Conditional Statements and Loops

8.1 - Compare Values

Solutions to review exercises

Exercise 1

Test whether these expressions are True or False

```
print(1 <= 1)
print(1 != 1)
print(1 != 2)
print("good" != "bad")
print("good" != "Good")
print(123 == "123")
```

Exercise 2

Fill in the blank so that each of the following expressions are True

3 __ 4

Any of the following:

```
3 < 4
3 <= 4
3 != 4
```

10 __ 5

Any of the following:

```
10 > 5
10 >= 5
```

Conditional Statements and Loops

10 != 5

"jack" __ "jill"

Any of the following:

"jack" < "jill"

"jack" <= "jill"

"jack" != "jill"

42 __ "42"

42 != "42"

8.2 - Add Some Logic

Solutions to review exercises

Exercise 1

Test whether these expressions are True or False

print((1 <= 1) and (1 != 1))

print(not (1 != 2))

print(("good" != "bad") or False)

print(("good" != "Good") and not (1 == 1))

Exercise 2

Add parentheses so that the following expressions all

evaluate to True

False == not True

print(False == (not True))

True and False == True and False

print((True and False) == (True and False))

not True and "A" == "B"

Conditional Statements and Loops

```
print(not (True and "A" == "B"))

# 8.3 - Control the Flow of Your Program

# Solutions to review exercises



# Exercise 1

# Display whether the length of user input is <, > or = 5 characters


my_input = input("Type something: ")


if len(my_input) < 5:
    print("Your input is less than 5 characters long.")
elif len(my_input) > 5:
    print("Your input is greater than 5 characters long.")
else:
    print("Your input is 5 characters long.")



# Exercise 2

# Number guessing program ("guess" the number 3)


print("I'm thinking of a number between 1 and 10. Guess which one.")
my_guess = input("Type in your guess: ")


if my_guess == "3":
    print("You win!")
else:
    print("You lose.")
# 8.4 - Challenge: Find the Factors of a Number
```

Conditional Statements and Loops

```
# Solution to challenge


# Display all the factors of a number chosen by the user


num = int(input("Enter a positive integer: "))
for divisor in range(1, num + 1):
    if num % divisor == 0:
        print(f"{divisor} is a factor of {num}")
# 8.5 - Break Out of the Pattern
# Solutions to review exercises



# Exercise 1
# Run in an infinite loop until the user types "q" or "Q"
while True:
    user_input = input('Type "q" or "Q" to quit: ')
    if user_input.upper() == "Q":
        break



# Exercise 2
# Display every number from 1 through 50 except multiples of 3
for i in range(1, 51):
    if i % 3 == 0:
        continue
    print(i)
# 8.6 - Recover From Errors
# Solution to review exercises
```

Conditional Statements and Loops

```
# Exercise 1
# Ask the user to enter an integer.
# Repeat the process if the user hasn't entered an integer.
while True:
    try:
        my_input = input("Type an integer: ")
        print(int(my_input))
        break
    except ValueError:
        print("try again")



# Exercise 2
# Print character and specified index in string

input_string = input("Enter a string: ")

try:
    index = int(input("Enter an integer: "))
    print(input_string[index])
except ValueError:
    print("Invalid number")
except IndexError:
    print("Index is out of bounds")
# 8.7 - Simulate Events and Calculate Probabilities
# Solutions to review exercises
```

Conditional Statements and Loops

```
from random import randint



# Exercise 1
# Write a function that simulates the roll of a die.
def roll():
    """Return random integer between 1 and 6"""
    return randint(1, 6)



# Exercise 2
# Simulate 10,000 rolls of a die and display the average number rolled.
num_rolls = 10_000
total = 0

for trial in range(num_rolls):
    total = total + roll()

avg_roll = total / num_rolls

print(f"The average result of {num_rolls} rolls is {avg_roll}")
# 8.7 - Simulate Events and Calculate Probabilities
# Solutions to review exercises



from random import randint
```

