Module 04

While Loops

Emily Zhao
CSCI–UA.0002

Agenda

- Review Module 4 / Go over Quiz 4
- Practice Problems

Module #4

- Repetition Structures + “while” loops
- Self-referential assignment statements
- Accumulator variables
- “While” loop control
  - Infinite loops
  - `Break and Continue`
  - Sentinels
- Data Validation
- Color in Turtle Graphics

What are some reasons why we would want to use a “while” loop?

“While” loops

- If we want to run the same code multiple times
- If we only want to run code under a certain **condition**

Example: Commission calculator for a sales team

Write a program that allows the user to calculate sales commission earned by each member of a sales team.

- Currently there are 3 people on the sales team, but there may be more in the future.
- Inputs: Gross sales (float) and commission rate (float)
- Process: Commission = gross sales * commission rate
- Output: Commission earned

Calculate for person #1

```
# get sales and commission rate for person #1
sales1 = float(input("Input gross sales: "))
comm_rate1 = float(input("Input commission rate: "))

# calculate commission for person #1
comm1 = sales1 * comm_rate1

# output commission earned for person #1
print("You made", comm1, "in commissions.")
```

Add code for person #2

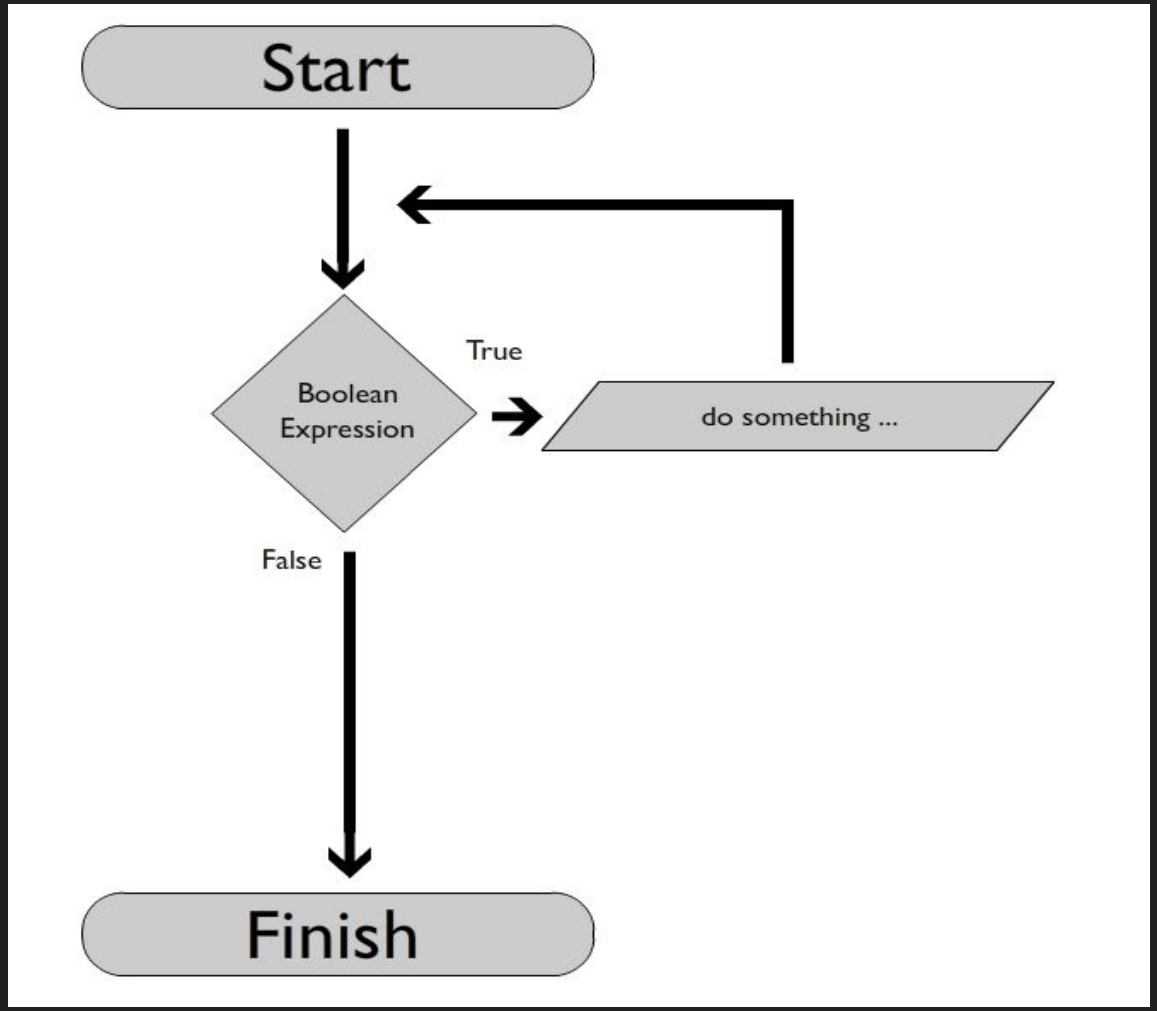
```
# get sales and commission rate for person #1
sales1 = float(input("Input gross sales: "))
comm_rate1 = float(input("Input commission rate: "))

# calculate commission for person #1
comm1 = sales1 * comm_rate1

# output commission earned for person #1
print("You made", comm1, "in commissions.")

# get sales and commission rate for person #2
sales2 = float(input("Input gross sales: "))
comm_rate2 = float(input("Input commission rate: "))

# calculate commission for person #2
comm2 = sales2 * comm_rate2

# output commission earned for person #2
print("You made", comm2, "in commissions.")
```

