PYTHON BOOT CAMP

Module 4: Selections

Objectives

- To write Boolean expressions by using comparison operators (§4.2).
- To generate random numbers by using the random.randint(a, b) or random.random() functions (§4.3).
- To program with Boolean expressions (AdditionQuiz) (§4.3).
- To implement selection control by using one-way if statements (§4.4)
- To program with one-way if statements (§4.5).
- To implement selection control by using two-way if .. else statements (§4.6).
- To implement selection control with nested if ... elif ... else statements (§4.7).
- To avoid common errors in if statements (§4.8).
- To program with selection statements (§4.9–4.10).
- To combine conditions by using logical operators (and, or, and not) (§4.11).
- To use selection statements with combined conditions (LeapYear, Lottery) (§§4.12–4.13).
- To write expressions that use the conditional expressions (§4.14).
- To understand the rules governing operator precedence and associativity (§4.15).

Area of a Circle (Reminder)

- Write a program that will calculate the area of a circle.
- Remember:
  - Step 1: Problem-solving Phase
  - Step 2: Implementation Phase

Area of a Circle (Reminder)

- Write a program that will calculate the area of a circle.
- **Step 1**: Design your algorithm
  1. Get the radius of the circle.
  2. Compute the area using the following formula:
     - area = radius x radius x π
  3. Display the result

Area of a Circle (Reminder)

- Write a program that will calculate the area of a circle.
- **Step 2**: Implementation (code the algorithm)

```
# Constants
PI = 3.14159

# Step 1: get radius
radius = eval(input("Please enter a value for the radius: "))

# Step 2: calculate area
area = PI * radius * radius

# Step 3: display the result
print("The area for the circle of radius", radius, "is", area)
```

Area of a Circle (Reminder)

- Write a program that will calculate the area of a circle.
- Question:
  - What would happen if the user entered a negative value for the radius?
- Answer:
  - The area would be negative!
  - Clearly that is wrong
- How can we prevent this?
  - Selection statements!

Introduction

- Selection Statements
  - We can use selection statements to choose actions when two or more alternatives exist
  - Example:

```
# Constants
PI = 3.14159

# Step 1: get radius
radius = eval(input("Please enter a value for the radius: "))

# Step 2: calculate area
area = PI * radius * radius

# Step 3: display the result
if radius < 0:
    print("Incorrect input value for radius was given.")
else
    print("The area for the circle of radius", radius, "is", area)
```

Boolean Types, Values, and Expressions

- So how do you compare two values in Python?
  - Such as a radius being greater than 0?
- Python provides six comparison operators:

TABLE 4.1 Comparison Operators

Python Operator	*Mathematics Symbol*	*Name*	*Example* (radius is 5)	*Result*
`<`	$<$	less than	`radius < 0`	`False`
`<=`	$\leq$	less than or equal to	`radius <= 0`	`False`
`>`	$>$	greater than	`radius > 0`	`True`
`>=`	$\geq$	greater than or equal to	`radius >= 0`	`True`
`==`	$=$	equal to	`radius == 0`	`False`
`!=`	$\neq$	not equal to	`radius != 0`	`True`

Boolean Types, Values, and Expressions

- Comparison Operators in Python
  - Caution:
    - The equal to comparison operator is two equal signs (**==**)
    - Remember: a single equal sign (**=**) is the assignment operator
  - The result of a comparison is a **Boolean value**
    - a Boolean value can be **true** or **false**
    - Example: the following will print the result "**True**"

```
radius = 1
print(radius > 0)
```

  - A **Boolean expression** is an expression that ultimately evaluates to True or False

Boolean Types, Values, and Expressions

- Comparison Operators in Python
  - A **Boolean variable** is a variable that holds a Boolean value
  - Of course, the two possible values are True or False
  - Example:
    - We can assign Boolean values to variables.
    - Here, we assign True to the variable lights_on

```
lights_on = True
```

Boolean Types, Values, and Expressions

- Comparison Operators in Python
  - Note:
    - Internally, Python uses 1 to represent True and 0 for False
    - You can use the int function to convert a Boolean value to an int
    - Example:

```
print(int(True))      # displays 1
print(int(False))     # displays 0
```

    - You can also use the bool function to convert a numeric value to a Boolean value
      - The bool function returns False if the value was 0; otherwise, it always returns true
    - Example:

```
print(bool(0)) # displays False
print(bool(4)) # displays True
```

Generating Random Numbers

- Python provides the **`randint`** function:
  - Syntax:

    `randint(a, b)`

    - This function generates a random `int` between the values `a` and `b`, inclusive
      - This means that the random value could possibly be the values a or b as well
  - Note:
    - We must have an import statement at the top of our program in order to use the randint function:

      ```
      import random
      ```

    - Now if we want a random value between 1 and 5, we can type:

      ```
      some_value = random.randint(1, 5)
      ```

Program 1: Math Learning Tool

- Write a program to help a first grader practice addition. Your program should randomly generate two `int` values (between 1 and 9, inclusive) and ask the user the answer to the addition of those two values. Finally, you should print True or False next to the result.
- Remember:
  - Step 1: Problem-solving Phase
  - Step 2: Implementation Phase

Program 1: Math Learning Tool

- Step 1: Problem-solving Phase
  - Generate two single-digit integers for number1 (e.g., 4) and number2 (e.g., 5)
    - For this, use `random.randint(1, 9)`
  - Prompt the student to answer, "What is 4 + 5?"
  - Check whether the student's answer is correct.

```
What is 1 + 7? 8 ↵Enter
1 + 7 = 8 is True
```

```
What is 4 + 8? 9 ↵Enter
4 + 8 = 9 is False
```

Program 1: Math Learning Tool

- Step 2: Implementation Phase

```
import random

# Generate random numbers
number1 = random.randint(0, 9)
number2 = random.randint(0, 9)

# Prompt the user to enter an answer
answer = eval(input("What is " + str(number1) + " + " + str(number2) + "? "))

# Display result
print("{} + {} = {} is {}".format(number1, number2, answer,
                                  number1 + number2 == answer))
```

```
What is 1 + 7? 8 ↵Enter
1 + 7 = 8 is True
```

```
What is 4 + 8? 9 ↵Enter
4 + 8 = 9 is False
```

