



# Conditionals and Booleans

CMPU 101 – Problem Solving and Abstraction

Peter Lemieszewski



# Boolean Values

- Are:
  - True or False
    - They can, in older programming languages, be represented by `{1, 0}`
    - Or even `{nonzero value, 0}`
    - i.e. they are relatively new data type
- They can result from evaluating expressions i.e. `5 > (2 + 2)`



# Combining Boolean Values

- To combine Boolean values, we can use and:
  - $\langle \text{expression 1} \rangle$  and  $\langle \text{expression 2} \rangle$
- To combine Boolean values we can use or:
  - $\langle \text{expression 1} \rangle$  or  $\langle \text{expression 2} \rangle$

## Short Circuiting (vb)

- Evaluation of an expression stops, or is “short-circuited” when:
  - For and: as soon as one of the expressions being combined evaluates to false. (why?)
  - For or: as soon as one of the expressions evaluates to true. (why?)



# Combining Boolean Values

- To combine Boolean values, we can use **and**:
  - `<expression 1>` and `<expression 2>`
- To combine Boolean values we can use **or**:
  - `<expression 1>` or `<expression 2>`

## Short Circuiting (vb)

- Evaluation of an expression stops, or is “short-circuited” when:
  - For **and**: as soon as one of the expressions being combined evaluates to false.  
(it’s pointless to proceed... if false, further evaluation can never become true)
  - For **or**: as soon as one of the expressions evaluates to true.  
(it’s pointless to proceed if true, further evaluation can never become false)
  - `What if there was no short circuiting?`

# Boolean Examples



```
Run Stop
>>> true and false
false
>>> true or false
true
>>> true and false and true and true and true
false
>>> false or false or false or false or true
true
>>> #which statements above employ short circuiting ?
```

- Answer: ?

# Boolean Examples



```
Run Stop
>>> true and false
false
>>> true or false
true
>>> true and false and true and true and true
false
>>> false or false or false or false or true
true
>>> #which statements above employ short circuiting ?
```

- Answer: true and false and true and true and true

# Boolean Examples



Run Stop

```
>>> (1 < 2) and (2 > 3)
false
>>> (1 <= 0) or ((1 == 1) and (2 > 3))
false
>>> #any short circuiting here?
```

- Answer: ?

# Boolean Examples



```
Run Stop
>>> (1 < 2) and (2 > 3)
false
>>> (1 <= 0) or ((1 == 1) and (2 > 3))
false
>>> #any short circuiting here?
```

- Answer: no, but how can we change the second expression such that:
  - there is short circuiting?



# We can change expression evaluation...

not! 


- To change an expression that evaluates to true to be false or vice versa,
- use not:

```
>>> not(1 == 0)
```

```
true
```

# Another example



```
▼  View Connect to Google Drive Run Stop
```

```
1 use context essentials2021
2 i1 = rectangle(10, 20, "solid", "red")
3 i2 = rectangle(20, 10, "solid", "blue")
4 image-width(i1) < image-width(i2)
5
```

```
true
>>>
```

# Introducing: If



- What is the result of this expression?

```
▼  View Connect to Google Drive
1 use context essentials2021
2 rect = rectangle(10, 20, "solid", "red")
3 ▼ if image-width(rect) < image-height(rect):
4   "portrait"
5 else:
6   "landscape"
7 end
8
```


If  
If you can keep your head when all about you  
Are losing theirs and blaming it on you;  
If you can trust yourself when all men doubt you,  
But make allowance for their doubting too;  
If you can wait and not be tired by waiting,  
Or, being lied about, don't deal in lies,  
Or, being hated, don't give way to hating,  
And yet don't look too good, nor talk too wise;

Poem by Rudyard Kipling. Excerpt from  
<https://en.wikipedia.org/wiki/If%E2%80%94>

# Introducing: If



- What is the result of this expression?

```
▼  View Connect to Google Drive
1 use context essentials2021
2 rect = rectangle(10, 20, "solid", "red")
3 ▼ if image-width(rect) < image-height(rect):
4   "portrait"
5 else:
6   "landscape"
7 end
8
```

```
"portrait"
>>>
```

## If

If you can keep your head when all about you  
Are losing theirs and blaming it on you;  
If you can trust yourself when all men doubt you,  
But make allowance for their doubting too;  
If you can wait and not be tired by waiting,  
Or, being lied about, don't deal in lies,  
Or, being hated, don't give way to hating,  
And yet don't look too good, nor talk too wise;

Poem by Rudyard Kipling. Excerpt from  
<https://en.wikipedia.org/wiki/If%E2%80%94>

# If/else expressions



To form an **if** expression:

```
if <expression>:  
    <expression>  
else:  
    <expression>  
end
```

*True–false  
question*

*True (“then”) answer*

*False (“else”) answer*

# If/else expressions



- Evaluation steps for *if* expressions
  1. If the **question** expression is not a value, evaluate it, and replace with value.
  2. If the **question** expression is true, replace entire if expression with **true answer** expression.
  3. If the question is false, replace entire if expression with **false answer** expression.
  4. If the question is a value other than true or false, → **error**

To form an **if** expression:

```
if <expression>:  
  <expression>  
else:  
  <expression>  
end
```

*True–false  
question*

*True (“then”) answer*

*False (“else”) answer*

# Portrait/Landscape Mode (Revisited)



▼  View Connect to Google Drive

```
1 use context essentials2021
2 rect = rectangle(10, 20, "solid", "red")
3 ▼ if image-width(rect) < image-height(rect):
4   "portrait"
5 else:
6   "landscape"
7 end
8
```

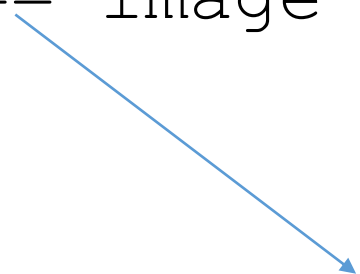
- Let's convert this code snippet to classify rectangles. We need a third classification though...
- Portrait
- Landscape
- square

# Portrait/Landscape/Square



- We need to “nest” or embed if statements for this third category

```
rect = rectangle(10, 20, "solid", "red")
if image-width(rect) < image-height(rect):
    "portrait"
else if image-width(rect) == image-height(rect):
    "square"
else:
    "landscape"
end
```



**Note: The “==” operator means “is equivalent”. It is not the assignment operator “=”. Don’t use assignment operators in if/else statements!**



# Function: Portrait/Landscape/Square



```
rect = rectangle(10, 20, "solid", "red") #used for testing in where:
fun image-type(img :: Image) -> String:
  doc: "Classify an image as portrait, square, or landscape"
  if image-width(img) < image-height(img):
    "portrait"
  else if image-width(img) == image-height(img):
    "square"
  else:
    "landscape"
  end
  where:
    image-type(rect) is "portrait"
    image-type(rectangle(10, 10, "solid", "blue")) is "square"
    image-type(rectangle(20, 10, "solid", "blue")) is "landscape"
  end
end
```

**Another note:** You don't need to use == to compare a value to true or false: you can just write the value or expression on its own!

# Acknowledgements



- This lecture incorporates material from:
- Kathi Fisler, Brown University,
- Gregor Kiczales, University of British Columbia,
- And, Jonathan Gordon