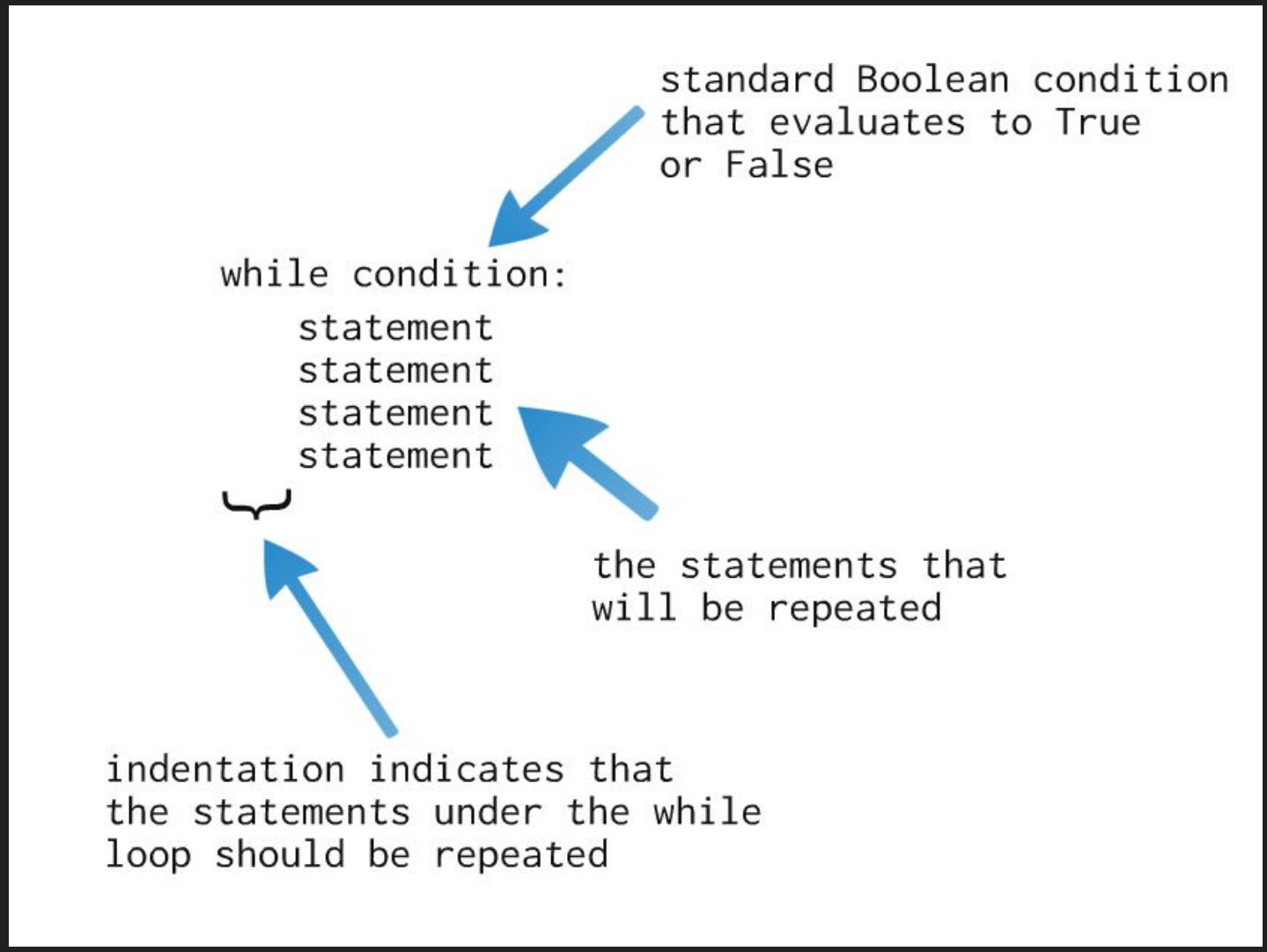
Add code for person #3

```
# get sales and commission rate for person #1
sales1 = float(input("Input gross sales: "))
comm_rate1 = float(input("Input commission rate: "))

# calculate commission for person #1
comm1 = sales1 * comm_rate1

# output commission earned for person #1
print("You made", comm1, "in commissions.")

# get sales and commission rate for person #2
sales2 = float(input("Input gross sales: "))
comm_rate2 = float(input("Input commission rate: "))

# calculate commission for person #2
comm2 = sales2 * comm_rate2

# output commission earned for person #2
print("You made", comm2, "in commissions.")

# get sales and commission rate for person #3
sales3 = float(input("Input gross sales: "))
comm_rate3 = float(input("Input commission rate: "))

# calculate commission for person #3
comm3 = sales3 * comm_rate3

# output commission earned for person #3
print("You made", comm3, "in commissions.")
```

