CS 106A: Variables Review

Wednesday, May 13
Today...
Today...

1. What is a Variable?
2. Arithmetic On Variables
3. Variables and Control Flow
4. Diagnostic Problem 4, Redux
What is a Variable?

Suitcases and Luggage Tags
What is a Variable?

Variables are like **baggage tags** that attach to **suitcases**.
What is a Variable?

Variables are like **baggage tags** that attach to **suitcases**.

Doesn’t assign any baggage tags, but *does* use suitcases:

```python
print(106)
```
What is a Variable?

Variables are like **baggage tags** that attach to **suitcases**.

Doesn’t assign any baggage tags, but does use suitcases:

```
   print(106)
```
What is a Variable?

Variables are like **baggage tags** that attach to **suitcases**.

 Doesn’t assign any baggage tags, but *does* use suitcases:

```
print(106)
```
What is a Variable?

Variables are like **baggage tags** that attach to **suitcases**.

Doesn’t assign any baggage tags, but *does* use suitcases:

```python
print(106)
```

Creates a suitcase and assigns a baggage tag to it:

```python
class_num = 106
print(class_num)
```
What is a Variable?

Variables are like **baggage tags** that attach to **suitcases**.

Doesn’t assign any baggage tags, but *does* use suitcases:

```
print(106)
```

Creates a suitcase and assigns a baggage tag to it:

```
class_num = 106
print(class_num)
```
What is a Variable?

Variables are like **baggage tags** that attach to **suitcases**.

Doesn’t assign any baggage tags, but does use suitcases:

```python
print(106)
```

Creates a suitcase and assigns a baggage tag to it:

```python
class_num = 106
print(class_num)
```
What is a Variable?

Variables are like **baggage tags** that attach to **suitcases**.

Doesn’t assign any baggage tags, but *does* use suitcases:

```
print(106)
```

Creates a suitcase and assigns a baggage tag to it:

```
class_num = 106
print(class_num)
```
Okay, How Do I Use Them?

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.
Okay, How Do I Use Them?

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
name = input("What is your name? ")
print("Hiya, " + name + "! I'm Python.")
```
Okay, How Do I Use Them?

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
name = input("What is your name? ")
print("Hiya, " + name + "! I'm Python.")
```

We need to keep track of what the user enters to use it in the greeting.
Okay, How Do I Use Them?

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
name = input("What is your name? ")
print("Hiya, " + name + "! I'm Python.")
```

We need to keep track of what the user enters to use it in the greeting.
Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
name = input("What is your name? ")
print("Hiya, " + name + "! I'm Python.")
```

This is the problem that variables solve: they allow you to keep track of important data throughout the execution of your program.

We need to keep track of what the user enters to use it in the greeting.
Okay, How Do I Use Them?

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
name = input("What is your name? ")
print("Hiya, " + name + "! I'm Python.")
```

This is the problem that variables solve: they allow you to keep track of important data throughout the execution of your program.

We need to keep track of what the user enters to use it in the greeting.
Okay, How Do I Use Them?

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
name = input("What is your name? ")
print("Hiya, " + name + "! I'm Python.")
```

This is the problem that variables solve: they allow you to keep track of important data throughout the execution of your program.

We need to keep track of what the user enters to use it in the greeting.
Another Use: Modifying Variables

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.
Another Use: Modifying Variables

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
pixel = image.get_pixel(x, y)
pixel.red = 255
pixel.green = 0
pixel.blue = 0
```
Another Use: Modifying Variables

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
pixel = image.get_pixel(x, y)
pixel.red = 255
pixel.green = 0
pixel.blue = 0
```

We need to hold on to the pixel...
Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
pixel = image.get_pixel(x, y)
pixel.red = 255
pixel.green = 0
pixel.blue = 0
```

We need to hold on to the pixel...

...so that we can modify its attributes here.
Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
pixel = image.get_pixel(x, y)
pixel.red = 255
pixel.green = 0
pixel.blue = 0
```

Another Use: Modifying Variables

We need to hold on to the pixel...

...so that we can modify its attributes here.
Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
pixel = image.get_pixel(x, y)
pixel.red = 255
pixel.green = 0
pixel.blue = 0
```

We need to hold on to the pixel...

...so that we can modify its attributes here.
Another Use: Modifying Variables

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```
pixel = image.get_pixel(x, y)
pixel.red = 255
pixel.green = 0
pixel.blue = 0
```

We need to hold on to the pixel...

