Class 4 - Nested Loops and Lists Part I

MISE Summer Programming Camp 2023

Recap of Class 3

- While loops
  - Repeating operations multiple times
  - Using the **continue** and **break** keywords
- For loops
  - A different way to write code that repeats
  - Variations on the range function

What are loops for?

Loops allow us to run a chunk of code repeatedly until we are "done".

Problem: print every integer from 1 to 100...

Anatomy of a while loop

```
while condition:
    body
```

A boolean expression (evaluates to True or False). Determines whether we keep looping over the while loop body or not.

While loop body. Runs every time expression evaluates to True. Usually multi-line

indent (commonly 2 or 4 spaces. Standardized in each codebase)

Anatomy of a for loop

```
for i in range(n):
    body
```

Repeats an action a number of times given by the integer n

The variable i *will take the value between* 0 *and* n-1 *throughout the execution*

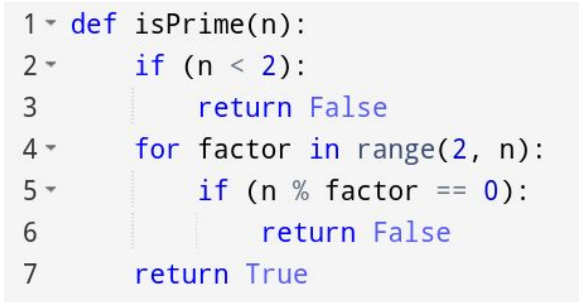
Runs n *times. Usually multi-line*

indent

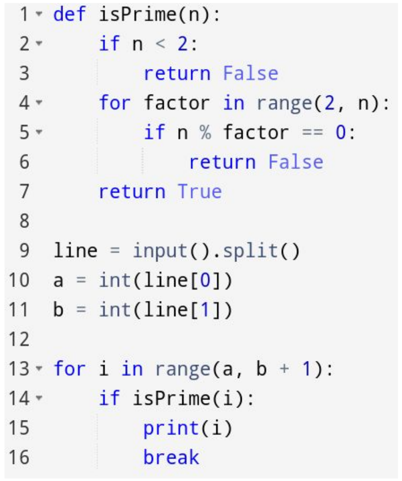
Review of homework 3

Let's go to: https://codeforces.com/

From last week



```python
def isPrime(n):
    if (n < 2):
        return False
    for factor in range(2, n):
        if (n % factor == 0):
            return False
    return True
```



```python
def isPrime(n):
    if n < 2:
        return False
    for factor in range(2, n):
        if n % factor == 0:
            return False
    return True

line = input().split()
a = int(line[0])
b = int(line[1])

for i in range(a, b + 1):
    if isPrime(i):
        print(i)
        break
```

Loops Part III: Nested Loops

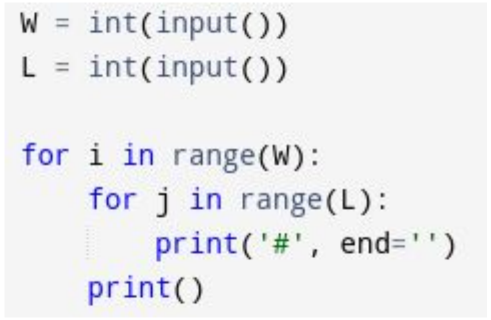
Motivation Problem

You are given two integers **W** and **L** and you have to print a **W** by **L** rectangle of the symbol **#**

For example if **W** = 3 and **L** = 4 we'd want to print:

```
####
####
####
```

Solution: nesting for loops!



```
W = int(input())
L = int(input())

for i in range(W):
    for j in range(L):
        print('#', end='')
    print()
```

Pay attention to the indentation!