“While” loops

A **condition-controlled loop** is a programming structure that causes a statement or set of statements to repeat as long as a condition evaluates to true




standard Boolean condition
that evaluates to True
or False
while condition:
statement
statement
statement
statement
the statements that
will be repeated
indentation indicates that
the statements under the while
loop should be repeated

- How do I calculate commissions for 3 people using a “while” loop?
- How do I calculate commissions indefinitely? Until I say stop?
- How do I calculate the total commissions for an unknown amount of people until I say to stop?

```
# How do I calculate commissions for 3 people using a while loop?

# accumulator variable
count = 0

while count < 3: # count will be 0, then 1, then 2 -> 3 times
    # get user input
    sales = float(input("Input gross sales: "))
    comm_rate = float(input("Input commission rate (i.e. .04): "))

    # calculate commission
    comm = sales * comm_rate

    # output commission
    print("You made", "$" + str(comm), "in commissions.")

    # increase count
    # count = count + 1
    count += 1
```

```
# How do I calculate commissions indefinitely?

calculate_comms = True # control variable

while calculate_comms == True:
    # get user input
    sales = float(input("Input gross sales: "))
    comm_rate = float(input("Input commission rate (i.e. .04): "))

    # calculate commission
    comm = sales * comm_rate

    # output commission
    print("You made", "$" + str(comm), "in commissions.")
```

```
# How do I calculate commissions as long as the user wants to?

calculate_comms = "yes" # control variable

while calculate_comms == "yes":
    # get user input
    sales = float(input("Input gross sales: "))
    comm_rate = float(input("Input commission rate (i.e. .04): "))

    # calculate commission
    comm = sales * comm_rate

    # output commission
    print("You made", "$" + str(comm), "in commissions.")

    # ask user if they want to continue
    calculate_comms = input("Do you want to continue? Type yes or no: ")

print("Thank you for using our program")
```

Programming Challenge: Divisibility Tester

Write a program that lets the user test to see if a series of numbers are evenly divisible by 3. If they are, print out a status message telling the user.