...so that we can modify its attributes here.
Another Use: Modifying Variables

Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```
pixel = image.get_pixel(x, y)
pixel.red = 255
pixel.green = 0
pixel.blue = 0
```

We need to hold on to the pixel...

...so that we can modify its attributes here.
Use a variable to keep track of important data in your program that you’ll need to refer to in the future.

```python
pixel = image.get_pixel(x, y)
pixel.red = 255
pixel.green = 0
pixel.blue = 0
```

We need to hold on to the pixel...

...so that we can modify its attributes here.
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

Scope

What happens?
parth = 'a unicorn'

def awesome():
kara = 'awesome'
print('awesome() says kara is ' + kara)
print('awesome() says parth is ' + parth)

def main():
kara = 'super cool'
print('main() says kara is ' + kara)
print('main() says parth is ' + parth)

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

```python
def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
```
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

```python
def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
```

What happens?

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
kara = 'awesome'

print('awesome() says kara is ' + kara)
print('awesome() says parth is ' + parth)

def main():
kara = 'super cool'

print('main() says kara is ' + kara)

print('main() says parth is ' + parth)

awesome()

print('main() says kara is ' + kara)

print('main() says parth is ' + parth)

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

What happens?

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

What happens?

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

What happens?

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

What happens?

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()  # Print statements work only within this function
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

What happens:
main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

What happens?

main() says kara is super cool
main() says parth is a unicorn
awesome() says kara is awesome
awesome() says parth is a unicorn
main() says kara is super cool
main() says parth is a unicorn
parth = 'a unicorn'

```python
def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
```

What happens?

- `main() says kara is super cool`
- `main() says parth is a unicorn`
- `awesome() says kara is awesome`
- `awesome() says parth is a unicorn`
- `main() says kara is super cool`
- `main() says parth is a unicorn`
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

main()
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

**def awesome():**
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

**def main():**
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'
kara = 'fabulous'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
parth = 'a unicorn'
kara = 'fabulous'

def awesome():
    kara = 'awesome'
    print('awesome() says kara is ' + kara)
    print('awesome() says parth is ' + parth)

def main():
    kara = 'super cool'
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)
    awesome()
    print('main() says kara is ' + kara)
    print('main() says parth is ' + parth)

What happens if we define a variable called kara at the top level?

Nothing! Functions always check inside their own scope before they check the global scope.
1. Variables! What are they good for?
   
   The broad answer: **keeping track of data** (you might change the data, use the data, some combination of both, etc.)
1. Variables! What are they good for?
   The broad answer: **keeping track of data** (you might change the data, use the data, some combination of both, etc.)

2. Functions *don’t* share variables.
   See our review session on Functions, Parameters & References Review to see how we can make functions share data and talk to each other.
1. Variables! What are they good for?
   The broad answer: keeping track of data (you might change the data, use the data, some combination of both, etc.)

2. Functions don’t share variables.
   See our review session on Functions, Parameters & References Review to see how we can make functions share data and talk to each other.

3. When you reference a variable, functions will resolve them in a specific order.
   First it checks locally, then globally.
Arithmetic On Variables
You can do some cool stuff with variables!

num = 1
num *= 2
num = num*2
num += 1
num = num+1
num **= 3
num = num**3

multiplies num by 2 and reassigns num to be that product

increments num by 1 and updates num to be the new value
takes the value of num, puts it to the third power and stores the value in num
You can do some cool stuff with strings!

- You can do "multiplication" and "addition" on string variables
- "multiplication" will just multiply the copies of the word you have
- "addition" will add some new characters to the end of your string (this is also called concatenation)

```python
name1 = "kara"
name2 = "parth"
name1 *= 2 -> "karakara"
names_together = name1+ name2 -> "karakaraparth"
```
# Returns the square root of a number