Generating Random Numbers

Check Yourself

- How do you generate a random integer **i** such that **0 ≤ i < 20**?

```
i = random.randint(0, 19)
```

- How do you generate a random integer **i** such that **10 ≤ i < 20**?

```
i = random.randint(10, 19)
```

- How do you generate a random integer **i** such that **10 ≤ i ≤ 50**?

```
i = random.randint(10, 50)
```

`if` Statements

- Python provides several different selection statements:
  - one-way if statements
  - two-way if-else statements
  - nested if statements
  - multi-way if-elif-else statements
  - and conditional expressions

- We start with the basic if statement…

`if` Statements

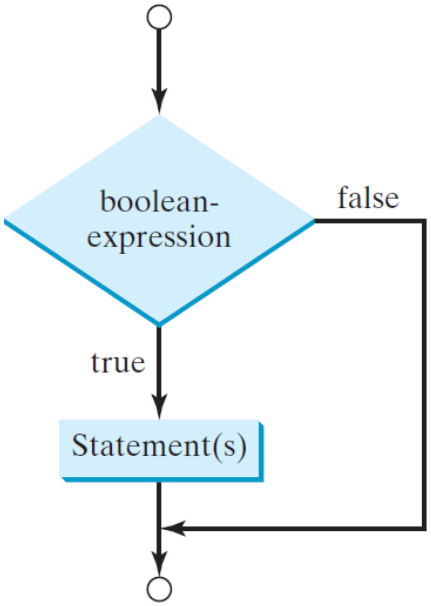
- Python one-way `if` statement:
  - A one-way `if` statement executes an action if and only if the condition is true
  - Syntax:

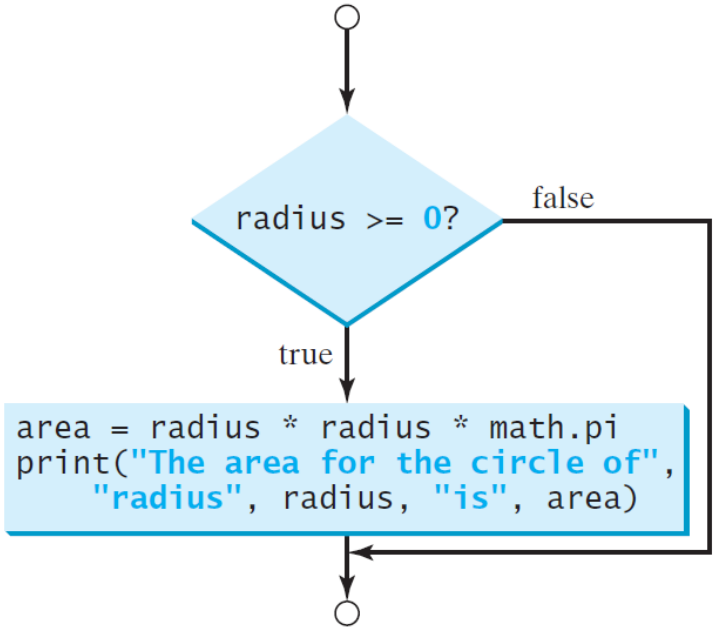
```
if boolean-expression:
    statement(s)
    # Note: these statements must be indented
```

  - All statements within a one-way `if` statement must be indented **<u>four spaces</u>**
    - Yes, you could use two spaces, or three spaces, or a tab
    - Most important is that you must be consistent with your choice
    - Important as well: the Python style guide says to use 4 spaces

`if` Statements

- Python one-way `if` statement:
  - Here's a flowchart showing the logic:





    - If the boolean-expression evaluates to true, the statements in the if block are executed.

Program 2: Math Learning Tool - Revisited

- Write a program to help a first grader practice addition. Your program should randomly generate two `int` values (between 1 and 9, inclusive) and ask the user the answer to the addition of those two values. This time, print an appropriate message to the user based on their given answer.
- Remember:
  - Step 1: Problem-solving Phase
  - Step 2: Implementation Phase

Program 2: Math Learning Tool - Revisited

- Step 1: Problem-solving Phase
  - Generate two single-digit integers for number1 (e.g., 4) and number2 (e.g., 5)
    - For this, use `random.randint(1, 9)`
  - Prompt the student to answer, "What is 4 + 5?"
  - Check whether the student's answer is correct.
  - This time, let us use one-way if statements to print something more meaningful to the user…

Program 2: Math Learning Tool - Revisited

- Step 2: Implementation Phase

```
import random

# Generate random numbers
number1 = random.randint(0, 9)
number2 = random.randint(0, 9)

# Prompt the user to enter an answer
answer = eval(input("What is " + str(number1) + " + " + str(number2) + "? "))

result = number1 + number2 == answer

# Display results:
if result == True:
    print("Correct! Great job!")

if result == False:
    print("Incorrect. The correct answer is as follows:")
    print("{} + {} = {}".format(number1, number2, number1 + number2))
```

- Start here
- But FIRST:
  - Give some examples of basic boolean expressions
  - Make some variables
  - Make some expressions
  - Use if statements to then print messages
- THEN
  - Revisit the math example

Program 3: Even or Odd

- Write a program asking the user to enter an integer number and then display if the number is even or odd.
- Remember:
  - Step 1: Problem-solving Phase
  - Step 2: Implementation Phase

Program 3: Even or Odd

- Write a program asking the user to enter an integer number and then display if the number is even or odd.
- Step 1: Problem-solving Phase
  - We start by getting a value from the user
    - That's easy
  - So how do we now check if it is even (or odd)
  - Remember: if an integer is even, there is no remainder when we divide by 2…
    - So, if an integer is even, we should get `number % 2 == 0`
    - And if an integer is odd, we should get `number % 2 == 1`

