Lecture 3
Decisions

Recap last lecture

- Modules
- OO
- Iteration
- Functions
- Scope
- Questions?

Morning Routine

On weekdays:

Alarm clock goes off
Hit snooze
Alarm clock goes off
Hit snooze
Alarm clock goes off
Get out of bed
Do yoga
Take a shower
Get dressed
Have breakfast
Cycle to the uni

In the weekend:

Get out of bed
Do yoga
Have breakfast
Take a shower
Get dressed

Morning Routine

```
def  weekRoutine() :
        Alarm clock goes off
        Hit snooze
        Alarm clock goes off
        Hit snooze
        Alarm clock goes off
        Get out of bed
        Do yoga
        Take a shower
        Get dressed
        Have breakfast
        Cycle to the uni
```

```
def  weekendRoutine() :
        Get out of bed
        Do yoga
        Have breakfast
        Take a shower
        Get dressed
```

Decisions

Conditional Execution

```
if BOOLEAN EXPRESSION:
    doSomething()
else:
    doSomethingElse()
```

Conditional Execution

```
if weekday:
    weekRoutine()
else:
    weekendRoutine()
```

George Boole

Boolean expressions

True / False

Values that a boolean variable can take

Comparison operators:

==, !=, <, >, <=, >=

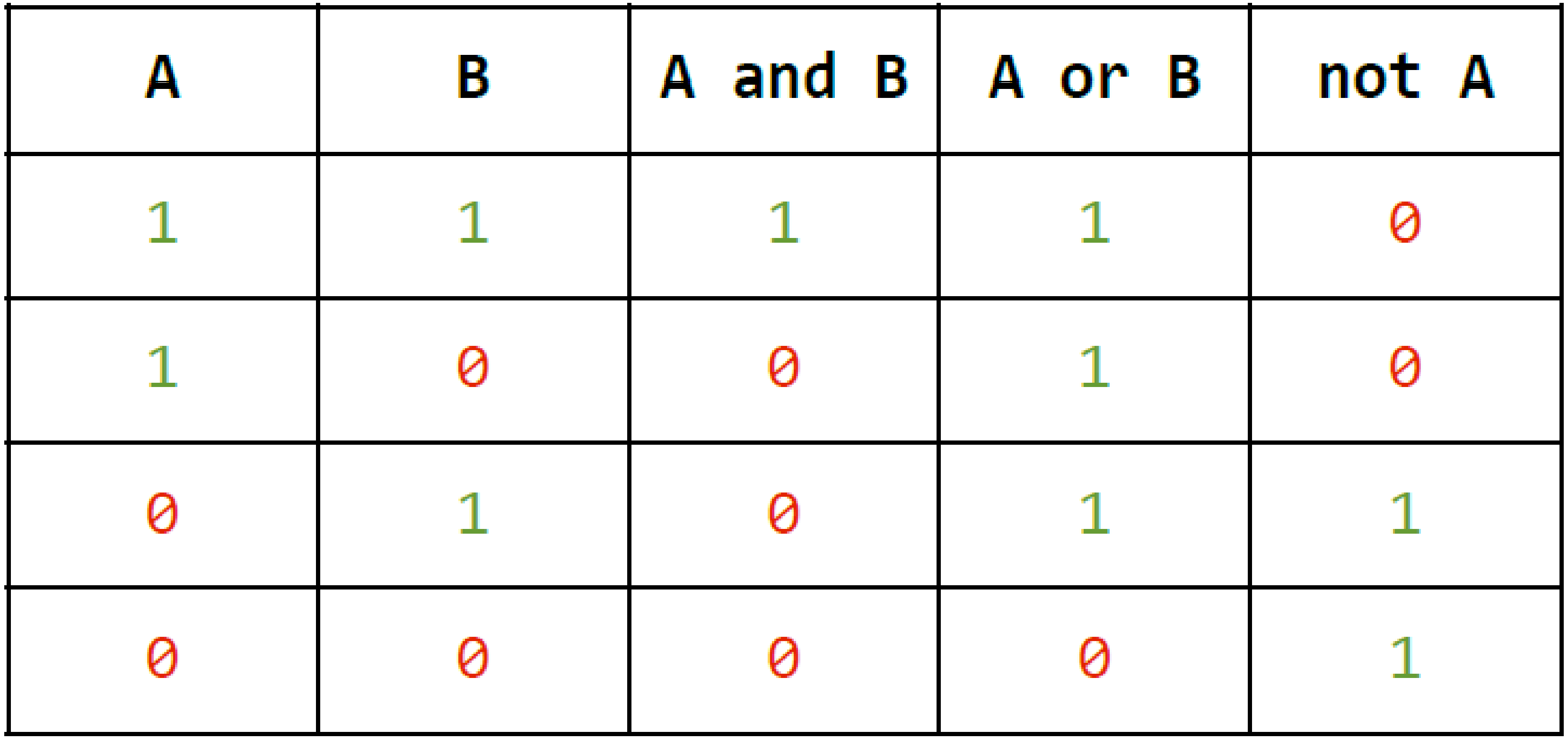
Logical operators:

and, or, not

Logical operators

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Logical operators



A	B	A and B	A or B	not A
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

Logical opposite

a < 2?

a >= 2

Logical opposite

a and b ?
not (a and b)
not a or not b

Logical opposite

A	B	A and B	not (A and B)	not A	not B	not A or not B
True	True	True	False	False	False	False
True	False	False	True	False	True	True
False	True	False	True	True	False	True
False	False	False	True	True	True	True

Nested conditionals

```
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

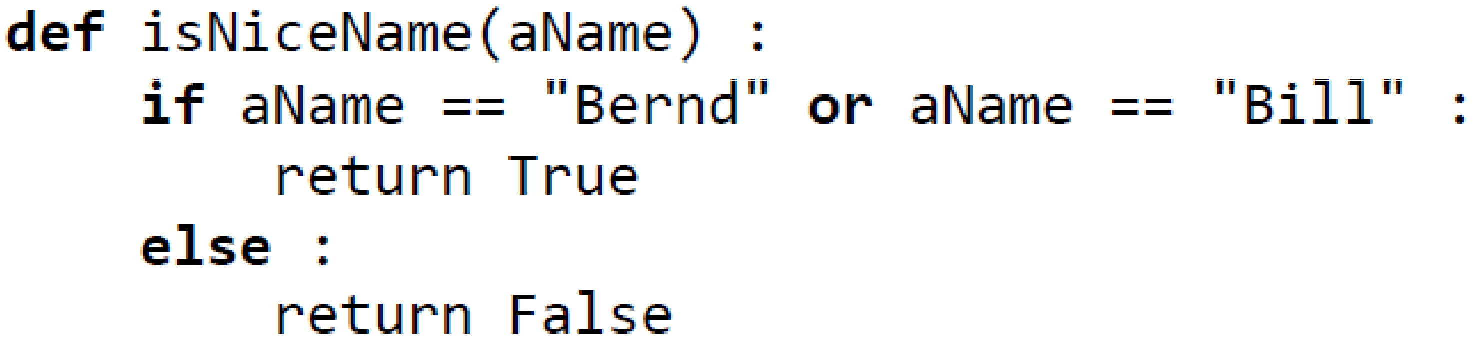
Why is this not ok?

```
if x < y:
    print("x is less than y")
if x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```

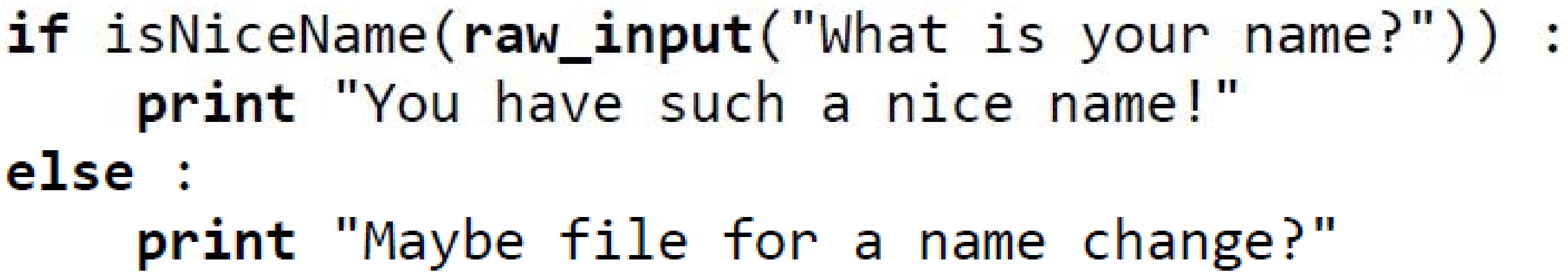
Chained conditionals

```
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```

Boolean functions