```ruby
math.sqrt(2)  # => 1.4142135623730951
```
# Returns the square root of a number
math.sqrt(2) # => 1.4142135623730951

# Returns a random integer from range(start, stop)
random.randrange(start, stop)
# Returns the square root of a number
math.sqrt(2) # => 1.4142135623730951

# Returns a random integer from range(start, stop)
random.randrange(start, stop)

# Returns a random float between start and stop
random.uniform(start, stop)
Variables and Control Flow
Using variables with if and while

- Recall that boolean expressions will evaluate to either True or False
  - $1 < 2 \rightarrow True$
  - `'abc' == 'def' \rightarrow False$
- You can use the values stored in your variables in these boolean expressions! This lets you make your code more generalizable
  - if num1 < num2:
    #code
Let's see a program!

Goal: we want to simulate a person flipping a coin. They want to get 10 heads before they stop. How can we keep track of how many tosses that takes?
# A program that tosses a coin until we get 10 heads and then prints how many flips it took to get those 10 heads

def simulate_coin_toss():
    num_heads = 0
    num_tosses = 0

    while (num_heads < 10):
        toss = random.randrange(0, 2)
        num_tosses += 1
        if toss == 0:
            num_heads += 1

    print("It took " + str(num_tosses) + " to get 10 heads")
# A program that tosses a coin until we get 10 heads and then prints how many flips it took to get those 10 heads

def simulate_coin_toss():
    num_heads = 0
    num_tosses = 0
    while (num_heads < 10):
        toss = random.randint(0,2)
        num_tosses += 1
        if toss == 0:
            num_heads += 1
    print("It took " + str(num_tosses) + " to get 10 heads")

# A variable that keeps track of how many coin tosses there have been
# A program that tosses a coin until we get 10 heads and then prints how many flips it took to get those 10 heads

def simulate_coin_toss():
    num_heads = 0
    num_tosses = 0
    while (num_heads < 10):
        toss = random.randint(0, 2)
        num_tosses += 1
        if toss == 0:
            num_heads += 1
    print("It took " + str(num_tosses) + " to get 10 heads")

    checks to see if we've landed on enough heads yet, if not, loop again!
Using variables in if/while

# a program that tosses a coin until we get 10 heads and then prints how many flips it took to get those 10 heads

```python
def simulate_coin_toss():
    num_heads = 0
    num_tosses = 0
    while (num_heads < 10):
        toss = random.randint(0, 2)
        num_tosses+=1
        if toss == 0:
            num_heads +=1
    print("It took " + str(num_tosses) + " to get 10 heads")
```

a variable that stores the result of our coin flip (0 for heads, 1 for tails)
#a program that tosses a coin until we get 10 heads and then prints how many flips it took to get those 10 heads

def simulate_coin_toss():
    num_heads = 0
    num_tosses = 0
    while (num_heads < 10):
        toss = random.randint(0,2)
        num_tosses+=1
        if toss == 0:
            num_heads +=1
    print("It took " + str(num_tosses) + " to get 10 heads")

incrementing the num_toss variable to keep track of our coin toss
# A program that tosses a coin until we get 10 heads and then prints how many flips it took to get those 10 heads

def simulate_coin_toss():
    num_heads = 0
    num_tosses = 0
    while (num_heads < 10):
        toss = random.randint(0,2)
        num_tosses += 1
        if toss == 0:
            num_heads += 1
    print("It took " + str(num_tosses) + " to get 10 heads")
Using variables in `if/while`

# a program that tosses a coin until we get 10 heads and then prints how many flips it took to get those 10 heads

def simulate_coin_toss():
    num_heads = 0
    num_tosses = 0
    while (num_heads < 10):
        toss = random.randrange(0, 2)
        num_tosses += 1
        if toss == 0:
            num_heads += 1
    print("It took " + str(num_tosses) + " to get 10 heads")

if we landed on heads, increment the num_heads variable!
Using variables in if/while

A program that tosses a coin until we get 10 heads and then prints how many flips it took to get those 10 heads

```python
def simulate_coin_toss():
    num_heads = 0
    num_tosses = 0
    while (num_heads < 10):
        toss = random.randint(0,2)
        num_tosses+=1
        if toss == 0:
            num_heads +=1
    print("It took " + str(num_tosses) + " to get 10 heads")
```

Print out how many tosses it took
Diagnostic Problem 4
Write a program that asks the user to enter a sequence of "non-decreasing" numbers one at a time. Numbers are non-decreasing if each number is greater than or equal to the last.

When the user enters a number which is smaller than their previously entered value, the program is over. Tell the user how long their sequence was.
Enter a sequence of non-decreasing numbers.
Enter num:
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1

sequence_length: 0
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1

