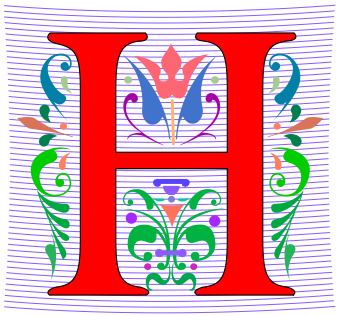
Compsci 101
turtles, for loop, accumulation, range

Susan Rodger

February 3, 2022

create a new empty string **answer**
for each character **ch** in the **phrase**
    if **ch** is not a vowel
        add **ch** to the end of **answer**
answer is the result

`H` is for …



- **HTTP**
  - A Protocol we use every day, and HTTPS
- **Hello World**
  - The quintessential first program: 40 years ago!
    http://helloworldcollection.de/
- **Hack**
  - Hacker, Hacktivism, Hack Duke
- **Hashing**
  - How Dictionaries work

Brian Fox

- See Wikipedia: http://bit.ly/brianfox2018
  - Bash Shell, Stallman, Wells Fargo, more
- See LifeHacker: http://bit.ly/brianfox-hack
  - Learned Logo at 8, wrote it at 21 for Apple!
  - Open Voting

There's nothing that I am better at than everyone else, except being me. There's no secret to being me. Follow your interests and work hard at them. Then you will play bass better, program better, cook better, ride motorcycles better, or anything else that you really want to do.

Announcements

- Assignment 2 out, due Feb 15
  - Assignment 2 Quiz due Feb 14!
- APT-2 out, due Tuesday, February 10
- Lab 4 Friday
  - Complete prelab 4 before going to lab
- Don’t discuss Exam 1 until it is handed back!

PFTD

- Import a file
- String and List Functions
- Turtles
- For loop/Accumulation

Main: if __name__ == '__main__':

- Main – where Python starts and ends in some file
- Consider file Food.py

```
def makeSandwich(food):
    print("Making the sandwich", food)

if __name__ == '__main__':
    makeSandwich('peanut butter and jelly')
```

Main vs. Import

- Import – another file with useful code (functions)
  - Ignores if __name__ == '__main__' in the other file
- Food.py

```
def makeSandwich(food):
    print("Making the sandwich", food)

if __name__ == '__main__':
    makeSandwich('peanut butter and jelly')
```

- FoodFriend.py

```
import Food

if __name__ == '__main__':
    Food.makeSandwich("bacon, lettuce, and tomato")
```

Run FoodFriend.py

- FoodFriend.py

```
import Food

if __name__ == '__main__':
    Food.makeSandwich("bacon, lettuce, and tomato")
```

Main vs. Import

- Run main in Food.py

- Run main in FoodFriend.py

Modify Main – Bad Code!

- Modify Food.py

```
def makeSandwich(food):
    print("Making the sandwich", food)

makeSandwich('hummus and sprouts')

if __name__ == '__main__':
    makeSandwich('peanut butter and jelly')
```

Code not indented, this is bad code!!!!

More String and List Methods

String

`.find(s)`	index of first occurrence of s
`.rfind(s)`	index of last occurrence of s (from Right)
`.upper()/ .lower()`	uppercase/lowercase version of string
`.strip()`	remove leading/trailing whitespace
`.count(s)`	number of times see s in string
`.startswith(s)`	bool of whether the string begins with s
`.endswith(s)`	bool of whether the string ends with s

List

`sum(lst)`	sum of the elements in lst
`max(lst)`	maximum value of lst
`min(lst)`	minimum value of lst
`.append(elm)`	Mutates the list by adding elm to the end of the list
`.count(elm)`	Number of times see elm in the list

Some String Methods

str = ‘ghosts’

x = str.find(‘s’)

x = str.rfind(‘s’)

x = str.upper()

x = str.count(‘s’)

x = str.startswith(‘gh’)

x = str.endswith(‘s’)

x = str.endswith(‘t’)