- The second for loop is indented once
- The first print is indented twice
- The second print is indented once

Indentation shows to which for loop each line of code belongs to

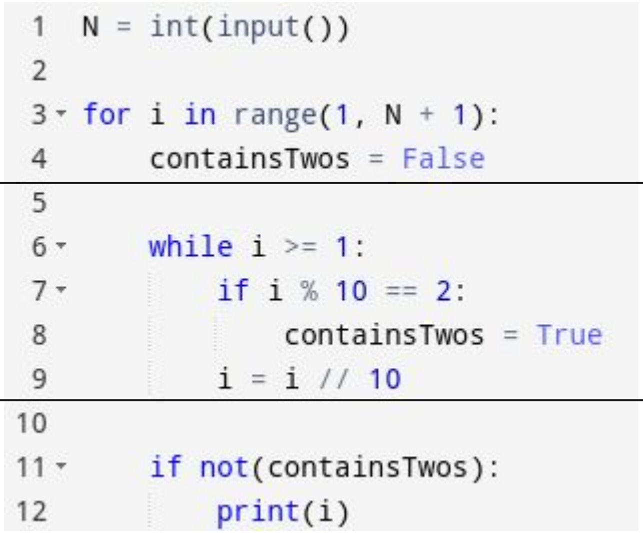
Python Tutor Link

A trickier example

Given an integer **N** print all integers between 1 and **N** that don’t have any digit 2 in its decimal representation

So if **N** = 30 the answer would be: 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 30

A trickier example



```python
N = int(input())

for i in range(1, N + 1):
    containsTwos = False
```

Part I

```python

    while i >= 1:
        if i % 10 == 2:
            containsTwos = True
        i = i // 10
```

Part II

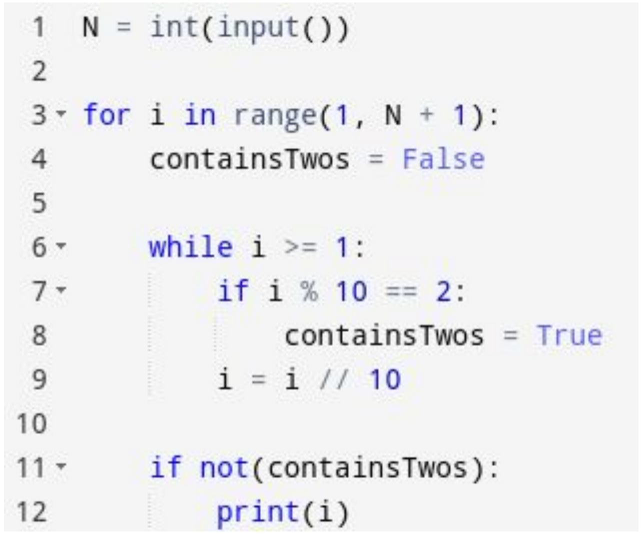
```python

    if not(containsTwos):
        print(i)
```

Part III

But wait!!!

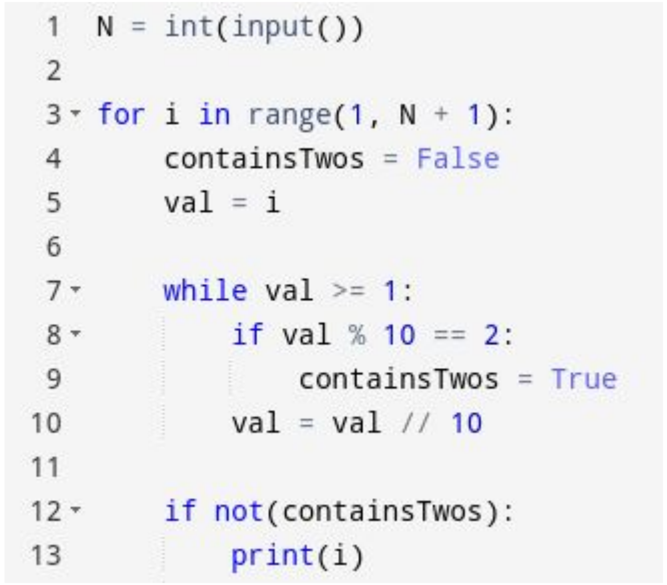
This code has a bug! It doesn't quite work!

