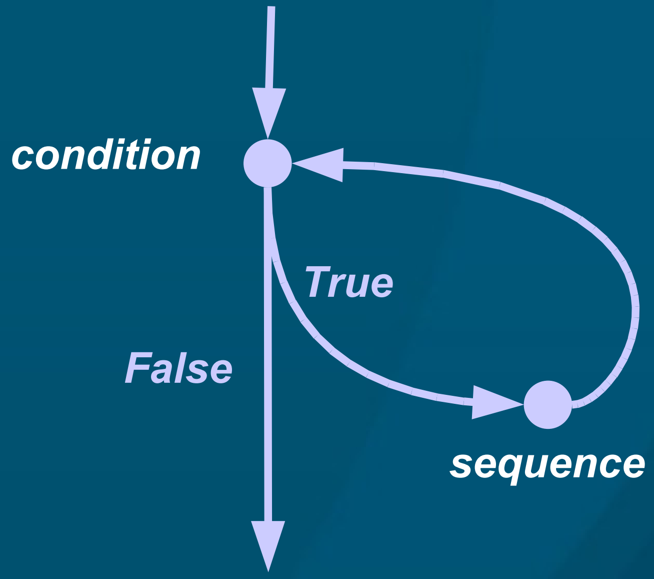
Example using if ... elif ... else ...

```
if numFriends <= 0:
        print “Awww, you need some friends!”
elif numFriends > 30:
        print “Wow, that's a lot of friends!”
else:
        applesPerFriend = numApples / numFriends
```

while loops

while *condition :*

statement sequence



- As with “if”, *condition* is a Boolean expression:
  - It is evaluated once before entering the loop,
  - And re-evaluated each time through the loop:
  - Top-of-loop testing
- *Statement sequence* is run only if *condition* evaluates to True

Sentinel variables

- A sentinel variable controls whether a loop continues: the loop only exits when the sentinel variable has a certain value

```
answer = 0
while answer != 4:
    answer = input(“Math quiz: 2 + 2 = ”)
```

  - Sentinel variable is answer
  - Sentinel value is 4

Counting loops

- A common form of loop uses a counter:

```
counter = 1
while counter <= max:
    sum = sum + counter
    counter = counter + 1
```

- What if we need to prematurely exit this loop?

```
counter = 1
while counter <= max:
    if need_to_exit_early():
        counter = max + 1
    ...
```

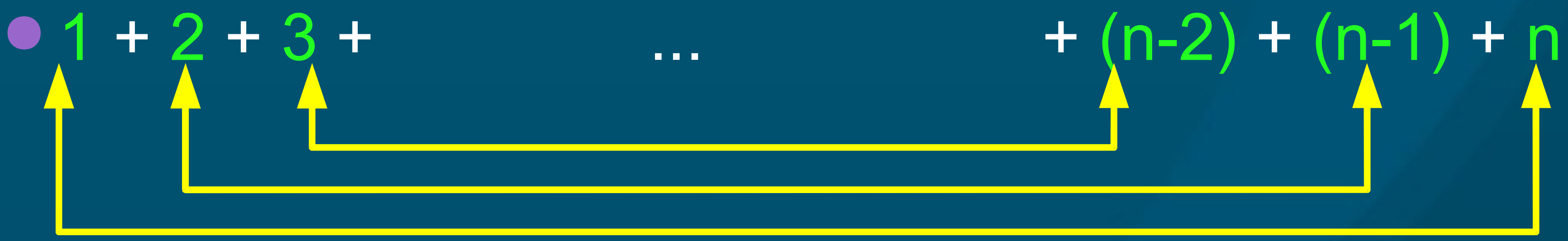
Closed forms instead of loops

- Sometimes with a bit of thought we can replace a loop with a single mathematical equation
  - “Work smarter, not harder”
- Example: Add the first *n* integers >0

```
sum = 0
counter = 1
while counter <= n:
    sum = sum + counter
    counter = counter + 1
print “Sum is %d.” % sum
```

Closed form solution

- But observe the pattern:



  - 1 + 2 + 3 + ... + (n-2) + (n-1) + n

- Each pair makes n+1; there are n/2 pairs:
- Closed form solution:

  sum = n * (n+1) / 2

  - (If n is type int, does the / cause problems?)

A few misc nifty tricks

- Absolute value built-in function: abs(-5.0) --> 5.0
- Increment/decrement, etc:
  - count += 1 # same as count = count + 1
  - numApples *= 2 # nA = nA * 2
  - No builtin "++" operator as in C++/Java
- Turn strings into all-caps:
  - import string
  - string.capitalize("Hello") # "HELLO"

Review of today (§3.1-3.8)

- Selection: if, if..else.., if..elif..else
- Loops: while
- Sentinel variables
- Loop counters
- Using mathematical closed forms instead of loops
- abs(), +=, string.capitalize()