Program 3: Even or Odd

- Write a program asking the user to enter an integer number and then display if the number is even or odd.
- Step 2: Implementation Phase

```
# Get input
number = int(input("Please enter an integer: "))

# Display results:
if number % 2 == 0:
    print("You entered an EVEN number.")

if number % 2 == 1:
    print("You entered an ODD number.")
```

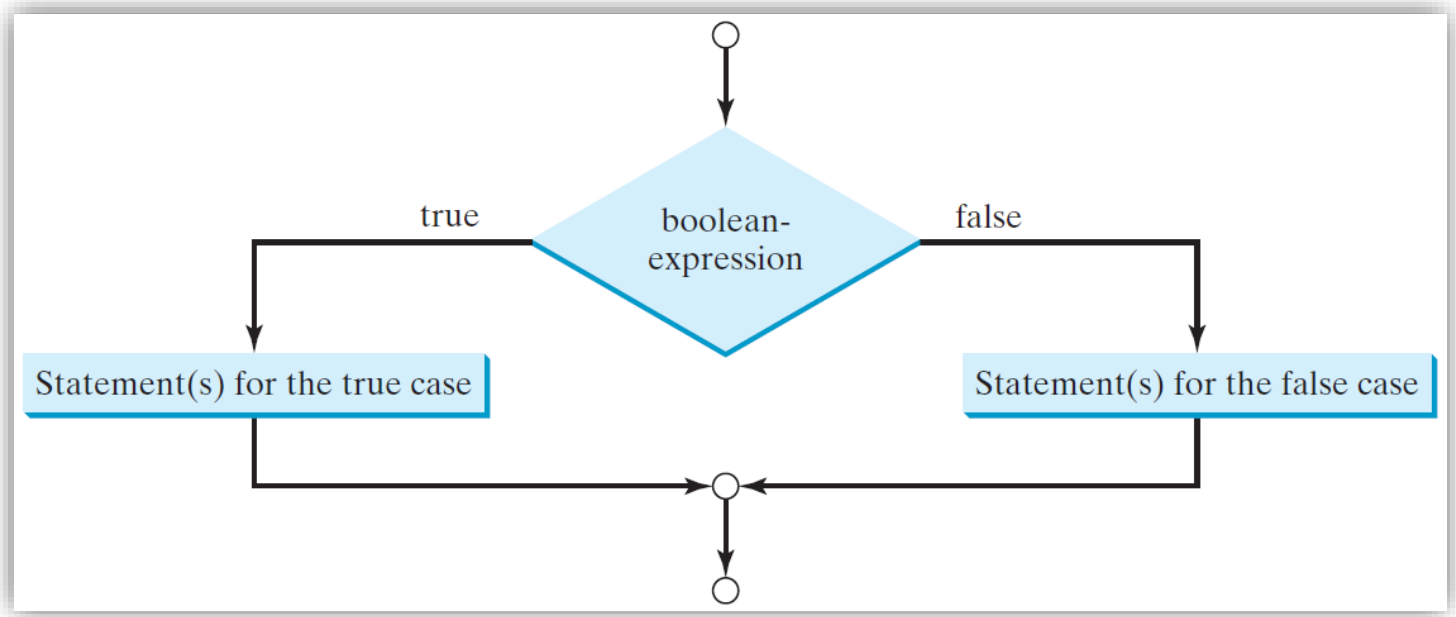
Two-way `if-else` Statements

- Remember:
  - With a one-way if statement, we only execute if the statement is true
    - If it is false, nothing is done
  - But what if you want to take alterative actions if false?
  - For this, we have a two-way if-else statement
  - Syntax:

```
if boolean-expression:
    statement(s)-for-the-true-case
else:
    statement(s)-for-the-false-case
```

Two-way `if-else` Statements

- Python two-way `if-else` statement:
  - Here's a flowchart showing the logic



Program 3: Even or Odd - Revisited

- Write a program asking the user to enter an integer number and then display if the number is even or odd (using two-way **`if-else`** statements)
- Step 2: Implementation Phase

```
# Get input
number = int(input("Please enter an integer: "))

# Display results:
if number % 2 == 0:
    print("You entered an EVEN number.")
else:
    print("You entered an ODD number.")
```

Program 2: Math Learning Tool – Revisited (Again)

- Step 2: Implementation Phase

```
import random

# Generate random numbers
number1 = random.randint(0, 9)
number2 = random.randint(0, 9)

# Prompt the user to enter an answer
answer = eval(input("What is " + str(number1) + " + " + str(number2) + "? "))

result = number1 + number2 == answer

# Display results:
if result == True:
    print("Correct! Great job!")
else:
    print("Incorrect. The correct answer is as follows:")
    print("{} + {} = {}".format(number1, number2, number1 + number2))
```

Program 4: Math Learning Tool - Subtraction

- Write a program to help a first grader practice subtraction. Your program should randomly generate two `int` values (between 1 and 9, inclusive). Next, you should make sure the first number is greater or equal to the second number.
  - Cuz we assume dealing with negatives is hard for them
- Next, prompt for the answer and display the result.
- Remember:
  - Step 1: Problem-solving Phase
  - Step 2: Implementation Phase

Program 4: Math Learning Tool - Subtraction

- Step 1: Problem-solving Phase
  - Generate two single-digit integers for number1 (e.g., 4) and number2 (e.g., 5)
    - For this, use `random.randint(1, 9)`
  - Use a one-way if statement to check if the first number is smaller than the second
    - If so, we need to swap them
    - This is easy in Python! Remember: simultaneous assignment!
  - Prompt the student to answer, "What is 7 – 3 "
  - Check whether the student's answer is correct.
  - Use a two-way if-else statements to print the result to the user

Program 4: Math Learning Tool - Subtraction

- Step 2: Implementation Phase

```
import random

# Generate random numbers
number1 = random.randint(0, 9)
number2 = random.randint(0, 9)

# IF number1 is smaller than number2, SWAP 'EM
if number1 < number2:
    number1, number2 = number2, number1;

# Prompt the user to enter an answer
answer = eval(input("What is " + str(number1) + " - " + str(number2) + "? "))

# Display results:
if number1 - number2 == answer:
    print("Correct! Great job!")
else:
    print("Incorrect. The correct answer is as follows:")
    print("{} - {} = {}".format(number1, number2, number1 - number2))
```

- Friday start here

Two-way `if-else` Statements

Check Yourself

- Write an if statement that increases pay by 3% if score is greater than 90, otherwise it increases pay by 1%.
- Answer:

```
if score > 90:
    pay = pay * 1.03
else:
    pay *= 1.01
```

  - Note: we used two different ways of multiplying just so you can see both used and become comfortable with both of them.