```
def isNiceName(aName) :
    if aName == "Bernd" or aName == "Bill" :
        return True
    else :
        return False
```



```
if isNiceName(raw_input("What is your name?")) :
    print "You have such a nice name!"
else :
    print "Maybe file for a name change?"
```

Iteration revisited

On weekdays:

- Alarm clock goes off
- Hit snooze
- Alarm clock goes off
- Hit snooze
- Alarm clock goes off
- Get out of bed
- Do yoga
- Take a shower
- Get dressed
- Have breakfast
- Cycle to the uni

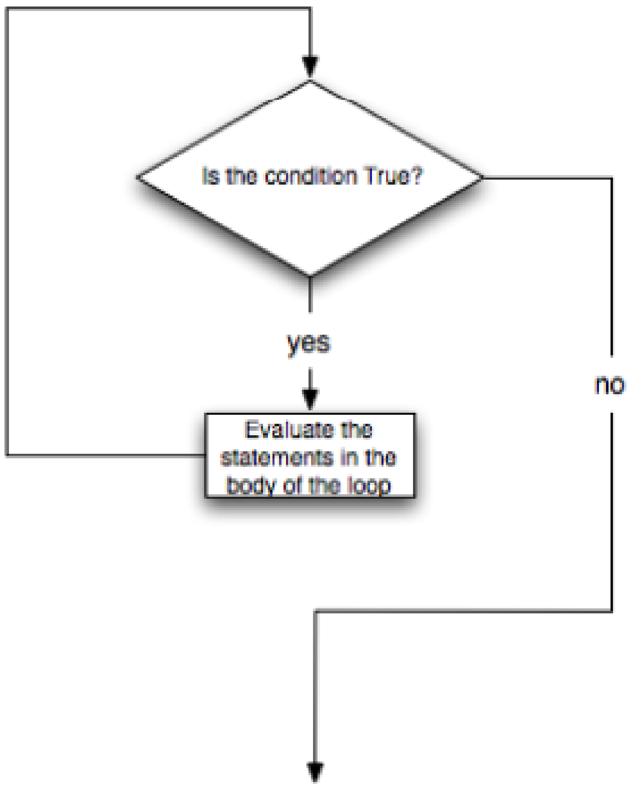
Using `for`

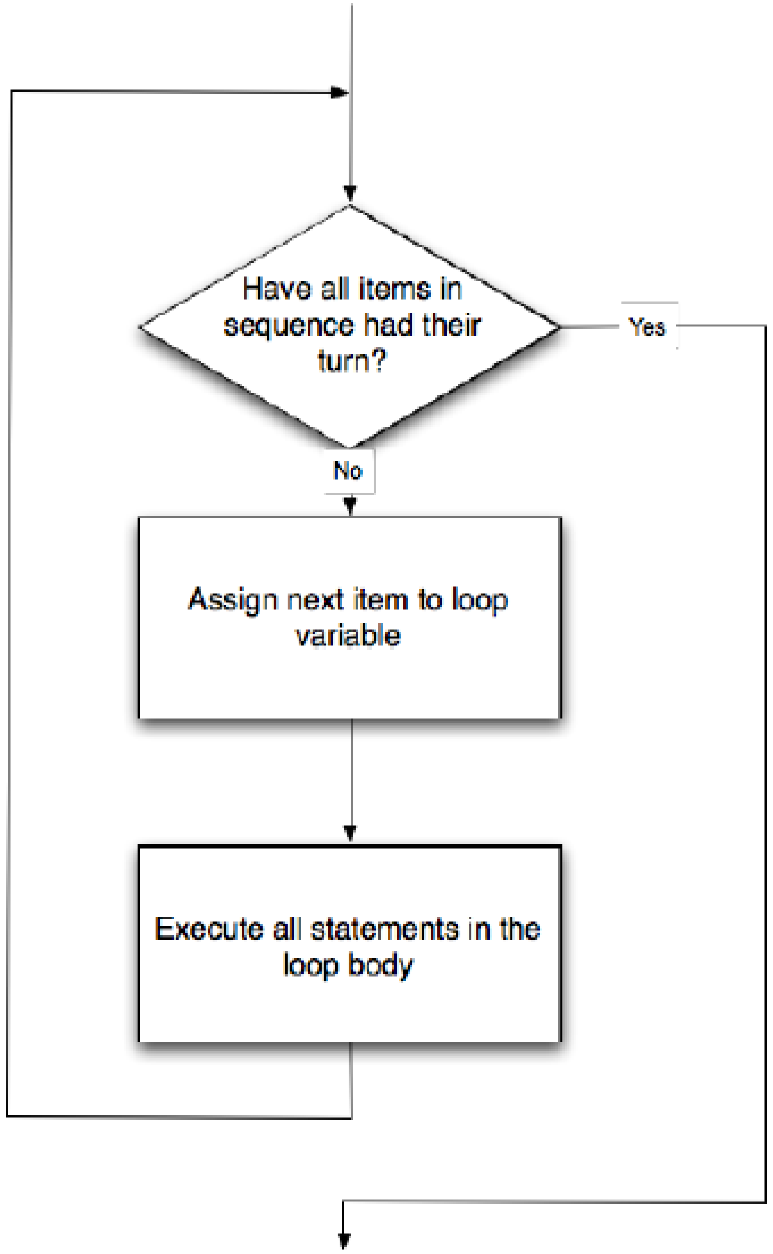
```
On weekdays:
    alarm clock goes off
    for i in range(2):
        hit snooze
        alarm clock goes off
    Get out of bed
    Do yoga
    Take a shower
    Get dressed
    Have breakfast
    Cycle to the uni
```

Using `while`

