Lesson 4
Control Flow and Loops

LEARNING OBJECTIVES

You will be able to...

- **Read** and **write** basic Python syntax for conditional statements and program control flow including if-else, comparison operators, for loop, and while loops
- **Analyze** the purpose, advantages, and disadvantages to using for loops and while loops using a worksheet
- **Execute** a basic program that uses conditional statements and loops

DO NOW

- Take out the pieces of paper in the bags on your tables
- In groups of three, order the pieces of paper as best you can (3 minutes)
- Was this hard to do?
    - Individually, write down why it's important to have a clear set of instructions for proper decision making (3 minutes)

CONTROL FLOW

- In computer science, **control flow** is the order in which instructions or function calls are executed in a program.

- Why do you think computers need precise control flow instructions? Do you think they would have trouble sorting through lines of instructions that are shuffled in a bag?

Part I: Conditionals

IF STATEMENTS

- An **if** statement tests a *condition*. **If** the condition is true, **then** whatever action is listed next gets carried out.
  - "*If* it is raining → *then* put on a raincoat and pack an umbrella"

```
teachers = ['barnabas', 'foster', 'ibrahim', 'cobbina']
     if len(teachers) > 3:
               print "Wow, we have a lot of teachers!"
```

- Syntax! Note the colon and indentation.

```
teachers = ['barnabas', 'foster', 'ibrahim', 'cobbina']
     if len(teachers) > 4:
               print "Wow, we have a lot of teachers!"
```

What is the output here? Is this an error?

```
No error and no output!
```

IF/ELSE

- If the condition in the if-statement is false, the actions under **else** will run:
    - “Otherwise, put on a t-shirt”

```
teachers = ['barnabas', 'foster', 'ibrahim', 'cobbina']
     if len(teachers) > 4:
               print "Wow, we have a lot of teachers!"
else:
     print "I wish we had even more teachers..."
```

Note: **else** does not need a conditional to evaluate. It is a way of catching all other cases

IF/ELIF/ELSE

- Many times, you will want to test a series of conditions, rather than just an either-or situation. You can do this with a series of **if-elif-else statements**

```
teachers = ['barnabas', 'foster', 'ibrahim', 'cobbina']
     if len(teachers) > 4:
               print "We have too many teachers!"
elif len(teachers) == 4:
     print "We have just the right number of teachers."
else:
     print "I wish we had even more teachers..."
```

- You need one **if** statement to begin, but there is no limit to how many conditions you can test. You can have as many **elif** statements as you want

INDIVIDUAL ACTIVITY: If/elif/else statements

Take 5 minutes to complete the first activity on the worksheet.

If you finish early, discuss your answers with either a teacher or a classmate.

Learning Check: IF/ELIF/ELSE

What is the difference in **output** between these two conditionals?

```
teachers = ['barnabas', 'foster', 'cobbina']
     if 'barnabas' in teachers:
               print "Hello Barnabas!"
if 'foster' in teachers:
               print "Hello Foster!"
if 'cobbina' in teachers:
               print "Hello Cobbina!"
```

```
teachers = ['barnabas', 'foster', 'cobbina']
     if 'barnabas' in teachers:
               print "Hello Barnabas!"
elif 'foster' in teachers:
               print "Hello Foster!"
elif 'cobbina' in teachers:
               print "Hello Cobbina!"
```

```
Hello Barnabas!
Hello Foster!
     Hello Barnabas!
Hello Cobbina!
```

Conditional Operators

- What if you would like to make more complicated if statements that check multiple conditions? You can use the **and** / **or** operators:

```
if x > 1 and x < 3:
          print "x equals 2!"

if x < 0 or x > 1:
          print "x can be anything except numbers between 0 and 1!"
```

- **if** , **and** evaluates both conditions and is only True if **both** conditions are True
- **if, or** evaluates both conditions and is True if **at least one** condition is True

Part II: Loops

LOOPS

- A loop is a sequence of *instructions* that are repeated until a certain *condition* is reached
  - While you are hungry → eat breakfast

Example: Adding money to a bank account

WHILE LOOPS

- A while loop tests an *initial condition*. If that condition is true, the *loop* starts executing.
- Every time the loop finishes, the condition is reevaluated. As soon as the condition becomes false, the loop stops executing.

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

Turn-and-talk: Why is it important that the condition of a while loop be as specific as possible?

FOR LOOPS

- For loops iterate over lists

```
animals = ["dog", "cat", "mouse"]
for animal in animals:
            print animal + " is an animal"
```

Syntax! “for” is always paired with “in”

Note the colon!

