# Introduction to Python

## Lecture 2: Loops and Conditionals

Daniel Kadyrov

July 4th, 2023

# Loops

- Loops are used to repeat a block of code a certain number of times.
- There are two types of loops in Python: `for` loops and `while` loops.
  - `for` loops are used to iterate over a sequence.
  - `while` loops are used to repeat a block of code while a condition is true.
- Loops are useful for automating repetitive tasks, iterating over data, and more.
- Loops can be broken out of using the `break` keyword.

# For Loops

- `for` loops are used to iterate over a sequence.
- The syntax for a `for` loop is as follows:

```
1  for <variable> in <sequence>:
2      <code>
```

- The `<variable>` is a variable that will be assigned to each element in the `<sequence>` one at a time.
- The `<code>` is the code that will be executed for each element in the `<sequence>`.
- The `<sequence>` can be a list, tuple, string, or any other iterable object.

# For Loops

- Here is an example of a `for` loop:

```
1  for i in [1, 2, 3, 4, 5]:
2      print(i)
```

- This will print the numbers 1 through 5.
- The `<variable>` `i` is assigned to each element in the list one at a time.
- The `<code>` `print(i)` is executed for each element in the list.

# For Loops

- Here is another example of a `for` loop:

```
1  for i in range(5):
2      print(i)
```

- This will print the numbers 0 through 4.
- The `range()` function returns a sequence of numbers.
- The `range()` function can take up to three arguments: `range(start, stop, step)`.
- The `start` argument is the number to start at (default is 0).
- The `stop` argument is the number to stop at (not included).
- The `step` argument is the number to increment by (default is 1).

# For Loops

- Here is another example of a `for` loop:

```
1  for i in range(1, 10, 2):
2      print(i)
```

- This will print the odd numbers from 1 to 9.
- The `range()` function can be used to iterate over a sequence of numbers.
- The `range()` function can be used to iterate over a sequence of numbers.

# While Loops

- `while` loops are used to repeat a block of code while a condition is true.
- The syntax for a `while` loop is as follows:

```
1  while <condition>:
2      <code>
```

- The `<condition>` is a boolean expression that is evaluated each time the loop is run.
- The `<code>` is the code that will be executed while the `<condition>` is true.

# While Loops

- Here is an example of a `while` loop:

```
1  i = 0
2  while i < 5:
3      print(i)
4      i += 1
```

- This will print the numbers 0 through 4.
- The `<condition>` `i < 5` is evaluated each time the loop is run.
- The `<code>` `print(i)` is executed while the `<condition>` is true.
- The `<code>` `i += 1` increments the variable `i` by 1 each time the loop is run.

# Nested Loops

- Loops can be nested inside each other.
- Here is an example of a nested `for` loop:

```
1  for i in range(5):
2      for j in range(5):
3          print(i, j)
```

- This will print the numbers 0 through 4.
- The `<code>` `print(i, j)` is executed for each element in the list.
- The `<variable>` `i` is assigned to each element in the list one at a time.
- The `<variable>` `j` is assigned to each element in the list one at a time.

# Conditionals

- Conditionals are used to execute a block of code if a condition is true.
- There are three types of conditionals in Python: `if` statements, `elif` statements, and `else` statements.
  - `if` statements are used to execute a block of code if a condition is true.
  - `elif` statements are used to execute a block of code if another condition is true.
  - `else` statements are used to execute a block of code if no other condition is true.
- Conditionals are useful for executing code based on certain conditions.
- Conditionals can be nested inside each other.

# If Statements

- `if` statements are used to execute a block of code if a condition is true.
- The syntax for an `if` statement is as follows:

```
1  if <condition>:
2      <code>
```

- The `<condition>` is a boolean expression that is evaluated.
- The `<code>` is the code that will be executed if the `<condition>` is true.

# If Statements

- Here is an example of an `if` statement:

```
1  if x > 0:
2      print("x is positive")
```

- This will print `x is positive` if the variable `x` is greater than 0.
- The `<condition>` `x > 0` is evaluated.
- The `<code>` `print("x is positive")` is executed if the `<condition>` is true.

# Elif Statements

- `elif` statements are used to execute a block of code if another condition is true.
- The syntax for an `elif` statement is as follows:

```
1  if <condition>:
2      <code>
3  elif <condition>:
4      <code>
```

- The `<condition>` is a boolean expression that is evaluated.
- The `<code>` is the code that will be executed if the `<condition>` is true.

# Elif Statements

- Here is an example of an `elif` statement:

```
1  if x > 0:
2      print("x is positive")
3  elif x < 0:
4      print("x is negative")
```

- This will print `x is positive` if the variable `x` is greater than 0.
- This will print `x is negative` if the variable `x` is less than 0.
- The `<condition>` `x > 0` is evaluated.
- The `<code>` `print("x is positive")` is executed if the `<condition>` is true.
- The `<condition>` `x < 0` is evaluated.
- The `<code>` `print("x is negative")` is executed if the `<condition>` is true.

# Else Statements

- `else` statements are used to execute a block of code if no other condition is true.
- The syntax for an `else` statement is as follows:

```
1  if <condition>:
2      <code>
3  elif <condition>:
4      <code>
5  else:
6      <code>
```

- The `<condition>` is a boolean expression that is evaluated.
- The `<code>` is the code that will be executed if the `<condition>` is true.

# Else Statements

- Here is an example of an `else` statement:

```
1  if x > 0:
2      print("x is positive")
3  elif x < 0:
4      print("x is negative")
5  else:
6      print("x is zero")
```

- This will print `x is positive` if the variable `x` is greater than 0.
- This will print `x is negative` if the variable `x` is less than 0.
- This will print `x is zero` if the variable `x` is equal to 0.
- The `<condition>` `x > 0` is evaluated.
- The `<code>` `print("x is positive")` is executed if the `<condition>` is true.
- The `<condition>` `x < 0` is evaluated.
- The `<code>` `print("x is negative")` is executed if the `<condition>` is true.
- The `<code>` `print("x is zero")` is executed if no other `<condition>` is true.

# Nested Conditionals

- Conditionals can be nested inside each other.
- This means that conditionals can be inside other conditionals.
- Here is an example of nested conditionals:

```
1  if x > 0:
2      if x > 10:
3          print("x is greater than 10")
4      else:
5          print("x is between 0 and 10")
6  else:
7      print("x is less than or equal to 0")
```

# Nested Conditionals

- This will print `x is greater than 10` if the variable `x` is greater than 10.
- This will print `x is between 0 and 10` if the variable `x` is between 0 and 10.
- This will print `x is less than or equal to 0` if the variable `x` is less than or equal to 0.
- The `<condition>` `x > 0` is evaluated.
- The `<condition>` `x > 10` is evaluated if the `<condition>` `x > 0` is true.
- The `<code>` `print("x is greater than 10")` is executed if the `<condition>` `x > 10` is true.
- The `<code>` `print("x is between 0 and 10")` is executed if the `<condition>` `x > 10` is false.
- The `<code>` `print("x is less than or equal to 0")` is executed if the `<condition>` `x > 0` is false.

# Nested Conditionals

- Here is another example of nested conditionals:

```
1  if x > 0:
2      if x > 10:
3          print("x is greater than 10")
4      elif x > 5:
5          print("x is between 5 and 10")
6      else:
7          print("x is between 0 and 5")
8  else:
9      print("x is less than or equal to 0")
```