Can you find the bug?



```python
N = int(input())

for i in range(1, N + 1):
    containsTwos = False

    while i >= 1:
        if i % 10 == 2:
            containsTwos = True
        i = i // 10

    if not(containsTwos):
        print(i)
```

Fixed Code

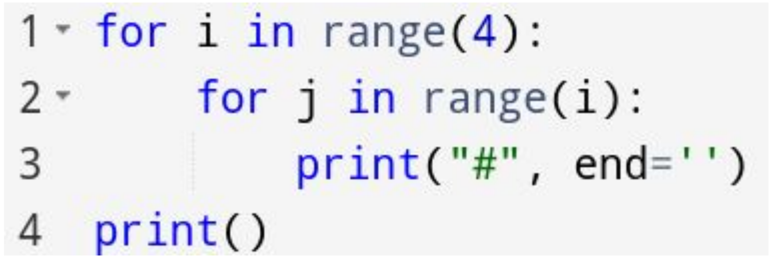

```python
N = int(input())

for i in range(1, N + 1):
    containsTwos = False
    val = i

    while val >= 1:
        if val % 10 == 2:
            containsTwos = True
        val = val // 10

    if not(containsTwos):
        print(i)
```

Note how we use an extra variable here

itempool.com/mise23/live

Pop Quiz 1:

What is the output of the following program:

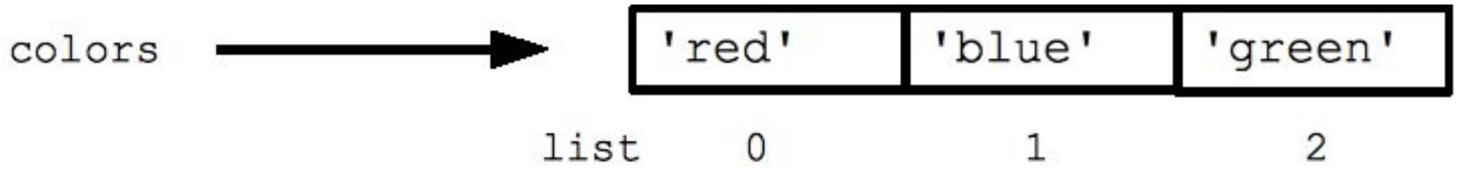


```python
for i in range(4):
    for j in range(i):
        print("#", end='')
print()
```

Lists and Tuples

Lists: how to store a collection of data

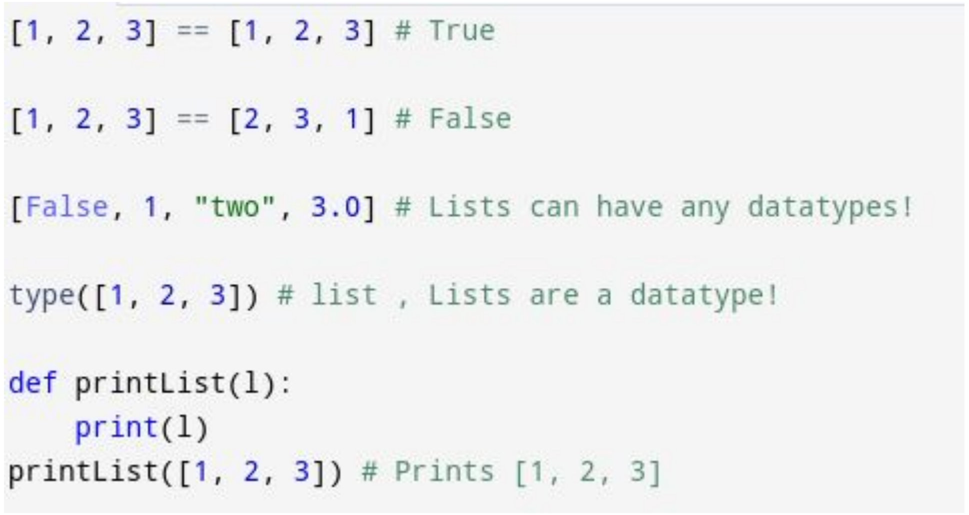
```
empty_list = []

colors = ["red", "blue", "green"]
numbers = [1, 2, 3, 4, 5]
```



```
colors[0] # red
colors[1] # blue
colors[2] # green
```

```
numbers[0] # 1
numbers[1] # 2
numbers[2] # 3
```

Demo: some list properties



```
[1, 2, 3] == [1, 2, 3] # True

[1, 2, 3] == [2, 3, 1] # False

[False, 1, "two", 3.0] # Lists can have any datatypes!

type([1, 2, 3]) # list , Lists are a datatype!

def printList(l):
    print(l)
printList([1, 2, 3]) # Prints [1, 2, 3]
```