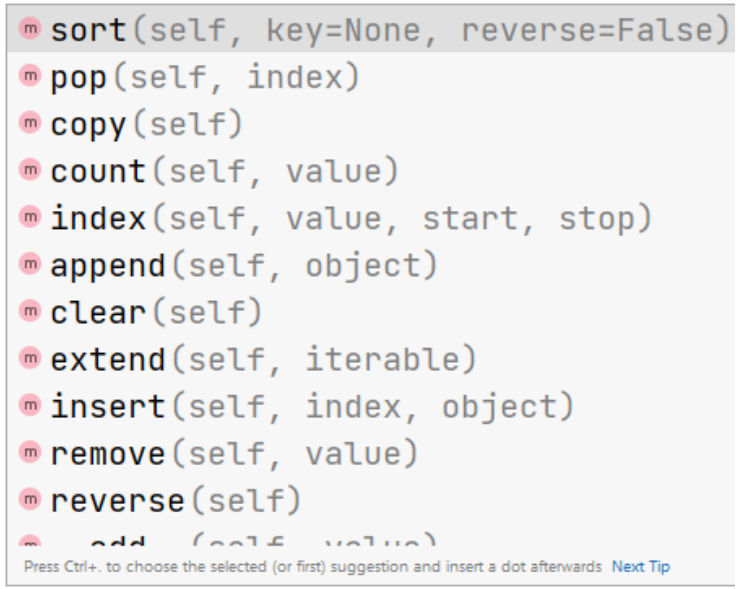
Some List Methods

```
lst = [5, 2, 4, 5, 5]
x = max(lst)
x = min(lst)
x = sum(lst)
x = lst.count(5)
lst.append(7)
x = lst.append(3)
```

lst is [5, 2, 4, 5, 5]

