Module 3

Control Statements

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.1.1: Program Flow

- Describes the way in which statements are executed.
- Python has top-down program flow.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.1.2: Control Statement

- Control statement is structure that conditionally changes program flow.
- Two main controls statements in Python:
  - `if`
  - `while`

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.2: The if Statement (1 of 2)

- The if statement allows execution of block of code if condition is true, otherwise it can run alternate block in `else` clause.
  - `else` clause is optional.
- Can chain multiple `if` statements together.

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.2: The if Statement (2 of 2)

```
if condition:
    # Run this code if the condition evaluates to True
else:
    # Run this code if the condition evaluates to False
```

Snippet 3.1

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.3: The while Statement (1 of 2)

- Allows execution of block of code repeatedly, as long as condition is true.
  - `else` clause is optional.
- Can also have `else` clause.
  - This will be executed exactly once when condition is no longer true.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.3: The while Statement (2 of 2)

```
while condition:
    # Run this code while condition is true
    # Replace the "condition" above with an actual condition
    # This code keeps running as long as the condition evaluates to True
else:
    # Run the code in here once the condition is no longer true
    # This code only runs one time unlike the code in the while block
```

Snippet 3.13

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.4: while Versus if

- `if` gives opportunity to branch execution of code based on condition.
- `while` gives opportunity to run block of code multiple times as long as condition is true.
  - Can be considered a loop.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.5: Loops

- Way to execute specific block of code multiple times.
- Used to iterate (or loop) over iterables.
- Iterables
  - Are anything that can be looped over.
  - Are anything that can appear on the right side of a `for` loop.
  - Think of as a collection of things that have been grouped together.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.6: The for Loop (1 of 2)

- Also referred to as the `for…in` loop.
- Used when you want to repeatedly execute a block of code a given number of times.
- Runs a predetermined number of times.
  - `while` loop runs an arbitrary number of times.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.6: The for Loop (2 of 2)

```
# Iterable can be anything that can be looped over e.g. a list
# Member is a single constituent of the iterable e.g. an entry in a list
for member in iterable:
    # Execute this code for each constituent member of the iterable
    pass
```

Snippet 3.33

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.6.1: Using else

- `else` statement optional for `for` loop.
- Will execute once when loop exits cleanly.
- Useful for debugging.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.7: The range Function

- The range function is a built-in function that generates list of numbers.
- Used to perform action predetermined number of times.
- Syntax: `range([start], stop, [step])`
  - `start`: Optional. Starting number in sequence.
  - `stop`: Generate number up to but not including this number.
  - `step`: Optional. Difference between each number in sequence.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.7: The range Function

- Syntax: `range([start], stop, [step])`
  - `start`: Optional. Starting number in sequence.
  - `stop`: Generate number up to but not including this number.
  - `step`: Optional. Difference between each number in sequence.

```
for a in range(1,4):  #creates a range of numbers starting at 1 end at 4
    for b in range(a): #selects values from range variable a
        print("*", end=' ') #prints the * based on the value, end returns a new line
    print()
```

```
*
**
***
```

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.7: The range Function

- Syntax: `range([start], stop, [step])`
  - `start`: Optional. Starting number in sequence.
  - `stop`: Generate number up to but not including this number.
  - `step`: Optional. Difference between each number in sequence.

```
sports = [“baseball”, “cricket”, “soccer”] #creates list
for x in sports: # x represents each list item iteration variable
        print(x) #displays each item one at a time
>>> baseball
>>> cricket
>>> soccer
```

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.7: The range Function

- Syntax: `range([start], stop, [step])`
  - `start`: Optional. Starting number in sequence.
  - `stop`: Generate number up to but not including this number.
  - `step`: Optional. Difference between each number in sequence.

```
>>> r = list(range(10)) #creates a range that starts at 0 and ends at 9
>>> print(r) #prints entire range

[0,1,2,3,4,5,6,7,8,9]
```

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.8: Nesting Loops

- Practice of placing loops inside other loops.
- Important to use for accessing data inside complex data structures.
- No limit to how far you can nest.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.9.1: The break Statement (1 of 2)

- The break statement allows you to exit loop based on external trigger.
- Usually used in conjunction with `if` statement.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.9.1: The break Statement (1 of 2)

- The break statement allows you to exit loop based on external trigger.
- Usually used in conjunction with `if` statement.

```
for value in “Python”: #the variable called value is passed the string Python
        if value == “t”: #the if statement is used to search the string for a value of t
                break #if the loop control variable identifies the letter t, it breaks from the execution
        print(value); #prints the indexes within the value variable
>>> P
>>> y
```

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.9.1: The break Statement (2 of 2)

```
# Loop over all numbers from 1 to 10
for number in range(1,11):
    # If the number is 4, exit the loop
    if number == 4:
        break

    # Calculate the product of number and 2
    product = number * 2
    # Print out the product in a friendly way
    print(number, '* 2 = ', product)

print('Loop completed')
```

Snippet 3.59

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.9.2: The continue Statement (1 of 3)

- The continue statement allows you to skip over part of loop where external condition is triggered, but goes back to top of loop and continues execution.
- Usually used in conjunction with `if` statement.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.9.2: The continue Statement (2 of 3)

```
# Loop over all numbers from 1 to 10
for number in range(1,11):
    # If the number is 4, continue the loop from the top
    if number == 4:
        continue

    # Calculate the product of number and 2
    product = number * 2
    # Print out the product in a friendly way
    print(number, '* 2 = ', product)


print('Loop completed')
```

Snippet 3.61

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.9.2: The continue Statement (3 of 3)

```
1 * 2 = 2
2 * 2 = 4
3 * 2 = 6
5 * 2 = 10
6 * 2 = 12
7 * 2 = 14
8 * 2 = 16
9 * 2 = 18
10 * 2 = 20
Loop completed
```

Snippet 3.62

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.9.3: The pass Statement (1 of 2)

- Allows you to handle an external trigger condition without affecting the execution of the loop.

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Lesson 3.9.3: The pass Statement (2 of 2)

```
# Loop over all numbers from 1 to 10
for number in range(1,11):
    # If the number is 4, proceed as normal
    if number == 4:
        pass

    # Calculate the product of number and 2
    product = number * 2
    # Print out the product in a friendly way
    print(number, '* 2 = ', product)

print('Loop completed')
```

Snippet 3.63

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Example: Sample 1

```
#request input of true of false from user
answer = input("Return YES or NO: Is SMC's mascot is a Roadrunner?:\n")
answer = str(answer.upper()) #parses string value to all upper case to prepare for comparison
if answer == "YES": #compares variable year to YES and prints correct if a match
    print('Correct’)
elif answer == "NO": #compares variable year to NO and prints incorrect if not a match
    print('Incorrect’)
elif answer != ("YES" or "NO"): #compares answers to YES or NO
print('Please answer YES or NO')print("Let's Go SMC!") #Displays Let's go SMC
```

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

Example: Sample 1

```
#Ask user for user name
user_name = input("Enter your User Name: ")

#This is a stored clear text password
stored_password = "Pa$$w0rd"

#variable initiated to value of False
authentication = False

#Loop while authentication is set to false
while authentication == False:
    password = input("Enter your password: ") #asks for user password
    if password == stored_password: #compares user input to stored password
        print(f"Welcome back {user_name} ") #displays user name with Welcome Back
        authentication = True #sets authentication to True in order to exit While loop
    else:
        print("Incorrect password, try again... ") #asks user to try again and remains in while loop
```

CENGAGE

Python Fundamentals, 1st Edition. © 2021 Cengage. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.