CS116 resources

Schema:

Process all items in a list.

```
for item in list:
      process item
```

Process items in a list that have a property.

```
for item in list:
      if item has property:
           process item
```

Process items in a list with a property one way and all the other items another way.

```
   for item in list:
      if item has property:
           process item
      else:
           process item another way
```

Process all items in a list given many different properties for choosing how they will be processed.

```
   for item in list:
      if item has property:
           process item one way
      elif item has a second property:
           process item a second way
      ...
      else:
          process item a last way
```

Accumulate a value from a list.

```
resultSoFar = an initial value
for item in list:
      resultSoFar = resultSoFar combined with processed item
return resultSoFar
```

Accumulate values in a list that have a property.

```
resultSoFar = an initial value
for item in list:
     if item has a property:
         resultSoFar = resultoFar combined with processed item
return resultSoFar
```

Process multiple lists at corresponding locations.

```
for index in range(0,length of lists):
      process list1[index]
      ...
      process listn[index]
```

Process a portion of a list starting at an *offset* location.

```
for index in range(0,length of chunk):
      process list[offset+index]
```

Process a list or lists where you need to compute the locations being used.

```
for index in range(0,length to be processed):
      process list[computation of index]
```

Process a region within a matrix.

```
for xindex in range(0,width to be processed):
      for yindex in range(0,height to be processed):
           process item at location (xoffset+xindex, yoffset+yindex)
```

Process an indefinite number of times.

```
while processing is not done:
      process stuff
```

Process recursively when iteration is impractical.

```
def recursiveFunction(parameters):
      if base case condition:
           return a value or do an operation
      else:
          use recursiveFunction(smaller parameter values)
```

Readings and Videos:

Variables, Assignment, and Expressions:
http://www.openbookproject.net/thinkcs/python/english2e/ch02.html
https://realpython.com/python-operators-expressions/
http://interactivepython.org/runestone/static/CS152f17/SimplePythonData/intro-VariablesExpressionsandState
http://interactivepython.org/runestone/static/CS152f17/SimplePythonData/Variables.html
http://interactivepython.org/runestone/static/CS152f17/Selection/BooleanValuesandBooleanExpressions.html

Objects, Methods, and Parameters:
http://interactivepython.org/runestone/static/CS152f17/PythonTurtle/OurFirstTurtleProgram.html
http://introtopython.org/classes.html

Definite loops and lists:
http://interactivepython.org/runestone/static/CS152f17/PythonTurtle/TheforLoop.html

If-else and if-elif-else:
http://interactivepython.org/runestone/static/CS152f17/Selection/ConditionalExecutionBinarySelection.html
http://interactivepython.org/runestone/static/CS152f17/Selection/Chainedconditionals.html

Accumulating a value and returning a value:
http://interactivepython.org/runestone/static/CS152f17/Functions/Functionsthatreturnvalues.html
http://interactivepython.org/runestone/static/CS152f17/Functions/TheAccumulatorPattern.html

Looping through a list with an index:
http://interactivepython.org/runestone/static/CS152f17/Lists/Listsandforloops.html

Nested loops in a two dimensional list (matrix):
https://www.youtube.com/watch?v=rUDdguWk-QQ
https://snakify.org/en/lessons/two_dimensional_lists_arrays/

Indefinite loops:
http://interactivepython.org/runestone/static/CS152f17/MoreAboutIteration/ThewhileStatement.html

Recursion:
http://interactivepython.org/runestone/static/CS152f17/IntroRecursion/toctree.html

Turtle API

- Constructors: to create a new turtle you can use
  `Turtle()`: creates a turtle in the center of the world
  `Turtle(x,y)`: creates a turtle at position (x,y) in the world
- `forward(distance)`: moves the turtle forward by the distance.
- `right(angle)`: turns right
- `left(angle)`: turns left
- `penup()`: lifts the pen
- `pendown()`: lowers the pen
- `setheading()`: sets the orientation of the turtle (0-east, 90-north, etc)
- `setpos(x,y)`: sets the turtle position to location (x,y)
- `pencolor(colorString)`: sets the pen color to string like "red", "blue", etc
- `fillcolor(colorString)`:'sets the fill color for a filled shape
- `begin_fill()`: starts fill so that it draws a filled shape
- `end_fill()`: ends drawing a filled shape

Picture API

- Constructors: to create a new picture you can use
  `Picture(width,height)`: creates a picture of the given size
  `Picture(filename)`: create a picture from a file
- `show()`: displays the picture
- `getPixels()`: returns the list of pixels
- `write(filename)`: writes the picture to a file
- `getPixel(x,y)`: returns the pixel at location (x,y)
- `getWidth()`: returns the width of the picture
- `getHeight()`: returns the height of the picture

Pixel API

- `getRed()`: returns the red value
- `getGreen()`: returns the green value
- `getBlue()`: returns the blue value
- `setRed(redValue)`: changes the red value
- `setGreen(greenValue)`: changes the green value
- `setBlue(blueValue)`: changes the blue value
- `getColor()`: returns the color
- `setColor(color)`: changes the color

Color API

- `colorDistance(anotherColor)`: returns a number indicating how distant the two colors are from one another
- `white`
- `black`
- `red`
- `green`
- `blue`
- `yellow`
- `pink`
- `magenta`
- `orange`

Sound API

- Constructors: to create a new sound you can use
  `Sound(size)`: creates and empty sound of the given size
  `Sound(filename)`: creates a sound from a file
- `getLength()`: returns the number of samples
- `getSamples()`: returns the list of samples
- `play()`: plays the sound
- `write(filename)`: writes the sound to a file
- `explore()`:explores the sound wave

SoundSample API

- `getValue()`: returns the sample value
- `setValue(newValue)`: changes the value of the sample

Text API

- Constructor: to create a new text you use
  `Text(filename)`: creates a text from a file
- `getWords()`: returns a list of words
- `getLines()`: returns a list of lines of text

Word API

- Constructor: to create a new word you use
  `Word(string)`: creates a word from a string
- `getLetters()`: returns list of letters