Two-way `if-else` Statements

Check Yourself

- What is the printout of the code in (a) and (b) if `number` is `30` and `35`, respectively?

```
if number % 2 == 0:
    print(number, "is even.")

print(number, "is odd.")
```

(a)

```
if number % 2 == 0:
    print(number, "is even.")
else
    print(number, "is odd.")
```

(b)

- Answer:
  - (a) prints

    ```
    30 is even
    30 is odd
    ```

  - (b) prints: `35 is odd`

Nested `if` and Multi-Way `if-elif-else` Statements

- Short story:
  - One `if` statement can be placed inside another `if` statement, resulting in a *nested* `if` *statement*
- Details:
  - You've seen `if` and `if-else` statements
  - What can you put inside those statements?
  - Answer: **<u>any</u>** legal Python statement
  - And this includes another `if` (or `if-else`) statement!

Nested `if` and Multi-Way `if-elif-else` Statements

- Nested `if` statements:
  - The inner `if` statement is said to be nested inside the outer `if` statement
  - And even the inner `if` statement can have another `if` statement inside of it
  - In fact, there is no limit to the depth of nesting!
  - Example:

```
if i > k:
    if j > k:
        print("i and j are greater than k")
else:
    print("i is less than or equal to k")
```

Nested `if` and Multi-Way `if-elif-else` Statements

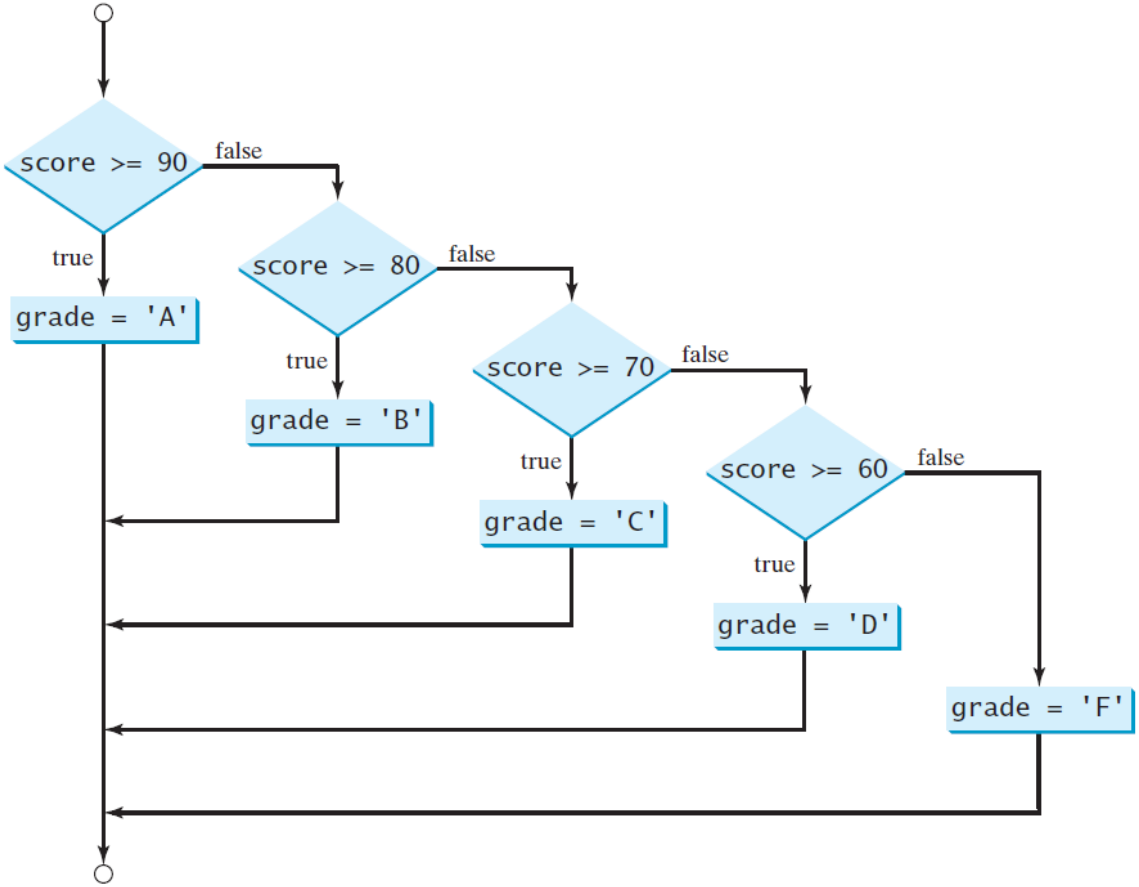
- Nested `if` statements:
  - So how/when is this useful?
    - Answer: whenever we want to implement multiple alternatives
    - Consider the following:

```
if score >= 90.0:
    grade = 'A'
else:
    if score >= 80.0:
        grade = 'B'
  else:
      if score >= 70.0:
          grade = 'C'
      else:
          if score >= 60.0:
              grade = 'D'
          else:
              grade = 'F'
```

- Execution is as follows:
  - The first condition (**score >= 90**) is tested.
    - If it is **True**, the grade becomes A.
  - If **False**, the second condition (**score >= 80**) is tested.
    - If **True**, the grade becomes B.
  - If **False**, the process continues until a condition is met or all of the conditions prove to be False.
  - If all of the conditions are **False**, the grade becomes F.
  - ***Note that a condition is tested only when all of the conditions that come before it are False

Nested `if` and Multi-Way `if-elif-else` Statements

- Control flow of nested `if-else` statements:



Nested `if` and Multi-Way `if-elif-else` Statements

- Nested `if` statements:

```
if score >= 60.0:
    grade = 'D'
else:
    if score >= 70.0:
        grade = 'C'
    else:
        if score >= 80.0:
            grade = 'B'
        else:
            if score >= 90.0:
                grade = 'A'
            else:
                grade = 'F'
```

  - Consider the code above:
    - What is grade when score is 95? Or 75?
      - The assigned grade would be a D…**<u>the code is logically wrong</u>**
      - Because as soon as a score is above (or equal to) a 60, no matter how high that score is, a grade of 'D' is given.