Conditional Statements and Loops

```
# Exercise 1

# Write a function that simulates the roll of a die.

def roll():

    """Return random integer between 1 and 6"""

    return randint(1, 6)



# Exercise 2

# Simulate 10,000 rolls of a die and display the average number rolled.

num_rolls = 10_000

total = 0


for trial in range(num_rolls):

    total = total + roll()


avg_roll = total / num_rolls


print(f"The average result of {num_rolls} rolls is {avg_roll}")

# 8.8 - Challenge: Simulate a Coin Toss Experiment

# Solution to challenge



# Simulate the results of a series of coin tosses and track the results


# This one is tricky to structure correctly. Try writing out the logic before

# you start coding. Some additional pointers if you're stuck:

# 1. You will need to use a `for` loop over a range of trials.

# 2. For each trial, first you should check the outcome of the first flip.

# 3. Make sure you add the first flip to the total number of flips.
```

Conditional Statements and Loops

```
# 4. After the first toss, you'll need another loop to keep flipping while you
#    get the same result as the first flip.


import random



def coin_flip():
    """Randomly return 'heads' or 'tails'."""
    if random.randint(0, 1) == 0:
        return "heads"
    else:
        return "tails"



flips = 0
num_trials = 10_000

for trial in range(num_trials):
    if coin_flip() == "heads":
        # Increment the number of flips by 1
        flips = flips + 1
        while coin_flip() == "heads":
            # Keep incrementing the total number of flips
            # until "tails" is returned by coin_flip()
            flips = flips + 1
        # Once coin_flip() return "tails", the loop will exit,
        # but we need to add one more to flips to track the
        # last flip that generated "tails"
        flips = flips + 1
```

Conditional Statements and Loops

```
    else:
        # coin_flip() returned "tails" on the first flip.
        # Increment the number of flips by 1
        flips = flips + 1
        while coin_flip() == "tails":
            # Keep incrementing the total number of flips
            # until "heads" is returned by coin_flip()
            flips = flips + 1
        # Once coin_flip() returns "heads", the loop will exit,
        # but we need to add one more to flips to track the
        # last flip that generated "heads"
        flips = flips + 1


avg_flips_per_trial = flips / num_trials
print(f"The average number of flips per trial is {avg_flips_per_trial}.")
# 8.8 - Challenge: Simulate a Coin Toss Experiement
# Alternative solution to challenge



# Simulate the results of a series of coin tosses and track the results


# This one is tricky to structure correctly. Try writing out the logic before
# you start coding. Some additional pointers if you're stuck:
# 1. You will need to use a `for` loop over a range of trials.
# 2. For each trial, first you should check the outcome of the first flip.
# 3. Make sure you add the first flip to the total number of flips.
# 4. After the first toss, you'll need another loop to keep flipping while you
#    get the same result as the first flip.
```

Conditional Statements and Loops

```
import random



def coin_flip():
    """Randomly return 'heads' or 'tails'."""
    if random.randint(0, 1) == 0:
        return "heads"
    else:
        return "tails"



flips = 0
num_trials = 10_000

for trial in range(num_trials):
    # Flip the coin once and increment the flips tally by 1
    first_flip = coin_flip()
    flips = flips + 1
    # Continue flipping the coin and updating the tally until
    # a different result is returned by coin_flip()
    while coin_flip() == first_flip:
        flips = flips + 1
    # Increment the flip tally once more to account for the
    # final flip with a different result
    flips = flips + 1

avg_flips_per_trial = flips / num_trials
print(f"The average number of flips per trial is {avg_flips_per_trial}.")
# 8.8 - Challenge: Simulate a Coin Toss Experiement
```

Conditional Statements and Loops

Alternative solution to challenge using functions

Simulate the results of a series of coin tosses and track the results

This one is tricky to structure correctly. Try writing out the logic before
you start coding. Some additional pointers if you're stuck:
1. You will need to use a `for` loop over a range of trials.
2. For each trial, first you should check the outcome of the first flip.
3. Make sure you add the first flip to the total number of flips.
4. After the first toss, you'll need another loop to keep flipping while you
get the same result as the first flip.

```
import random


def single_trial():
    """Simulate repeatedly flipping a coin until both heads and tails are seen."""
    # This function uses random.randint() to simulate a single coin toss.
    # randint(0, 1) randomly returns 0 or 1 with equal probability. We can
    # use 0 to represent heads and 1 to represent tails.

    # Flip the coin the first time
    flip_result = random.randint(0, 1)
    # Keep a tally of how many times the coin has been flipped. We've only
    # flipped once so the initial count is 1.
    flip_count = 1

    # Continue to flip the coin until randint(0, 1) returns something
```

