Session 3: Loops, Conditionals, and Sequential Thinking

3.1 Counting

For loops allow us to repeat things.

Start with counting from 1 to 3:

```
1, 2, 3.
```

We can think of this counting using:

```
start = 1    <- First number
step  = 1    <- Add 1 to previous number
end   = 3    <- Do not count past 3
```

We can also count by 2s.

To count:

```
5, 7, 9
```

we have:

```
start = 5
step  = 2
end   = 9.
```

In Python, we also add the stop index. The idea is to not count past the stop.

For counting 1, 2, 3:

```
stop = 4  <- Do NOT include 4 in your counting.
```

For counting 5, 7, 9:

```
stop = 11 <- Do NOT include 11.
```

In Mathematics, you may have also covered this material under *arithmetic progressions*.

3.2 Counting with for loops

In [ ]:
```
# Increasing a variable example
# - Change the line i=3 and see what happens.
# - Change the 2 and see what happens.

i = 3
print(i)

# Note special example:
#     i = i + 2
# In regular algebra, this is IMPOSSIBLE. Why?
# However, the computer will process this fine!
#  1. It processes the right-hand-side first.
#  2. Then it assigns the result to i (the left-hand-side)
# What happened here?

i = i + 2
print(i)
```

In [ ]:
```
# The following example shows repearing code.
# Can you see which code is repeated?

# Work through this code line-by-line
# - Change start_i to see what happens.
# - Change step_i  to see what happens.
# - Add comments to document your understanding.

start_i = 1
step_i  = 1

i = start_i
print(i)

i = i + step_i # increases i
print(i)

i = i + step_i # increases i
print(i)
```

In [ ]:
```
# You can use a loop to avoid retyping code that repeats.
# The code below does the same thing using a for-loop.
# Note the special syntax:
# - NO empty space at the begining of a line.
# - Note the range() command that generates 1, 2, 3.
# - The for-loop ends with  :
# - After the for-loop, you must have some leading space for the repeating cod
e.
# - ALL of the code INSIDE the for-loop must START at the same position.

start_i = 1
step_i  = 1

# Stop at 4.
stop_i  = 4

for i in range(start_i, stop_i, step_i):
    print(i)
    # The COMMENT starts at the same position as print(i)
    # Everything inside the loop starts at the SAME position.
```

In [ ]:
```
# Write the code to count from 1 to 10
# Your code should output: 1, 2, 3, ..., 10.
# Copy and paste the code from above and be careful about the syntax!
```

In [ ]:
```
# Write the code to count from 2 to 20 by 2
# Your code should ouput:  2, 4, 6, ..., 20.
```

In [ ]:
```
# Create your own loop to count as far as you want and by the step you want.
```

3.2 Conditional statements and inequalities

3.2.1 Conditions using equalities and inequalities

Consider the numbers 1 to 10.

Which numbers satisfy $x > 5$?

Which numbers satisfy $x < 5$?

In computing, $(x < 5)$ and $(x > 5)$ are called conditions :-).

Another condition is $(x == 5)$ which is only satisfied if $x = 5$.

3.2.2 Basic conditional statements

In [ ]:

```
# Guess my number game
# Add comments to document your understanding.
# - Run the code and enter 6. Observe what happens.
# - Run the code and enter 5. Observe what happens.

# Note that int() converts it to an integer.
# int(1.1) gives 1 so that spaces are removed :-).
number = int(input(" Input your number "))
secret_number = 5
if (number == secret_number):
    print("You guessed it!")
    print("You win.")
else:
    print("You did not guess it correctly!")
    print("I win!")

print("End of the game!")
```

We want to program computers to execute different code based on their input.

To make the computers do that, we use the if-statement.

Consider the following code:

```
Code1
if (condition)
   Code2a
else:
   Code2b
Code3
```

Two different cases need to be considered.

Based on the example above, add comments that identify:

```
Code1, Code2a, Code2b, and Code3
```

What was the condition that was checked?

Study the code above and make the connections to the following cases:

Case 1. Condition is satisfied after Code1 is executed.
Computer executes:

```
Code1
Evaluate condition
Code2a
Code3
```

Case 2. Condition is NOT satisfied after Code1 is done.
Computer executes:

```
Code1
Evaluate condition
Code2b
Code3
```

3.2.3 Complex conditional statements

In [ ]:

```
# Run the code below by entering:
#   6, 4, and 5.
# Carefully observe which message is printed.
#
# Change the 3 messages to help the user.

number = int(input(" Input your number "))
secret_number = 5

if (secret_number > number):
    print("Your guess is ... I") # FIX the message
elif (secret_number < number):
    print("Your guess is ... II") # FIX the message
else:
    print("Your guess is ... III") # FIX the message

print(" ")
print("End of game")
```

A more complicated conditional statement is:

```
Code1
if (condition 1)
  Code2a
elif (condition 2)
  Code2b
else
  Code2c
Code3
```

We need to consider three cases.

Look at the code above and identify:

```
 Code1, Code2a, Code2b, Code2c, and Code 3.
```

What are conditions 1, 2a, 2b, 2c, and 3?
Which numbers satisfy conditions 1, 2, and 3?

We have the following execution cases.

Can you identify the following cases?

Case 1. If condition 1 is satisfied after Code1.</br> Computer executes:

```
Code1
Evaluate condition 1
Code2a
Code3
```

Case 2. If condition is is NOT satisfied after Code1.</br> The computer will then check condition 2. Assume that condition 2 is satisfied.
Computer executes:

```
Code1
Evaluate condition 1
Evaluate condition 2
Code2b
Code3
```

Case 3. Neither condition 1 nor 2 is satisfied.</br> Computer executes:

```
Code1
Evaluate condition 1
Evaluate condition 2
Code2c
Code3
```

We will try to demonstrate this below.

3.3 Guess a random number game

3.3.1 Random numbers

In [ ]:
```
# Get the Random library
# Note that all computer games relie on random numbers!
# Can you see how?
# Run the code multiple times
# What do you observe?
from random import *
print(randint(1, 10)) # Pick a random number between 1 and 10.
```

3.3.2 A while loop that can guess a random number

In [ ]:
```
# Examine the code:

from random import *
secret_number = randint(1, 10) # generate a random number between 1 and 10

computer_guess_number = 1
while (computer_guess_number != secret_number):
    print("I tried "+str(computer_guess_number))
    computer_guess_number = computer_guess_number + 1

print("I guessed it!")
print("The random number is "+str(computer_guess_number))
```

While loops repeat code until a condition is met.

Their format is:

```
Code1
while (condition):
    Code2
Code3
```

Identify Code1, Code2, and Code3 in the example above.

We consider two cases.

Case 1. The condition is satisfied after Code1.
The computer executes:

```
Code1
Evaluate condition
Code2
Evaluate condition
Code2
:
... until the condition FAILS
Code3
```

Case 2. The condition is NOT satisfied after Code1.
The computer simply executes:

```
Code1
Evaluate condition
Code3
```

Main Activitity: Guess a random number with hints

Write a game that asks the user to guess a random number.
Then, give the user a hint if the number is high or low.
When the right number is found, print a success message.