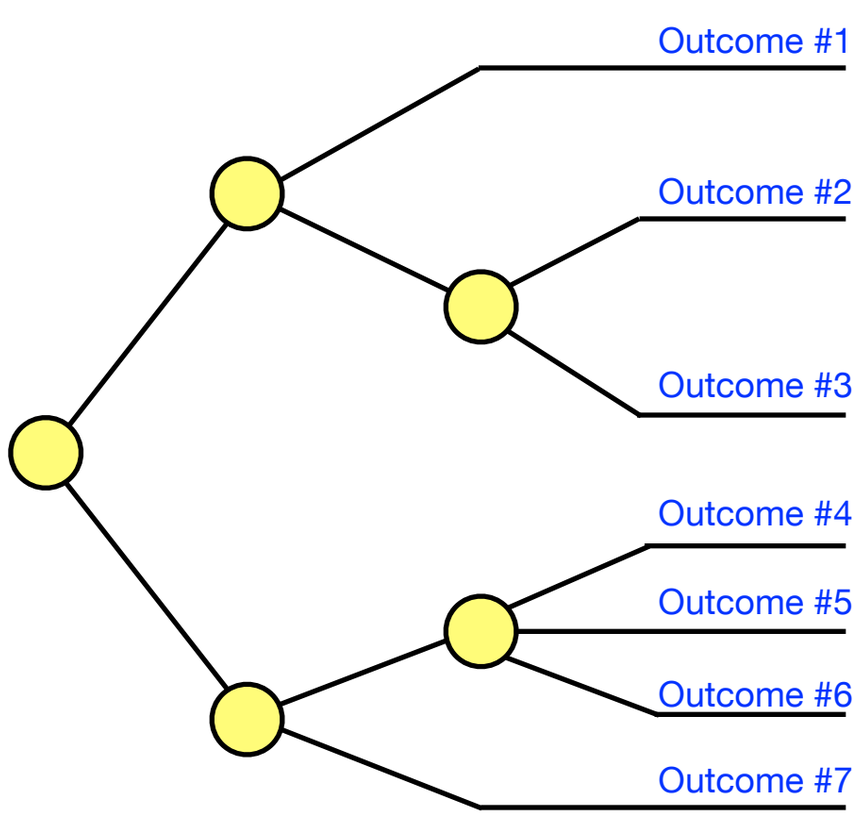
CS100: CPADS

Decisions

David Babcock / James Moscola
Department of Physical Sciences
York College of Pennsylvania