```
On weekdays:
    alarm clock goes off
    while I_am_tired and time_to_work > 40:
        hit snooze
        alarm clock goes off
    Get out of bed
    Do yoga
    Take a shower
    Get dressed
    Have breakfast
    Cycle to the uni
```

while


Is the condition True?
yes
no
Evaluate the statements in the body of the loop


Have all items in sequence had their turn?
Yes
No
Assign next item to loop variable
Execute all statements in the loop body

Why do we need `while`?

- More work, but more powerful
- Sometimes you don’t know how many iterations you need in advance

Accumulator pattern

Increment a counter or running total on each loop iteration

Accumulator Pattern

```
total = 0
for  x in range(10):
     total = total + x
```

Now with `while`

```
total = 0
for  x in range(10):
     total = total + x
```

Now with `while`

```
total = 0
i = 0
while i < 10:
    total = total + i
    i = i + 1
```

BLACK JACK

Black Jack

- Bank draws one card
- You draw cards to get as close to 21 as possible. After each card you get the choice to pass or draw one more
- You lose ('bust') if you go over 21
- Bank then draws cards until 17 or higher
- If bank goes over 21, it busts and you win
- If you have a card total higher than the bank, you win.
- Otherwise, you lose

Recap

- Boolean expressions (`==, and/or/not`)
- Conditional execution (`if, else, elif`)
- Accumulator pattern (running total)
- Iteration using `while`
- Black Jack

Next Lecture

Wednesday May, 11th: 10:45- 12:30

(me with a jetlag)

This week’s homework

- Read chapters Decisions and Iteration Revisited (until randomly walking turtles)
- Solve these problems
  - Decisions: 1-3, 7-9
  - Approximation of Pi with simulation (under Labs)
  - If you feel cool, Iteration revisited: 1-6 (but you need to read up on Image Processing for that
- Bring these solutions to lab session (hard copy):
  - Decisions: 2, 8, approximation of pi