Lab 6

- Review Lab 5
- Review relevant content
- Lab 6

Relational Operators

- Reminder: instead of, for example,

```
num < 0 or num > 0
```

can use

```
num != 0
```


Checking if a str contains a substring

Instead of using a method, could use `in` operator because didn't care where in the string it was:

```
if "r" in phrase:
```


Reversing a String

- Both are correct implementations
  - Preference depends on how you think

```
reverse = ""
for index in range(len(mystr)-1,-1,-1):
    reverse = reverse + mystr[index]
```

```
reverse = ""
for char in mystr:
    reverse = char + reverse
```

Over `str`ing

- Why do you ***not*** need to use `str` in the following code segments?

```
origString = str( input("What is your string? ") )
```

```
print(str(fahr), "degrees F is", str(cels), "degrees C")
```

Goal: Simplify/reduce code
→ Less code → easier to understand, less error-prone

Over `sys.exit()`

- You don't need to use sys.exit() every time you want to exit.
  - Typically use for an ***early*** exit
- You can let the program exit "naturally"

```
if "." not in mystr:
    # error message…
    sys.exit()
else:
    # code to handle file extension …
    sys.exit()
```


String Formatting Rules of Thumb

- Choose widths that are wider than your largest value
- Don't combine tabs and format specifiers
  - Can have unintended consequences
  - Plus, aren't necessary
- Use the appropriate code for the replacement value, e.g., f for a float
- Match the width of the column headings to the width of the data

Compare Palindrome Solutions (1)

```
possiblePal=input("What is your string?")

backwards=""
for x in range(0,len(possiblePal)):
    backwards=backwards + possiblePal[-x-1]

stringReplace=possiblePal.replace(" ","")
backwardsReplace=backwards.replace(" ", "")

stringLower=stringReplace.lower()
backwardsLower=backwardsReplace.lower()

if stringLower == backwardsLower:
    print(possiblePal, "is a palindrome")
else:
    print(possiblePal, "is not a palindrome")
```

Compare Palindrome Solutions (2)

```
possiblePal=input("What is your string?")

pPalReplace=possiblePal.replace(" ","")
pPalLower=pPalReplace.lower()

backwards=""
for x in range(0,len(pPalLower)):
    backwards=backwards + pPalLower[-x-1]

if pPalLower== backwardsLower:
    print(possiblePal, "is a palindrome")
else:
    print(possiblePal, "is not a palindrome")
```


Compare Palindrome Solutions (2)

```
possiblePal=input("What is your string?")

backwards=""
for x in range(0,len(possiblePal)):
    backwards=backwards + possiblePal[-x-1]

stringReplace=possiblePal.replace(" ","")
backwardsReplace=backwards.replace(" ", "")

stringLower=stringReplace.lower()
backwardsLower=backwardsReplace.lower()

if stringLower == backward
    print(possiblePal, "is
else:
    print(possiblePal, "is
```

```
possiblePal=input("What is your string?")

pPalReplace=possiblePal.replace(" ","")
pPalLower=pPalReplace.lower()

backwards=""
for x in range(0,len(pPalLower)):
    backwards=backwards + pPalLower[-x-1]

if pPalLower== backwardsLower:
    print(possiblePal, "is a palindrome")
else:
    print(possiblePal, "is not a palindrome")
```


Compare Palindrome Solutions (2)

```
possiblePal=input("What is your string?")

backwards=""
for x in range(0,len(possiblePal)):
    backwards=backwards + possiblePal[-x-1]

stringReplace=possiblePal.replace(" ","")
backwardsReplace=backwards.replace(" ", "")

stringLower=stringReplace.lower()
backwardsLower=backwardsReplace.lower()

if stringLower == backwar
    print(possiblePal, "i
else:
    print(possiblePal, "i
```

```
possiblePal=input("What is your string?")

pPalReplace=possiblePal.replace(" ","")
pPalLower=pPalReplace.lower()

backwards=""
for x in range(0,len(pPalLower)):
    backwards=backwards + pPalLower[-x-1]

if pPalLower== backwardsLower:
    print(possiblePal, "is a palindrome")
else:
    print(possiblePal, "is not a palindrome")
```

This version is the preferred version. Why?

Review

- How can we find the ASCII value for a character?
- How can we find the character associated with an ASCII value?

Review

- What is the syntax for representing a list?
- What are some common list methods and operations?
- How do we open a file?
- How do we read from a file?

Fibonacci Sequence

- Goal: Solve using *list*
- $F_0=F_1=1$
- $F_n=F_{n-1}+F_{n-2}$
- Example sequence: 1, 1, 2, 3, 5, 8, 13, 21, …

Fibonacci Sequence

- Create a list of the 1st 15 Fibonacci numbers
  - $F_0=F_1=1$; $F_n=F_{n-1}+F_{n-2}$

```
fibs = []
fibs.append(1)
fibs.append(1)
for x in range(13):
   newfib = fibs[-1]+fibs[-2]
   fibs.append(newfib)

for num in fibs:
   print(num)
```

- Grow list as you go

Fibonacci Sequence

- Create a list of the 1st 15 Fibonacci numbers
  - $F_0=F_1=1$; $F_n=F_{n-1}+F_{n-2}$

```
fibs = [0]*15
fibs[0] = 1
fibs[1] = 1
for x in range(2, len(fibs)):
   newfib = fibs[x-1]+fibs[x-2]
   fibs[x] = newfib

for num in fibs:
   print(num)
```

- Create whole list
- Update values

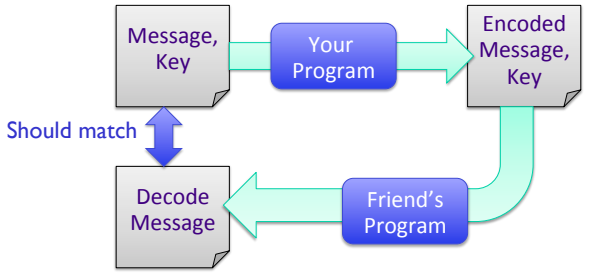
Data Types of Loop Variables

What are the data types of the loop variable **x**?

```
string = "some string"
dataFile = open("datafile.dat", "r")

for x in range(len(string)):
   # loop body …

for x in string:
   # loop body …

for x in dataFile:
   # loop body …
```


Caesar Cipher

- Write an encoding/decoding program
  - Encode a message
  - Give to a friend to decode



Caesar Cipher (Partial) Algorithm

- For each character in the message
  - Check if the character is a space; if it is, it stays a space
  - Otherwise
    - Convert the character to its ASCII value
    - Add the key to that value
    - Make sure that the new value is a "valid" ASCII value, i.e., that that new value is in the range of lowercase letter ASCII values
      - If not, "wrap around" to adjust that value so that it's in the valid range
    - Convert the ASCII value into a character

Lab 6

- List practice
- ASCII practice
- File practice
  - Caesar Cipher
- Larger programs, practice problem solving