<u>Takeaway</u>:
You must pay attention to the logic of your nested if statements.

Nested `if` and Multi-Way `if-elif-else` Statements

- Multi-way **`if-elif-else`** statements:
  - Too much nesting becomes difficult to read
  - Python solves this with a multi-way if statement!

```
if score >= 90.0:
    grade = 'A'
else:
    if score >= 80.0:
        grade = 'B'
  else:
      if score >= 70.0:
          grade = 'C'
      else:
          if score >= 60.0:
              grade = 'D'
          else:
              grade = 'F'
```

Equivalent

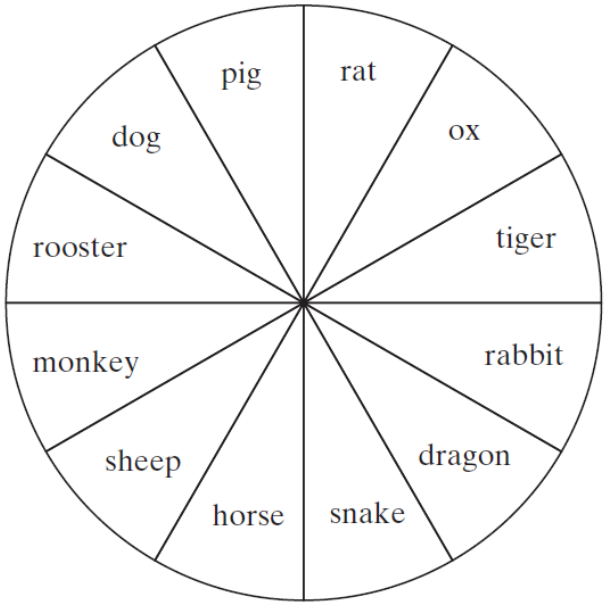
This is better

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

  - The right option is absolutely the preferred choice here
  - `elif` stands for “else if”

Program 5: Chinese Zodiac

- Write a program to find the Chinese zodiac sign for a given year. Ask the user to enter a year and then display the correct Chinese zodiac sign.
- Remember:
  - Step 1: Problem-solving Phase
  - Step 2: Implementation Phase



Program 5: Chinese Zodiac

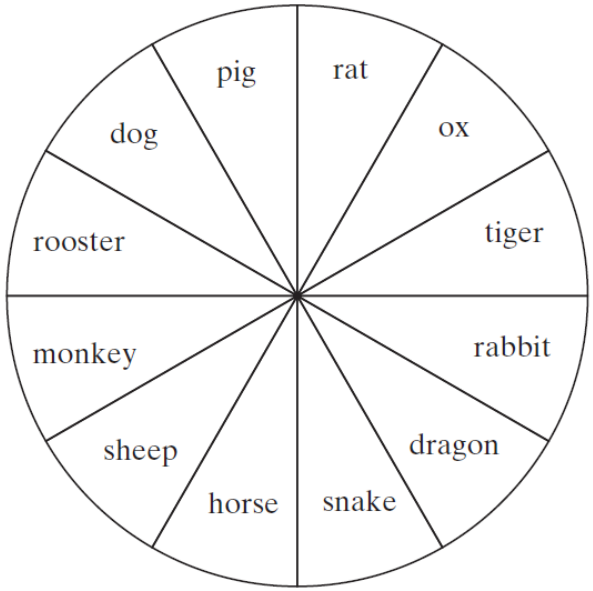
- Write a program to find the Chinese zodiac sign for a given year. Ask the user to enter a year and then display the correct Chinese zodiac sign.
- Step 1: Problem-solving Phase
  - Start by asking the user for any year
    - That's easy
  - But for a given year, 2018 for example, how do we determine the zodiac?
    - Turns out that it's really easy and based on a 12-year cycle
    - Year 0 is the monkey.
    - So when is the monkey again? Year 12, 24, 36, 48, 60, 72, …

Program 5: Chinese Zodiac

- Write a program to find the Chinese zodiac sign for a given year. Ask the user to enter a year and then display the correct Chinese zodiac sign.
- Step 1: Problem-solving Phase
  - But for a given year, 2018 for example, how do we determine the zodiac?
    - Year 1 is the rooster
    - When is the rooster again? Year 13, 25, 37, 49, 61, 73, …
  - So given a year, how can we find the zodiac?
  - We divide it by 12 and take the remainder!
    - And this is precisely what mod (%) gives you!

Program 5: Chinese Zodiac

- Write a program to find the Chinese zodiac sign for a given year. Ask the user to enter a year and then display the correct Chinese zodiac sign.
- Step 1: Problem-solving Phase



`year % 12` =

- 0: monkey
- 1: rooster
- 2: dog
- 3: pig
- 4: rat
- 5: ox
- 6: tiger
- 7: rabbit
- 8: dragon
- 9: snake
- 10: horse
- 11: sheep

Program 5: Chinese Zodiac

- Write a program to find the Chinese zodiac sign for a given year. Ask the user to enter a year and then display the correct Chinese zodiac sign.
- Step 1: Problem-solving Phase
  - So we read a year from the user
  - From that, we calculate the zodiac year
  - The rest is one long `if-elif-else` statement…

Program 5: Chinese Zodiac

- Step 2: Implementation Phase
  - So we read a year from the user
  - From that, we calculate the zodiac year
  - The rest is one long `if-elif-else` statement…