Pop Quiz 2:

Which of the following would are the same as `[3, 1, 2]`in, when compared by ==):

```
[1, 2, 3]
```

```
["3", 1, 2]
```

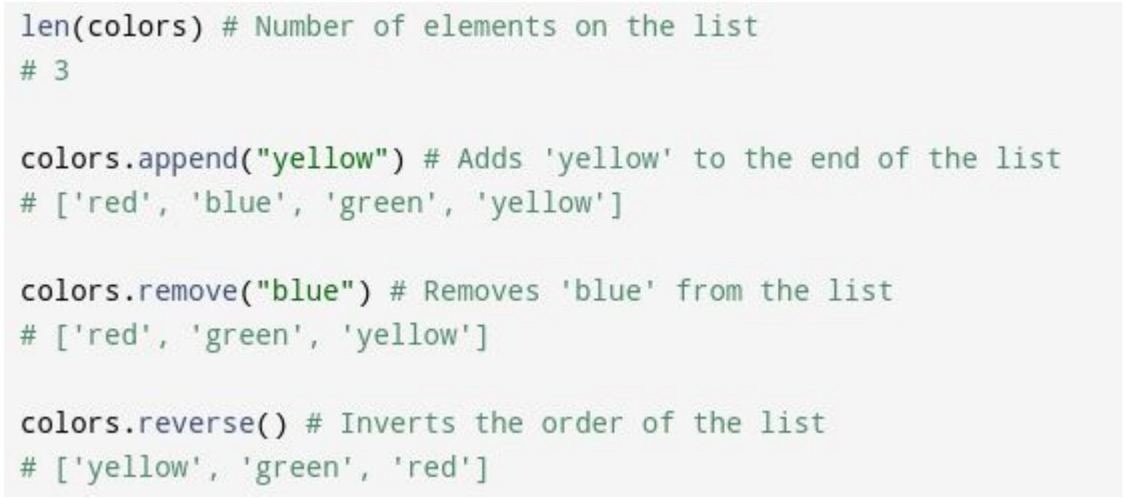
```
[int("3"), 1, 2]
```

```
[3, 1]
```

```
[3, 1] + [2]
```

How to use a list

Suppose we have some list **colors = [“red”, “blue”, “green”]**



```
len(colors) # Number of elements on the list
# 3

colors.append("yellow") # Adds 'yellow' to the end of the list
# ['red', 'blue', 'green', 'yellow']

colors.remove("blue") # Removes 'blue' from the list
# ['red', 'green', 'yellow']

colors.reverse() # Inverts the order of the list
# ['yellow', 'green', 'red']
```

Mutability and Tuples

Lists are *mutable*, meaning we can add and replace elements of a list:

```python
a = [1, 2, 3]
a[1] = 3
print(a) # [1, 3, 3]
a.append(4)
print(a) # [1, 3, 3, 4]
```

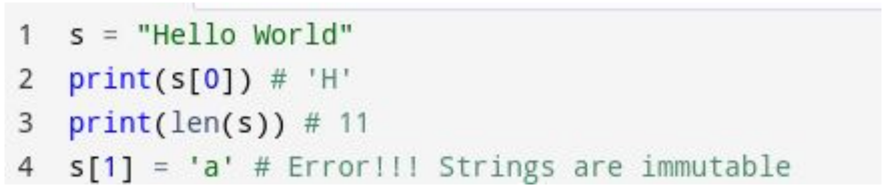
Tuples are like lists, but you can't modify them, they are *immutable*

```python
a = (1, 2, 3) # Tuple definition
a[1] = 3 # Error!!!
a.append(4) # Error!!!
```

There are a couple of reasons why tuples are interesting, here's one called *packing*

```python
t = (1, 2, 3) # Tuple definition
(a, b, c) = t
print(a) # 1
print(b) # 2
print(c) # 3

(a, b) = (b, a)
print(a) # 2
print(b) # 1
```