Conditional Statements and Loops

```
    # different than the original flip_result
    while flip_result == random.randint(0, 1):
        flip_count = flip_count + 1

    # The last step in the loop flipped the coin but didn't update the tally,
    # so we need to increase the flip_count by 1
    flip_count = flip_count + 1
    return flip_count


def flip_trial_avg(num_trials):
    """Calculate the average number of flips per trial over num_trials total trials."""
    total = 0
    for trial in range(num_trials):
        total = total + single_trial()
    return total / num_trials


print(f"The average number of coin flips was {flip_trial_avg(10_000)}")
# 8.9 - Challenge: Simulate an Election
# Solution to challenge


# Simulate the results of an election using a Monte Carlo simulation

from random import random

num_times_A_wins = 0
num_times_B_wins = 0
```

Conditional Statements and Loops

```
num_trials = 10_000
for trial in range(0, num_trials):
    votes_for_A = 0
    votes_for_B = 0

    # Determine who wins the 1st region
    if random() < 0.87:
        votes_for_A = votes_for_A + 1
    else:
        votes_for_B = votes_for_B + 1

    # Determine who wins the 2nd region
    if random() < 0.65:
        votes_for_A = votes_for_A + 1
    else:
        votes_for_B = votes_for_B + 1

    # Determine who wins the 3rd region
    if random() < 0.17:
        votes_for_A = votes_for_A + 1
    else:
        votes_for_B = votes_for_B + 1

    # Determine overall election outcome
    if votes_for_A > votes_for_B:
        num_times_A_wins = num_times_A_wins + 1
    else:
        num_times_B_wins = num_times_B_wins + 1
```

Conditional Statements and Loops

```
print(f"Probability A wins: {num_times_A_wins / num_trials}")

print(f"Probability B wins: {num_times_B_wins / num_trials}")

# 8.9 - Challenge: Simulate an Election

# Alternate solution to challenge



# Simulate the results of an election using a Monte Carlo simulation


from random import random



def run_regional_election(chance_A_wins):
    """Return the result of a regional election, either "A" or "B".

    The chances of "A" winning are determined by chance_A_wins.
    """
    if random() < chance_A_wins:
        return "A"
    else:
        return "B"



def run_election(regional_chances):
    """Return the result of an election, either "A" or "B".

    regional_chances is a list or tuple of floats representing the
    chances that candidate "A" will win in each region.
```

Conditional Statements and Loops

```
    For example, run_election([.2, .5, .7]) will run an election with
    three regions, where candidate "A" has a 20% chance to win in the
    first region, 50% in the second, and 70% in the third.
    """
    num_regions_won_by_A = 0
    for chance_A_wins in regional_chances:
        if run_regional_election(chance_A_wins) == "A":
            num_regions_won_by_A = num_regions_won_by_A + 1

    # Return the results. Note that the number of regions won by candidate
    # "B" is the total number of regions minus the number of regions won by
    # candidate "A". The total number of regions is the same as the length
    # of the regional_chances list.
    num_regions_won_by_B = len(regional_chances) - num_regions_won_by_A
    if num_regions_won_by_A > num_regions_won_by_B:
        return "A"
    else:
        return "B"



CHANCES_A_WINS_BY_REGION = [0.87, 0.65, 0.17]
NUM_TRIALS = 10_000

# Run the Monte-Carlo simulation
num_times_A_wins = 0
for trial in range(NUM_TRIALS):
    if run_election(CHANCES_A_WINS_BY_REGION) == "A":
        num_times_A_wins = num_times_A_wins + 1
```

Conditional Statements and Loops

```
# Display the probabilities that candidate A or candidate B wins the
# election. Note the probability that B wins can be calculated by
# subtracting the probability that A wins from 1.
print(f"Probability A wins: {num_times_A_wins / NUM_TRIALS}")
print(f"Probability B wins: {1 - (num_times_A_wins / NUM_TRIALS)}")
# 6.4 - Run in Circles
# Solutions to review exercises


# Exercise 1
# print the integer 2 through 10 using a "for" loop
for i in range(2, 11):
    print(i)


# Exercise 2
# print the integer 2 through 10 using a "while" loop
i = 2
while i < 11:
    print(i)
    i = i + 1


# Exercise 3
def doubles(num):
    """Return the result of multiplying an input number by 2."""
    return num * 2
```

Conditional Statements and Loops

```
# Call doubles() to double the number 2 three times

my_num = 2

for i in range(0, 3):

    my_num = doubles(my_num)

    print(my_num)
```