LISTING 4.5 ChineseZodiac.py

```
year = eval(input("Enter a year: "))
zodiacYear = year % 12
if zodiacYear == 0:
    print("monkey")
elif zodiacYear == 1:
    print("rooster")
elif zodiacYear == 2:
    print("dog")
elif zodiacYear == 3:
    print("pig")
elif zodiacYear == 4:
    print("rat")
elif zodiacYear == 5:
    print("ox")
elif zodiacYear == 6:
    print("tiger")
elif zodiacYear == 7:
    print("rabbit")
elif zodiacYear == 8:
    print("dragon")
elif zodiacYear == 9:
    print("snake")
elif zodiacYear == 10:
    print("horse")
else:
    print("sheep")
```

Program 5: Chinese Zodiac

- Step 2: Implementation Phase
  - Result:

```
Enter a year: 1963 ↵Enter
rabbit
```

```
Enter a year: 1877 ↵Enter
ox
```

- Stop here on Friday

Nested `if` and Multi-Way `if-elif-else` Statements

Check Yourself

- Suppose **x = 3** and **y = 2**; show the output, if any, of the following code.

```
if x > 2:
    if y > 2:
        z = x + y
        print("z is", z)
    else:
        print("x is", x)
```

Output:

```
x is 3
```

- What is the output if **x = 3** and **y = 4**?

```
Z is 7
```

- What is the output if **x = 2** and **y = 2**?

```
No output
```

Common Errors in Selection Statements

- Common errors usually differ with the language
- With Python, the main error is indentation!
- Remember:
  - Indentation is central to Python!
  - Python "understands" your code based off of its indentation
  - If you want a segment of code to be executed only under an `if` condition (or only under an `else` condition), then it must all be indented together as a group.

Common Errors in Selection Statements

- With Python, the main error is indentation!
- Consider the following example:

```
radius = -20

if radius >= 0:
    area = radius * radius * math.pi
print("The area is", area)
```

(a) Wrong

```
radius = -20

if radius >= 0:
    area = radius * radius * math.pi
    print("The area is", area)
```

(b) Correct

  - Why is (a) wrong?
    - Clearly, we only want to print if the radius is greater or equal to zero
    - This means the print statement must be in the `if` block
    - Which means it must be indented!

Common Errors in Selection Statements

- With Python, the main error is indentation!
- Consider another example:

```
i = 1
j = 2
k = 3

if i > j:
    if i > k:
        print('A')
else:
    print('B')
```

(a)

```
i = 1
j = 2
k = 3

if i > j:
    if i > k:
        print('A')
    else:
        print('B')
```

(b)

  - There are two `if` statements and one `else` statement
  - Which `if` clause is matched by the `else` statement?
    - This is FULLY determined based off of the indentation
    - In (a), the `else` is matched to the first `if` clause
    - In (b), the `else` is matched with the nested `if` clause

Program 6: Rock, Paper, Scissors

- Write a program to play the famous Rock, Paper, Scissors game with the computer. Your program should prompt the user to enter a choice for rock, paper, or scissors. The computer will then randomly choose an option and a winner will be determined.

```
scissor (0), rock (1), paper (2): 1 ↵Enter
The computer is scissor. You are rock. You won.
```

```
scissor (0), rock (1), paper (2): 2 ↵Enter
The computer is paper. You are paper too. It is a draw.
```

Program 6: Rock, Paper, Scissors

- Step 1: Problem-solving Phase
  - We need to ask the user for a choice
    - They could enter 0, 1, or 2 (for rock, paper or scissors) as shown on the last slide
    - Or they could actually enter the word “rock”, “paper” or “scissors”
      - Only issue with that is we’re now dealing with spelling issues and keyboard mistakes if we go that route
      - So for now, let’s stick with the 0, 1, or 2
    - The computer then randomly generates a choice
      - How many choices? 3 of them. Specifically, 0, 1, or 2
      - How do we do this?
      - `choice_computer = random.randint(0, 2)`
    - Use nested `if` and multi-way `if-elif-else` to solve!

Program 6: Rock, Paper, Scissors

- Step 2: Implementation Phase
  - See Portal for a sample solution!

Common Errors in Selection Statements

- Tip:
  - New programmers often write code that assigns a test condition to a Boolean variable as shown below:

```
if number % 2 == 0:
    even = True
else:
    even = False
```

  - This code is not wrong.
  - But, it can be simplified into one line:

```
even = number % 2 == 0
```

    - Here, the result of `(number % 2 == 0)` is assigned directly into `even`

Common Errors in Selection Statements

Check Yourself

- Which of the following statements are equivalent? Which are properly indented?

```
if i > 0:
    x = 0
    y = 1
else:
    y = 0
    z = 0
```

(a)

```
if i > 0:
    x = 0
  y = 1
else:
    y = 0
    z = 0
```

(b)

```
if i > 0:
  x = 0
  y = 1
else:
    y = 0
    z = 0
```

(c)

```
if i > 0:
  x = 0
  y = 1
else:
    y = 0
  z = 0
```

(d)

- Answer:
  - (A) and (C) are equivalent. (actually, they are the same!)
  - (B) and (D) are incorrectly indented.

Common Errors in Selection Statements

Check Yourself

- Are the following statements correct? Which is better?

```
if age < 16:
    print("Cannot get a driver's license")
if age >= 16:
    print("Can get a driver's license")
```

(a)

```
if age < 16:
    print("Cannot get a driver's license")
else:
    print("Can get a driver's license")
```

(b)

- Answer:
  - Both are correct
  - Option (b) is better because only one condition is tested/used

Common Errors in Selection Statements

Check Yourself

- What is the output of the following code if **number** is **14**, **15**, and **30**?

```
if number % 2 == 0:
    print(number, "is even")
if number % 5 == 0:
    print(number, "is multiple of 5")
```

(a)

```
if number % 2 == 0:
    print(number, "is even")
elif number % 5 == 0:
    print(number, "is multiple of 5")
```

(b)

- Answer:
  - Number is 14: (a) displays “14 is even” (b) displays “14 is even”
  - Number is 15: both (a) and (b) display “15 is a multiple of 5”
  - Number is 30: (a) displays “30 is even” <u>and</u> “30 is a multiple of 5”, while (b) displays “30 is even”

Common Errors in Selection Statements

Check Yourself

- Rewrite the following statement using a Boolean expression:

```
if count % 10 == 0:
    newLine = True
else:
    newLine = False
```

- Answer:

```
newline = count % 10 == 0
```

- Start here

Logical Operators

- We've seen how to use conditions to determine whether a statement should be executed
  - Use a Boolean expression with an if or if/else statement
- Sometimes, a statement should only be executed if multiple conditions are true
- And for this, we have **logical operators**
  - aka **Boolean operators**
  - `not`, `and`, and `or`
- The logical operators `not`, `and`, and `or` can be used to create a composite condition.

