Math 2300: Discrete Mathematics
Introduction to Python 2 - If, if else, and while Textbook Section

Terms, Concepts, and Examples

We want to consider statements that will involve tests or conditions in order to execute.

- Python can skip over a **block** of lines using an **if statement** and a Boolean expression. Blocks always start with a colon : on the previous line and require every line to be indented.

  The general syntax for an if statement is

```
if condition:
        indentedStatementBlock
```

  Example: Try putting this code in Python tutor and change the values of w.

```
w = # an int value
if w >50:
        print("There is a $25 charge for luggage that heavy.")
print("Thank you for your business")
```

  Solution: It reads pretty much like English. If it is true that the weight is greater than 50, then print the statement about an extra charge. If it is not true that the weight is greater than 50, then don't do the indented part: skip printing the extra luggage charge.

- When you want to force exactly one of two blocks (as opposed to just skipping a block) you can use an **if else statement**.

  The general syntax for an if else statement is

```
if condition:
        indentedStatementBlockforTrue
else:
        indentedStatementBlockforFalse
```

  Example: Try putting this code in Python tutor and change the values of t.

```
t = # an int value
if t >80:
        print("Wear shorts.")
else:
        print("Wear long pants.")
print("Get some exercise outside.")
```

  Solution: If the value you put in for t is bigger than 80 the code will print out "Wear shorts." Then on the next line, print "Get some exercise outside." If the value for t is less than or equal to 80, it will print "Wear long pants." and on the next line "Get some exercise outside." Since the last print statement is not part of the if else block, it prints in both cases.

  Video Example of If and If-Else

- When you want to execute a block of lines multiple times you can use an **while statement** and a Boolean expression. The block will repeat until the Boolean expression is false.

  The general syntax for a while statement is

```
while condition:
        indentedBlock
```

  The indentedBlock often contains a main action to be repeated and a line to prepare the variables for the next time through the loop.

  Example: Try putting this code in Python tutor.

```
temperature = 115
while temperature > 112: # first while loop code
    print(temperature, "Too hot! Add an ice cube.")
    temperature = temperature - 1
print('The tea is cool enough.')
```

  Solution: It reads in English as "While the tea is too hot, add an ice cube." You test your tea, if it is true that the tea is too hot, you add ice. Each time you add ice, you can imagine the temperature goes down a degree. As long as you test and find out the tea is too hot, you keep adding ice.

  Video Example of While

Practice Problems

1.
```
score = # an int value from 0 to 100
if score >= 90:
        letter = 'A'
else:   # grade must be B, C, D or F
      if score >= 80:
            letter = 'B'
      else:  # grade must be C, D or F
            if score >= 70:
                letter = 'C'
            else:    # grade must D or F
                if score >= 60:
                    letter = 'D'
                else:
                    letter = 'F'
print letter
```

   Determine the value of the variable letter when score is 92, 84 and 59.

2.
```
if outside == False:
        if (n >= 1 and n <= 10):
```

```
            ans = True
        else:
            ans = False
else:
        if (n <= 1 or n >= 10):
            ans = True
        else:
            ans = False
print(ans)
```

Determine the value of the variable ans in the following cases:

(a) n=5, outside = False

(b) n=11, outside = False

(c) n=11, outside = True

(d) n=9, outside = True

3.
```
count = 5
while count >0:
        print("Welcome")
        count -= 1
```

What will this code print out?

4. Write Python code to satisfy the following conditions. Then test your code on the values of the variables given.

   (a) Given an int n, return the absolute difference between n and 21, except return double the absolute difference if n is over 21.

   It should return 2 when n=19. It should return 11 when n=10. What will the code return when n=21 or n=25?

   (b) We have a loud talking parrot. The "hour" parameter is the current hour time in the range 0..23. We are in trouble if the parrot is talking and the hour is before 7 or after 20. Return True if we are in trouble.

   It should return True when the parrot is talking and the hour is 6. It should return False when the parrot is not talking and the hour is 6. What does it return if the parrot is talking and the hour is 7?