Note: “animal” is a variable we’ve chosen to represent “item in the given list”

INDIVIDUAL ACTIVITY: Loops and efficiency

Take 5 minutes to complete the second activity on the worksheet.

If you finish early, discuss your answers with either a teacher or a classmate.

FOR LOOPS

- For loops can also iterate over numerical ranges. Range(number) loops from zero to the given number

```
for i in range(4):
    print i
# 0, 1, 2, 3
```

Note that 4 is excluded!

- Range(num1, num2) loops from the lower number to the higher one:

```
for i in range(4, 9):
    print i
# 4, 5, 6, 7, 8
```

Note that 9 is excluded!

FOR LOOPS

- Range(num1, num2, num3) loops from the first argument to the second one, using the third argument as a step size:

```
for i in range(0, 11, 2):
    print i
# 0, 2, 4, 6, 8, 10
```

Turn-and-talk: How is the “range” for loop similar to iterating over a list?

```
for i in range(4):

for animal in ["dog", "cat", "mouse", "lion"]:
```

EFFICIENCY IN LOOPS

- Turn-and-talk: Why is it so important to have your loops run as efficiently as possible?

Why is this code inefficient? What would you do to fix it?

Move line 3 to the top
since x never changes!

```
1 multipliers = [1,2,3,4]
2 for num in multipliers:
3       x = 1000 * 5^2 * 6.345
4       print x * num
```

INDIVIDUAL ACTIVITY: Loops and efficiency

Take 5 minutes to complete the third activity on the worksheet.

If you finish early, discuss your answers with either a teacher or a classmate.

Two different loops can solve the same problem

For Loops

```
for number in range(1,10):
        print number
```

While Loops

```
number = 1
while number  < 10:
        print number
        number += 1
```

RECAP OF TYPES OF LOOPS

- for item in list:

  …
- for x in range(10):

  …
- for x in range(10, 5, -1):

  …
- while x < 10:

  …

PARTNER ACTIVITY: Recap of Loops

Take 5 minutes to complete the fourth activity on the worksheet with the person sitting next to you.

If you finish early, discuss your answers with either a teacher or a classmate.

COMBINING CONDITIONALS AND LOOPS

```
# A list of desserts I like.
desserts = ['ice cream', 'chocolate', 'roasted plantain', 'asana']
favorite_dessert = 'asana'

# Print the desserts out, but let everyone know my favorite dessert.
for dessert in desserts:
    if dessert == favorite_dessert:
        print dessert + " is my favorite dessert!"
    else:
        # I like these desserts, but they are not my favorite.
        print "I like " + dessert + " but it is not my favorite"
```

BREAK AND CONTINUE

- A ***break*** statement is only found inside loops, and tells the program to immediately exit out of the set of loop instructions *if* a condition is met

```
x = 0
     for x in range(5):
               x = x + 1
               if x == 3:
                         break
               print "Printing " + str(x)
     print "Loop is over"
```

What does this code output?

```
Output:
Printing 1
Printing 2
Loop is over
```

BREAK AND CONTINUE

- A ***continue*** statement tells the program to skip the current iteration of the loop, but to go on and complete the rest of the loop

```
x = 0
     for x in range(5):
               x = x + 1
               if x == 3:
                         continue
               print "Printing " + str(x)
     print "Loop is over"
```

What does this code output?

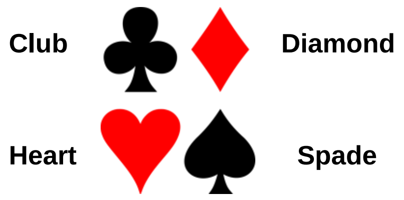
```
Output:
Printing 1
Printing 2
Printing 4
Printing 5
Loop is over
```

NESTED LOOPS

- Some situations call for putting one loop inside another, called ***nesting***
- Turn-and-talk: Why would this be useful? Brainstorm some examples

Example: Printing all cards in a deck

```
suits = ['Spades', 'Clubs', 'Diamonds', 'Hearts']
values = ['Ace', 2, 3, 4, 5, 6, 7, 8, 9, 10, 'Jack', 'Queen', 'King']
for suit in suits:
    for value in values:
        print str(value) + " of " + str(suit)
```



INDIVIDUAL ACTIVITY: Nested Loops

Take 5 minutes to complete the second and third activities on the worksheet.

If you finish early, discuss your answers with either a teacher or a classmate.

LET’S WRAP IT UP!

- Discuss answers to worksheet

- Today we learned that **control flow** is the order in which instructions are executed in a program

- Turn-and-talk: Why are control flow statements so important? How do they allow us to write more complicated programs, as well as more concise and efficient code?