Logical Operators

- The following table simply lists the operators

TABLE 4.3 Boolean Operators

Operator	*Description*
`not`	logical negation
`and`	logical conjunction
`or`	logical disjunction

- What remains is to understand how each operator works
  - And for this, we introduce you to truth tables…

Logical Operators

- Truth Tables:
  - Truth tables are used to show the "truth values" of Boolean expressions
  - For example, the Truth Table below shows the truth values of the expression "`not p`"
    - This assumes a Boolean variable `p` already exists
    - And of course, `p` can be one of two values: `true` or `false`

TABLE 4.4 Truth Table for Operator `not`

p	not p	*Example (assume* `age = 24, gender = 'F'`*)*
`True`	`False`	`not (age > 18)` is `False`, because `(age > 18)` is `True`.
`False`	`True`	`not (gender == 'M')` is `True`, because `(gender == 'M')` is `False`.

Logical Operators

TABLE 4.5 Truth Table for Operator `and`

p_1	p_2	p_1 and p_2	*Example (assume* `age = 24, gender = 'F'`*)*
`False`	`False`	`False`	`(age > 18) and (gender == 'F')` is `True`, because `(age > 18)` and `(gender == 'F')` are both `True`.
`False`	`True`	`False`	
`True`	`False`	`False`	`(age > 18) and (gender != 'F')` is `False`, because `(gender != 'F')` is `False`.
`True`	`True`	`True`	

TABLE 4.6 Truth Table for Operator `or`

p_1	p_2	p_1 and p_2	*Example (assume* `age = 24, gender = 'F'`*)*
`False`	`False`	`False`	`(age > 34) or (gender == 'F')` is `True`, because `(gender == 'F')` is `True`.
`False`	`True`	`True`	
`True`	`False`	`True`	`(age > 34) or (gender == 'M')` is `False`, because `(age > 34)` and `(gender == 'M')` are both `False`.
`True`	`True`	`True`	

Program 7: Test Boolean Operators

- Write a program to check whether a number is divisible by **`2 and 3`**, by **`2 or 3`**, and by **`2 or 3 but not both`**.
- We start by examining the output:

```
Enter an integer: 18 ↵Enter
18 is divisible by 2 and 3
18 is divisible by 2 or 3
```

```
Enter an integer: 15 ↵Enter
15 is divisible by 2 or 3
15 is divisible by 2 or 3, but not both
```

Program 7: Test Boolean Operators

- Write a program to check whether a number is divisible by **`2 and 3`**, by **`2 or 3`**, and by **`2 or 3 but not both`**.
- Remember:
  - Step 1: Problem-solving Phase
    - When we examine the input, we can see that the requested three conditions are explicitly being tested.
    - So we have three if statements
    - The first two are mostly straightforward, albeit new
    - The third if statement is an even larger Boolean expression…

Program 7: Test Boolean Operators

- Step 2: Implementation Phase

```
# Receive an input
number = eval(input("Enter an integer: "))

# Check if number is divisible by BOTH 2 and 3
if number % 2 == 0 and number % 3 == 0:
    print(number, "is divisible by 2 and 3")

# Check if number is divisible by 2 or 3
if number % 2 == 0 or number % 3 == 0:
    print(number, "is divisible by 2 or 3")

# Check if number is divisible by 2 or 3, but not both of them
if (number % 2 == 0 or number % 3 == 0) and not (number % 2 == 0 and number % 3 == 0):
    print(number, "is divisible by 2 or 3, but not both")

# we could have written the last if statement as follows:
if (number % 2 == 0 or number % 3 == 0) and (number % 2 != 0 or number % 3 != 0):
    print(number, "is divisible by 2 or 3, but not both")
```

Program 7: Test Boolean Operators

- Step 2: Implementation Phase
  - De Morgan's Law:
    - A famous law that can be used to simply Boolean expressions:

      `not (condition1 and condition2)` is the same as

      `not condition1 or not condition2`

      Also:

      `not (condition1 or condition2)` is the same as

      `not condition1 and not condition2`

    - In the last program, we had this line:

      ```
      not (number % 2 == 0 number % 3 == 0)
      ```

    - That could be rewritten as follows:

      ```
      (number % 2 != 0 or number % 3 != 0)
      ```

Logical Operators

Check Yourself

- Assuming that x is 1, show the result of the following Boolean expressions:

```
True and (3 > 4)                    False
not (x > 0) and (x > 0)             False
(x > 0) or (x < 0)                  True
(x != 0) or (x == 0)                True
(x >= 0) or (x < 0)                 True
(x != 1) == not (x == 1)            True
```

Logical Operators

Check Yourself

- Write a Boolean expression that evaluates to `True` if variable `num` is between 1 and 100.
- Answer:

```
(num > 1) and (num < 100)
```

- Write a Boolean expression that evaluates to `True` if variable `num` is between 1 and 100 or the number is negative.
- Answer:

```
((num > 1) and (num < 100)) or (num < 0)
```

Logical Operators

Check Yourself

- Write a Boolean expression that evaluates true if weight is greater than 50 or height is greater than 160.
- Answer:

```
weight > 50 or height > 160
```

- Write a Boolean expression that evaluates true if either weight is greater than 50 or height is greater than 160, but not both.
- Answer:

```
(weight > 50 or height > 160) and not (weight > 50 and height > 160)
```

- Friday, do Leap year and conditional expression

Program 8: Determining Leap Years

- Write a program to check whether a given year is a leap year.
- Remember:
  - Step 1: Problem-solving Phase
    - Leap year comes every four years
    - But how do we know which year is officially a leap yaer
    - It's actually mathematically based.
    - Here is the rule
      - A year is a leap year if it is divisible by 4 but not by 100 or if it is divisible by 400.