Extension: Start off by asking the user to enter in the number that should be used during the test (i.e. enter 5 if you want to test to see if a range of numbers is evenly divisible by 5)

Programming Challenge: Divisibility Tester

```
# set up "control variable"
check_divis = "yes"

while check_divis == "yes":
    num = int(input("Enter a number to check divisibility by 3: "))

    if num % 3 == 0:
        print("Your number is divisible by 3.")
    else:
        print("Your number is not divisible by 3.")

    check_divis = input("Would you like to check another number's divisibility? Type yes or no: ")
else:
    print("Thanks for using our program.")
```

Programming Challenge: Divisibility Tester Extension

```
# Divisibility Tester Extension

# set up "control variable"
check_divis = "yes"

# ask user for divisor input
divisor = int(input("Pick a divisor: "))

while check_divis == "yes":
    num = int(input("Enter a number to check divisibility by " + str(divisor) + ": "))

    if num % divisor == 0:
        print("Your number is divisible by " + str(divisor) + ".")
    else:
        print("Your number is not divisible by " + str(divisor) + ".")

    check_divis = input("Would you like to check another number's divisibility? Type yes or no: ")
else:
    print("Thanks for using our program.")
```

```
# counter variable
count = 5

while count < 10:
    print(count)
    count -= 1
```

Infinite loop

(control + c to end)

```
# counter variable
count = 0

while count < 10:
    count += 1
    print(count)
```

> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Break

The break keyword is used to break out a loop.

Continue

Used to end the current iteration in a loop and continues to the next iteration.

```
# counter variable
count = 0

while count < 10:
    count += 1
    if count % 2 == 0:
        continue
    print(count)
```

> 1, 3, 5, 7, 9

```
# counter variable
count = 0

while count < 10:
    count += 1
    if count % 2 == 0:
        break
    print(count)
```

```
> 1
```

Sentinels

A sentinel value is a pre-defined value that the user can type in to indicate that they are finished entering data

Example:

```
>> Enter a test score (type -1 to end): 100
>> Enter a test score (type -1 to end): 80
>> Enter a test score (type -1 to end): -1
>> Your test average is:  90 %
```

Programming Challenge: Adding Machine

- Write a program that continually asks the user for an integer
- Add the supplied integer to a total variable
- When the user enters a 0 value end the program and display the sum for the user

Programming Challenge: Adding Machine

```
# Adding Machine

# initialize total variable
total = 0

while True:
    num = int(input("Input an integer to add: "))

    # conditional for sentinel to break out of program
    if num == 0:
        break;

    # if we're still running, add num to total
    total += num

print("Your total is:", total)
print("Thanks for using our program.")
```

Data Validation

You can’t always trust the user to supply you with usable data.
"Validate" the user's input by checking to see if it meets our criteria. If not we will need to ask the user to re-supply the value.

Example:

```
Enter a positive integer:  -5
Invalid, try again!
Enter a positive integer:  0
Invalid, try again!
Enter a positive integer:  5
```

Data Validation

```
# Data validations

while True:
    num = int(input("Input an integer to add: "))

    if num < 0:
        print("Sorry, invalid number, try again.")
        continue
    else:
        break

print(num)
```

Programming Challenge: Rock, Paper, Scissors Tournament

Write a program that lets the user play a game of Rock, Paper, Scissors against the computer

Extension: End the game when either the player or the computer earns 3 points

Programming Challenge: Prime Number Checker

- Write a program that asks the user for an integer
- Test to see if the number is prime. A prime number is any number that is only divisible by 1 and itself.

Homework

- Assignment #3 (due next class)
- Self-Paced Learning Module #5 (due next week)
- Quiz #5 (due next week)