Data Analysis

Lists & A Few Other Things
exam1 this week (lec00–lec04)
Expected homework due dates posted.
Office hours slightly altered (check website).
Recap
def hand(bird):
    while bird > 2:
        bird = bird - 1
    return bird

t = hand(4)

What is the final value of t?
A 4
B 3
C 2
D 1
def hand( bird ):
    while bird > 2:
        bird = bird - 1
    return bird

t = hand( 4 )

What is the final value of t?
A  4
B  3  ⋆return is inside loop!
C  2
D  1
s = 'ABCDEFGH'
t = 

i = 0
while i < 8:
    t = t + s[ i+1 ]
    i += 2

What is the final value of t?

A  ”ACEG”
B  ”BDFH”
C  ”ABCDEF”
D  ”ABEF”
s = 'ABCDEFGH'
t = 

i = 0
while i < 8:
    t = t + s[i+1]
    i += 2

What is the final value of t?
A ”ACEG”
B ”BDFH” *
C ”ABCDEF”
D ”ABEF”
s = '0123456789'
t = 
i = 0
while i < 5:
    if (i%2) == 1:
        t = t + s[i-1]
    if (i%2) == 0:
        t = t + s[i+1]
i = i + 1

What is the final value of t?
A "92143"
B "103254"
C "10325"
D "921436"
E None (loop doesn’t terminate)
s = '0123456789'
t = ''
i = 0
while i < 5:
    if (i%2) == 1:
        t = t + s[i-1]
    if (i%2) == 0:
        t = t + s[i+1]
i = i + 1

What is the final value of t?
A "92143"
B "103254"
C "10325"
D "921436"
E None (loop doesn’t terminate)

Draw a flowchart!
s = '0123456789'
t = 
 i = 0
while i < 5:
    if (i%2) == 1:
        t = t + s[ i-1 ]
    if (i%2) == 0:
        t = t + s[ i+1 ]
    i = i + 1

What is the final value of t?
A   ”92143”
B   ”103254”
C   ”10325” ⋄
D   ”921436”
E  None (loop doesn’t terminate)
What changes does this code need to be properly executable?

```python
if x < 1.5:
    x = x + 1
if x == (1.5 or 2.0):
    x = x - 1
```
What changes does this code need to be properly executable?

```python
if x < 1.5:
    x = x + 1
if x == 1.5 or x == 2.0:
    x = x - 1
```
for i in range(10):
    print(i ** 2)
The `range` function returns a set of integers.

Two arguments:
- (optional) the starting value of the range (inclusive)
- the ending value of the range (exclusive)
- (optional) the step size
for i in range( 10,0,-1 ):
    print(i ** 2)
Container Data Types
colors = ['red', 'yellow', 'blue', 'jale', 'ulfire']
for color in colors:
    print( color.title() )
The list type represents an ordered collection of items.

- list is an iterable and a container.
- Containers hold values of any type (doesn’t have to be the same).
We create a list as follows:
- opening bracket [
- one or more comma-separated data values
- closing bracket ]
lists work a bit like strings:

```
x = [ 10, 3.14, "Ride" ]
print( x[1] )
print( x[1:3] )
print( len(x) )
```
But strings are **immutable** (we can’t change contents without creating a new string):

```python
s = "good advise"
s[9] = 'c'  # nope
s = s[:9] + 'c' + s[9:]  # this way
```
But strings are immutable (we can’t change contents without creating a new string):

```python
s = "good advise"
s[9] = 'c'  # nope
s = s[:9] + 'c' + s[9:]  # this way
```
We can change list content—they are mutable.

```python
x = [ 4, 1, 2, 3 ]
x[3] = -2  # sets an element
x.append(5)  # adds an element
del x[1]  # removes an element
x.sort()  # sorts in place
x.extend([5])  # adds an iterable
```
String Methods
These produce Boolean output.

- `isdigit()` Does a string contain only numbers?
- `isalpha()` Does a string contain only text?
- `islower()` Does a string contain only lower-case letters?
- `isupper()` Does a string contain only upper-case letters?
answer = input( 'How do you feel? ' )
if not answer.isalpha():
    print( "I don’t understand." )
else:
    print( "Ah, you feel %s." % answer )
Exercise

Write a program for a user to create a new password. The program should accept a password attempt from the user and check it with the function `validate_password`. If the password is valid, the program ends. If the password is invalid, the program asks for a new attempt, repeating until the user enters a valid password.
pwd = input("Enter a password: ")
while not validate_password(pwd):
    pwd = input("INVALID! Try again: ")
print("Your password is valid.")
Example: if statement

```python
ans = input( "Enter a number:" )
if float(ans) < 0:
    print( "The number was negative." )
```
Control flow represents actual sequence of lines executed by processor.

Conditional execution lets you execute (or not) a block of code based on logical comparison.
We often need to make decisions with several options.

Branched conditional execution lets you execute one of several blocks of code.
def absolute(x):
    if x >= 0:
        return x
    else:
        return -x
We create an if/else statement as follows:
- the keyword if
- a logical comparison (results in bool)
- a block of code
- the keyword else
- a different block of code
These produce Boolean output.

- `in`  Is one string inside of the other?
- `not in`  Is one string not inside of the other?

These also work with lists.
def fun(s):
    return s.isalpha() and 'a' in s

x = fun( "sam" ) and fun( "AS" )

What is the value of x?
A True  
B False
def fun(s):
    return s.isalpha() and 'a' in s

x = fun( "sam" ) and fun( "AS" )

What is the value of x?
A True
B False ⋆
Sometimes we need to make more than one decision.

We can **nest** blocks.

```python
word = input( 'Enter a Scrabble word: ' )
if not word.isalpha():
    print( 'There are only letters in Scrabble!' )
else:
    if not word.isupper():  # why not 'word.islower()
        word = word.upper()
    print( 'You entered %s.' % word )
```
Nesting

Conditional Execution

```
not alphabetical?

True
"There are only letters in Scrabble."

False

not upper-case?

True
.upper()

False
"You entered %s."
```
def evenpos(x):
    if x >= 0:
        if (x%2) == 0:
            return x
        else:
            return x + 1
    else:
        if (x%2) == 0:
            return -x
        else:
            return (-x) + 1
Sometimes we need to select among many choices.
if day == 1:
    print("Sunday")
else:
    if day == 2:
        print("Monday")
    else:
        if day == 3:
            print("Tuesday")
        else:
            if day == 4:
                print("Wednesday")
            else:
                if day == 5:
                    print("Thursday")
                else:
                    if day == 6:
                        print("Friday")
                    else:
                        if day == 7:
                            print("Saturday")
if day == 1:
    print("Sunday")
elif day == 2:
    print("Monday")
elif day == 3:
    print("Tuesday")
elif day == 4:
    print("Wednesday")
elif day == 5:
    print("Thursday")
elif day == 6:
    print("Friday")
elif day == 7:
    print("Saturday")
else:
    print("That is not a valid day.")
We create an `if/elif/else` statement as follows:

- the keyword `if`
- a logical comparison (results in `bool`)
- a `block` of code
- the keyword `elif`
- a logical comparison (results in `bool`)
- a `block` of code
- the keyword `else`
- a different `block` of code
Next steps
Next steps

- Complete quiz06 (due 2/9)
- Complete exam1 (2/7–2/9)
- Wait for hw03 (due 2/17)
- Read for the next class