Program 8: Determining Leap Years

- Write a program to check whether a given year is a leap year.
- Remember:
  - Step 1: Problem-solving Phase
    - Here is the rule
      - A year is a leap year if it is divisible by 4 but not by 100 or if it is divisible by 400.
    - So we can build this out in steps…

```
is_leap_year = (year % 4 == 0)
```

    - Now add the second part of the condition

```
is_leap_year = (year % 4 == 0 and year % 100 != 0)
```

    - And finally, the last condition:

```
is_leap_year = (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
```

Program 8: Determining Leap Years

- Write a program to check whether a given year is a leap year.
- Remember:
  - Step 2: Implementation Phase

```
# Get input from user
year = eval(input("Enter a year: "))

# Check if the year is a leap year
is_leap_year = (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

# Display the result
if is_leap_year:
    print(year, "is a leap year.")
else:
    print(year, "is not a leap year.")
```

Program 9: Lottery

- Write a program to play a simple lottery game
  - Your program should randomly generate a two-digit number
  - You should prompt the user to enter a two digit number
  - Determine winnings based on the following:
    - If the user's input matches the lottery in the exact order, the award is $10,000.
    - If all the digits in the user's input match all the digits in the lottery number (but not in the correct order), the award is $3,000.
    - Finally, if only one digit in the user's input matches a digit in the lottery number, the award is $1,000.

Program 9: Lottery

- Write a program to play a simple lottery game
- Remember:
  - Step 1: Problem-solving Phase
    - Checking if the guess matches the random number is easy
    - But what about checking the individual digits…
      - This requires some thought
    - Example: imagine we have guess = 57
      - How can we isolate the 5 and the 7
      - Meaning, we want to have two new variables:
        - guess_digit1 = 5
        - guess_digit2 = 7
      - How can we do that? Answer: integer division and mod!!!
        - 57 // 10 = 5
        - 57 % 10 = 7

```
      5
10 ) 57
    -50
      7
```

Program 9: Lottery

- Write a program to play a simple lottery game
- Remember:
  - Step 1: Problem-solving Phase
    - So we get the digits for both numbers
      - The guess
      - And the randomly generated number
    - We then have three checks
      - IF the original numbers are identical (thus, same order)
        - Winnings are $10,000
      - ELIF both digits match, but now not in order
        - So guess_digit1 == random_digit2 and vice versa
        - Winnings are $3,000
      - ELIF only one of the digits match
        - Winnings are $1000

Program 9: Lottery

- Write a program to play a simple lottery game
- Step 2: Implementation Phase
  - Here's the expected run/output of the program:

```
Enter your lottery pick (two digits): 45 ↵Enter
The lottery number is 12
Sorry, no match
```

```
Enter your lottery pick (two digits): 23 ↵Enter
The lottery number is 34
Match one digit: you win $1,000
```

Program 9: Lottery

- Write a program to play a simple lottery game
- Step 2: Implementation Phase

```
import random

# Generate a lottery
rand_num = random.randint(0, 99)

# Prompt the user to enter a guess
guess = eval(input("Enter your lottery pick (two digits): "))

# Get digits from lottery
rand_num_digit1 = lottery // 10
rand_num_digit2 = lottery % 10

# Get digits from guess
guess_digit1 = guess // 10
guess_digit2 = guess % 10

print("The lottery number is", lottery)
```

Program 9: Lottery

- Write a program to play a simple lottery game
- Step 2: Implementation Phase

```
# Check the guess
if guess == lottery:
    print("Exact match: you win $10,000")

# Check if both digits match but in wrong order
elif (guess_digit2 == rand_num_digit1 and guess_digit1 == rand_num_digit2):
    print("Match all digits: you win $3,000")

# Check if only one of the digits match
elif (guess_digit1 == rand_num_digit1
        or guess_digit1 == rand_num_digit2
        or guess_digit2 == rand_num_digit1
        or guess_digit2 == rand_num_digit2):
    print("Match one digit: you win $1,000")

else:
    print("Sorry, no match")
```

Conditional Expressions

- Python also has something cool for conditions
  - called Conditional Expressions
  - The idea is to evaluate an expression based on a condition
  - Consider the following:

```
if x > 0:
    y = 1
else:
    y = -1
```

  - Python can do this in one line:

```
y = 1 if x > 0 else -1
```

Conditional Expressions

- Python also has something cool for conditions
  - The overall syntax:

```
expression1 if boolean-expression else expression2
```

  - Example:
    - Suppose you have two numbers: number1 and number2
    - And you want to assign the larger of those into “max”
    - You could do this with an if/else statement as on the last slide
    - Or we can use a conditional expression as follows:

```
max = number1 if number1 > number2 else number2
```

  - Note:
    - This accomplishes the same goal as an if/else
    - It is not faster, and some find the if/else simply easier to read

Operator Precedence and Associativity

- We've now learned many operators
  - Time for an updated Precedence Chart!

TABLE 4.7 Operator Precedence Chart

Precedence	*Operator*
↓	+, - (Unary plus and minus)
	** (Exponentiation)
	not
	*, /, //, % (Multiplication, division, integer division, and remainder)
	+, - (Binary addition and subtraction)
	<, <=, >, >= (Comparison)
	==, != (Equality)
	and
	or
	=, +=, -=, *=, /=, //=, %= (Assignment operators)

Operator Precedence and Associativity

Check Yourself

- Evaluate the following:

```
True or True and False
True and True or False
```

- And another:

```
2 * 2 - 3 > 2 and 4 - 2 > 5
2 * 2 - 3 > 2 or 4 - 2 > 5
```

PYTHON BOOT CAMP

Module 4: Selections