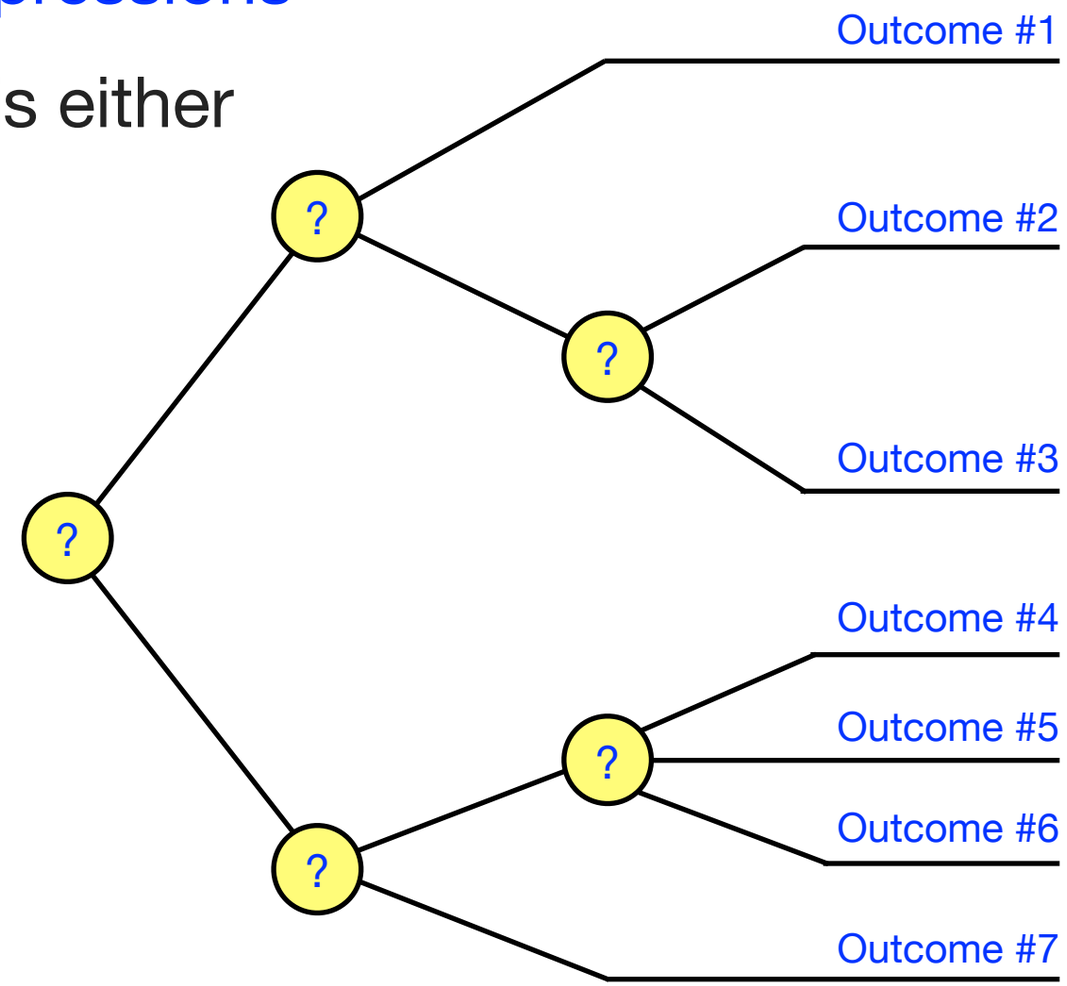
Decisions

- **Just like a human, programs need to make decisions**
  - Should turtle turn left or right?
  - Should this piece of code execute?
  - Should a different piece of code execute?
  - Which piece of code should execute?
- **Most programs have multiple branches of execution**
  - Can produce different output based on the decisions made while the program was run



Decisions

- **To determine which branch of code should be executed requires that a decision be made**
  - Decisions are based on boolean expressions
  - The result of a boolean expression is either *True* or *False*



- **In Python . . .**
  - A value of 0 is considered *False*
  - **Any** other value is considered *True* (even negative numbers)

Comparison Operators

- **Boolean expressions can consist of comparison operators and logical operators (you have already seen arithmetic operators such as +, -, *, etc.)**
  - Will always simplify to either *True* or *False*

- **Comparison Operators compare values and return True / False:**

- Equality	==	$x == y$	Is x equivalent to y?
- Inequality	!=	$x != y$	Is x not equivalent to y?
- Greater than	>	$x > y$	Is x greater than y?
- Less than	<	$x < y$	Is x less than y?
- Greater Than/Equal	>=	$x >= y$	Is x greater than or equal to y?
- Less Than/Equal	<=	$x <= y$	Is x less than or equal to y?

Logical Operators

- **Logical operators combine multiple boolean (True / False) values into a single boolean value**
  - AND *and* *x and y* True if *BOTH x AND y* are true
  - OR *or* *x or y* True if *EITHER x OR y* are true
  - NOT *not* *not x* True if *x* is False (negates a boolean expression)

Examples:

(True *or* False) ==> True
(False *or* True) ==> True
(True *or* True) ==> True
(False *or* False) ==> False

(True *and* True) ==> True
(True *and* False) ==> False

(*not* False) ==> True
(*not* True) ==> False

More Examples:

(*not* (True *and* False)) ==> ?
(*not* (False)) ==> True

((True *or* False) *and* True) ==> ?
((True) *and* True) ==> True

Combining Comparison and Logical Operators

- **Comparison and Logical Operators can be combined to create more complex boolean expressions ("questions")**

Example #1:

```
x = 5
y = 6

((x <= 6) and (y == 6))

# ------------------------

# ((5 <= 6) and (6 == 6))
# ( (True)  and  (True) )
#          ( True )
```

Example #2:

```
x = 5
y = 6

((x <= 6) and (x+2 == y))

# ------------------------

# ((5 <= 6) and (5+2 == 6))
# ((5 <= 6) and   (7 == 6))
# ( (True)  and   (False) )
#         ( False )
```

Conditional Expressions

- **Conditional expressions are used to make a decision and control the flow of a program**
- **A conditional expression in Python starts with the keyword '`if`' and can take multiple different forms**

`if-statement`

```
if condition:
  STATEMENTS to executes if condition is true
```

`if-else-statement`

```
if condition:
  STATEMENTS to executes if condition is true
else:
  STATEMENTS to executes if condition is false
```

Executes one or the other, but NOT BOTH

Conditional Expressions

if-elif-statement

```
if condition:
  STATEMENTS to executes if condition is true
elif condition2:
  STATEMENTS to executes if condition2 is true
```

Executes one or the other, but NOT BOTH

if-elif-statement

```
if condition:
  STATEMENTS to executes if condition is true
elif condition2:
  STATEMENTS to executes if condition2 is true
elif condition3:
  STATEMENTS to executes if condition3 is true
...
```

Conditional Expressions

if-elif-else-statement

```
if condition:
  STATEMENTS to executes if condition is true
elif condition2:
  STATEMENTS to executes if condition2 is true
else:
  STATEMENTS to executes if condition and condition2 are BOTH false
```

Examples

Example #1:

```
if x<=21:
  print 'Good'
else:
  print 'Bad'
```

Example #2a:

```
if x > y:
  print 'x is greater than y'
if x < y:
  print 'x is less than y'
if x==y:
  print 'x is equal to y'
```

Example #2b:

```
if x > y:
  print 'x is greater than y'
elif x < y:
  print 'x is less than y'
elif x == y:
  print 'x is equal to y'
else:
  print 'Error'
```

Conditional Iteration

- **Previously discussed fixed iteration**
  - Repeating a block of code a fixed number of times (known before loop starts executing)
- **Don't always know how many time we want a loop to execute**
  - Conditional iteration combines decisions with loops
  - Repeat a block of code *until* some condition is met

Conditional Iteration

- **"While condition is true, do this"**

`while-loop`

```
while condition:
  STATEMENTS to executes while condition is true
```

- **IMPORTANT: Be sure to update at least one of the values in your condition inside the `while-loop`**
  - If the condition is not altered inside the loop, then the loop will NEVER terminate (i.e. infinite loop)

Conditional Iteration Example

- **The following will prompt a user for input, and continue to prompt a user for input until the user enters a value that is *greater than 0***

while-loop

```
num = 0

while num <= 0:
  var = raw_input("Enter a value greater than 0: ")
  num = int(var)
```

- **NOTE: The condition is dependent on the value of `num`, therefore `num` MUST change insider the loop**

A More Interesting Example

- **This examples asks for user input, but only allows the user 10 tries to get the correct input. A message is printed at the end to indicate how many attempts it took the user.**

```
user_input = 0
num_attempts = 0
MAX_ATTEMPTS = 10

while ((user_input <= 0) and (num_attempts < MAX_ATTEMPTS)):
    var = raw_input("Enter a value greater than 0: ")
    user_input = int(var)
    num_attempts = num_attempts + 1


if (num_attempts == 1):
    print "User provided correct input on first try"
elif (num_attempts >= 10):
    print "User failed to provide correct input after 10 attempts"
else:
    print "User provided correct input on attempt # %i" % num_attempts
```