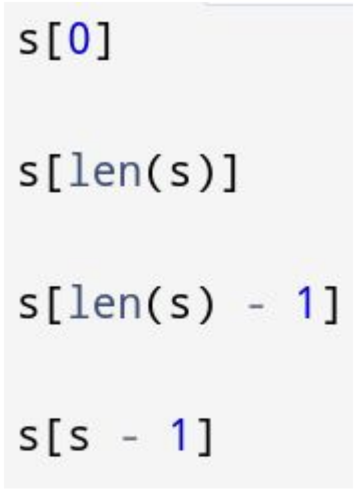
Recall strings? They are tuples of characters!



```python
s = "Hello World"
print(s[0]) # 'H'
print(len(s)) # 11
s[1] = 'a' # Error!!! Strings are immutable
```

Pop Quiz 3:

Which of the following prints the last character of a string variable 's':



```
s[0]

s[len(s)]

s[len(s) - 1]

s[s - 1]
```

List References

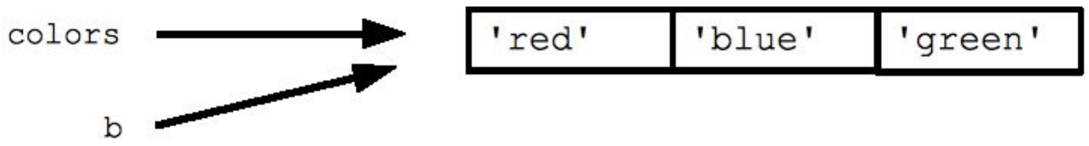
What's the output of the following code:

```python
a = 1
b = a
a = 2
print(b)
```

```python
a = [1]
b = a
a[0] = 2
print(b)
```

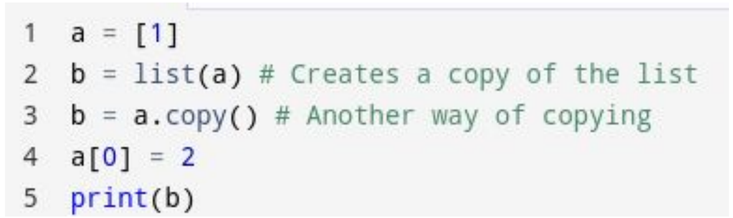
```python
a = [1]
b = a
a = [2]
print(b)
```

```python
colors = ["red", "blue", "green"]
b = colors
```



Copying a list

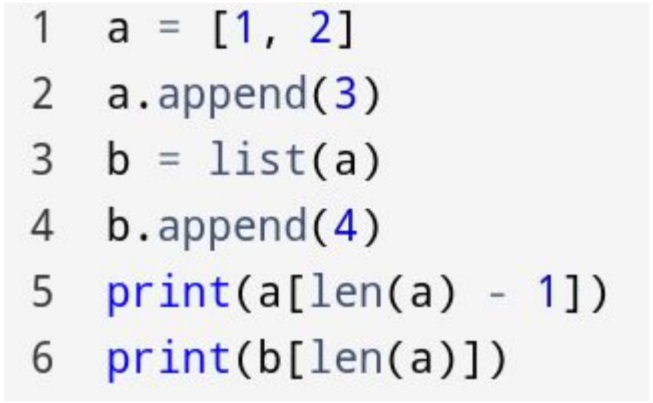
To fix the problem from the previous slide we can “copy” a list, which means creating a distinct clone of the original list.



```
a = [1]
b = list(a) # Creates a copy of the list
b = a.copy() # Another way of copying
a[0] = 2
print(b)
```

Pop Quiz 4:

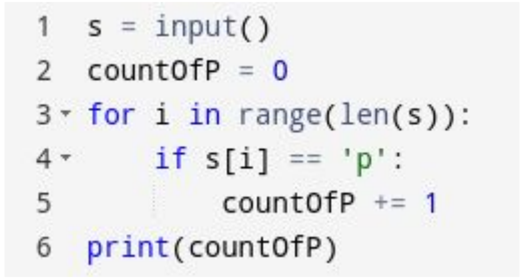
What is the output of the following code:



```python
a = [1, 2]
a.append(3)
b = list(a)
b.append(4)
print(a[len(a) - 1])
print(b[len(a)])
```