```python
print("Enter a sequence of non-decreasing numbers.")
sequence_length = 0
num1 = int(input('Enter a num:'))
```
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2

sequence_length: 0
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2

```python
print("Enter a sequence of non-decreasing numbers.")
sequence_length = 0
num1 = int(input('Enter a num:'))
num2 = int(input('Enter a num:'))
```
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2

STOP, before asking for another number: is num2 >= num1?
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2

STOP, before asking for another number: is num2 >= num1?

```python
print("Enter a sequence of non-decreasing numbers.")
sequence_length =0
num1 = int(input('Enter a num:'))
num2 = int(input('Enter a num:'))
if num2 >= num1:
```
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2

STOP, before asking for another number: is num2 >= num1?
Yes it is! So we ask for another number and increment our counter variable.
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2

STOP, before asking for another number: is num2 $\geq$ num1?
Yes it is! So we ask for another number and increment our counter variable

```python
print("Enter a sequence of non-decreasing numbers.") sequence_length = 0 num1 = int(input('Enter a num:')) num2 = int(input('Enter a num:')) if num2 $\geq$ num1:
    sequence_length +=1
```
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3

sequence_length: 1
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3

```
print("Enter a sequence of non-decreasing numbers.
sequence_length = 0
num1 = int(input('Enter a num:'))
num2 = int(input('Enter a num:'))
if num2 >= num1:
    sequence_length +=1
num3 = int(input('Enter a num:'))
```
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3

sequence_length: 1
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3

```python
print("Enter a sequence of non-decreasing numbers.")
sequence_length = 0
num1 = int(input('Enter a num: '))
num2 = int(input('Enter a num: '))
if num2 >= num1:
    sequence_length += 1
    num1 = num2
sequence_length: 1
```
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3

sequence_length: 1
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3

```
print("Enter a sequence of non-decreasing numbers.")
sequence_length = 0
num1 = int(input('Enter a num:'))
num2 = int(input('Enter a num:'))
if num2 >= num1:
    sequence_length +=1
    num1 = num2
num2 = int(input('Enter a num:'))
```

sequence_length: 1
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3

STOP, before asking for another number: is num2 >= num1?
Yes it is! So we ask for another number and increment our counter variable

sequence_length: 2
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3

STOP, before asking for another number: is num2 >= num1?
Yes it is! So we ask for another number and increment our counter variable

print("Enter a sequence of non-decreasing numbers.")
sequence_length = 0
num1 = int(input('Enter a num:'))
num2 = int(input('Enter a num:'))
while num2 >= num1:
    sequence_length += 1
    num1 = num2
    num2 = int(input('Enter a num:'))

sequence_length: 2
Asking for a nondecreasing sequence

Enter a sequence of non-decreasing numbers.
Enter num: 1
Enter num: 2
Enter num: 3
Enter num: 4
Enter num: 2

STOP, before asking for another number: is num2 >= num1?
No it isn't! So we print out the ending comments

sequence_length: 3
Enter a sequence of non-decreasing numbers.

Enter num: 1
Enter num: 2
Enter num: 3
Enter num: 4
Enter num: 2

STOP, before asking for another number: is num2 >= num1?
No it isn't! So we print out the ending comments

```python
print("Enter a sequence of non-decreasing numbers.")
sequence_length = 0
num1 = int(input('Enter a num: '))
num2 = int(input('Enter a num: '))
while num2 >= num1:
    sequence_length +=1
    num1 = num2
    num2 = int(input('Enter a num: '))
print("Thanks for playing!")
print("Sequence length: " + str(sequence_length))
```

sequence_length: 3
Solution

```python
print("Enter a sequence of non-decreasing numbers.")
sequence_length = 0
num1 = int(input('Enter a num:'))
num2 = int(input('Enter a num:'))
while num2 >= num1:
    sequence_length += 1
    num1 = num2
    num2 = int(input('Enter a num:'))
print("Thanks for playing!")
print("Sequence length: " + str(sequence_length))
```