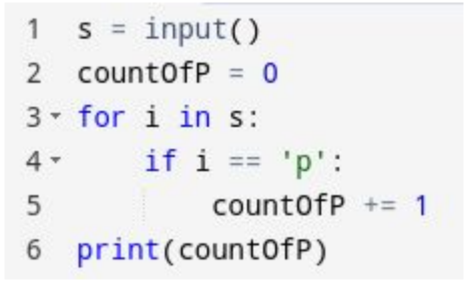
Example problem 1

You are given a string. Can you count how many times the letter ‘p’ shows up in the string?



```python
s = input()
countOfP = 0
for i in range(len(s)):
    if s[i] == 'p':
        countOfP += 1
print(countOfP)
```

Alternative solution:



```python
s = input()
countOfP = 0
for i in s:
    if i == 'p':
        countOfP += 1
print(countOfP)
```

Anatomy of a for loop revisited

for i in *list*:

The variable i will take each value in the list throughout the execution

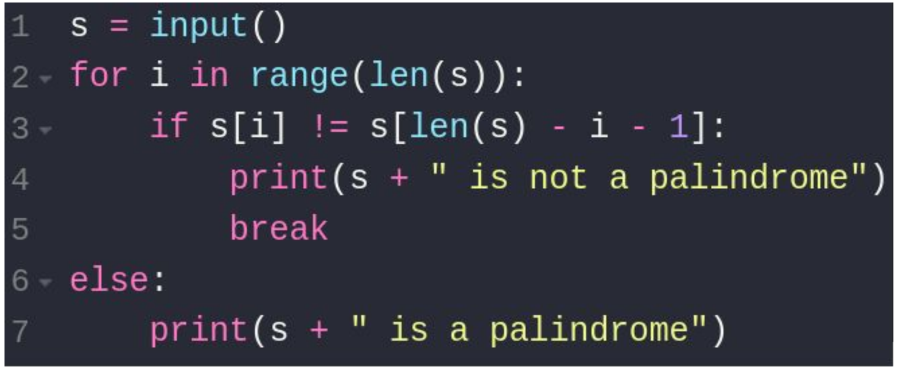
    body

Runs once per element

indent

Example problem 2

You are given a string. Determine if it is a palindrome (which means it is the same as its reverse).

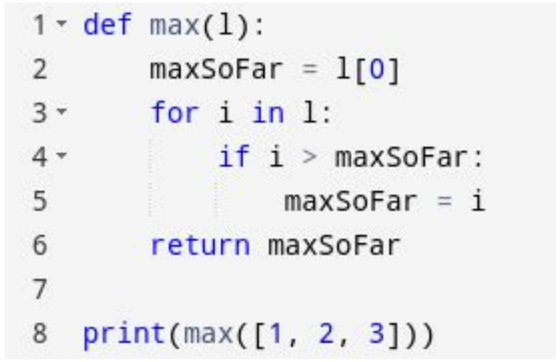


```python
s = input()
for i in range(len(s)):
    if s[i] != s[len(s) - i - 1]:
        print(s + " is not a palindrome")
        break
else:
    print(s + " is a palindrome")
```

Extra challenge: can you do the same in less iterations (less repetitions of the body of the for loop)?

Example problem 3

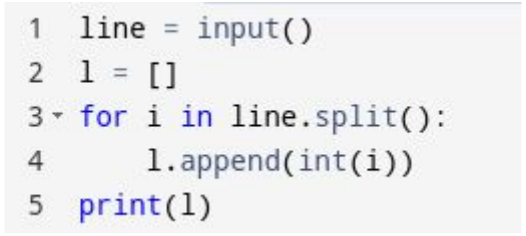
Implement a function that takes in a list of integers and returns the maximum of all of them.



```python
def max(l):
    maxSoFar = l[0]
    for i in l:
        if i > maxSoFar:
            maxSoFar = i
    return maxSoFar

print(max([1, 2, 3]))
```

Example problem 4

Write a program that reads a line of integers separated by spaces and turns that into a list with those values.



```
line = input()
l = []
for i in line.split():
    l.append(int(i))
print(l)
```

If you want to read more about Python lists, the following link has a comprehensive description of all properties of lists you need to know: https://realpython.com/python-lists-tuples/

What’s next?

Homework will be posted on Piazza by tomorrow! ← You won’t learn anything if you don’t try the homeworks

Class 5: Lists Part II

How to create lists of many dimensions

How to use advanced features of lists