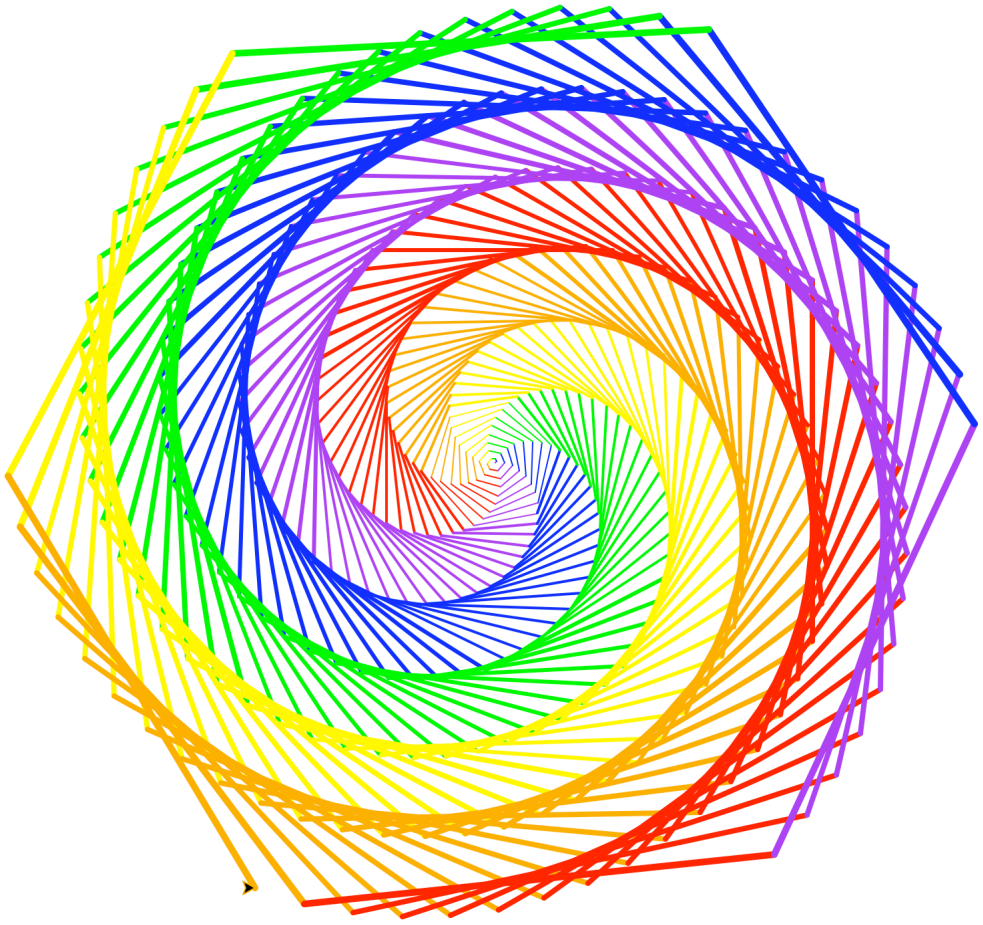
Three ways to use functions with List

- x = max(lst)
  - lst is parameter to the max function that has a return value
- x = lst.count(5)
  - lst has its own functions that can be applied to a list
- lst.append(7)
  - mutates/changes lst, no return value



WOTO-1 – Import, Strings and Lists
http://bit.ly/101s22-0203-1

Run Turtle, Run



Turtle Programming

- **Must:**
  - Import turtle module
  - Create window/Screen
  - Last thing - exit on click
  - Create turtles to use, name/type/value
- **Review Turtle commands and concepts**
  - http://bit.ly/turtle_tutorial for more, and book

- **See Snowpeople.py, ColorMyWorld.py, and Spiro.py for some ideas**
  - Color, Position, Leaving Turtle where started
  - Many more commands than this

Put yourself in the turtle t…

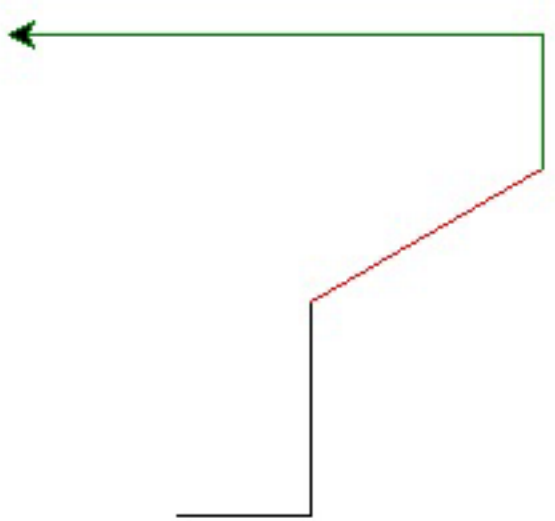
```
t.forward(50)          # turtle moves forward
                           #  drawing  a line
t.left(90)             # turtle turns to its left
t.pencolor("blue")     # change pen color
t.forward(100)         # turtle moves forward
                           # drawing  line, new color
```

Example: simple.py

```
import turtle

def drawPicture(turt):
    t.forward(50)
    t.left(90)
    t.forward(80)
    t.pencolor('red')
    t.right(60)
    t.forward(100)
    t.pencolor('green')
    t.left(60)
    t.forward(50)
    t.left(90)
    t.forward(200)


if __name__ == '__main__':
    win = turtle.Screen()
    t = turtle.Turtle()
    drawPicture(t)
    win.exitonclick()
```



Example: Simple.py parts

```
import turtle
```

- Import at the top

```
if __name__ == '__main__':
    win = turtle.Screen()
    t = turtle.Turtle()
    drawPicture(t)
    win.exitonclick()
```

Example: Simple.py DrawPicture

```
def drawPicture(turt):
    turt.forward(50)
    turt.left(90)
    turt.forward(80)
    turt.pencolor('red')
    turt.right(60)
    turt.forward(100)
    turt.pencolor('green')
    turt.left(60)
    turt.forward(50)
    turt.left(90)
    turt.forward(200)
```

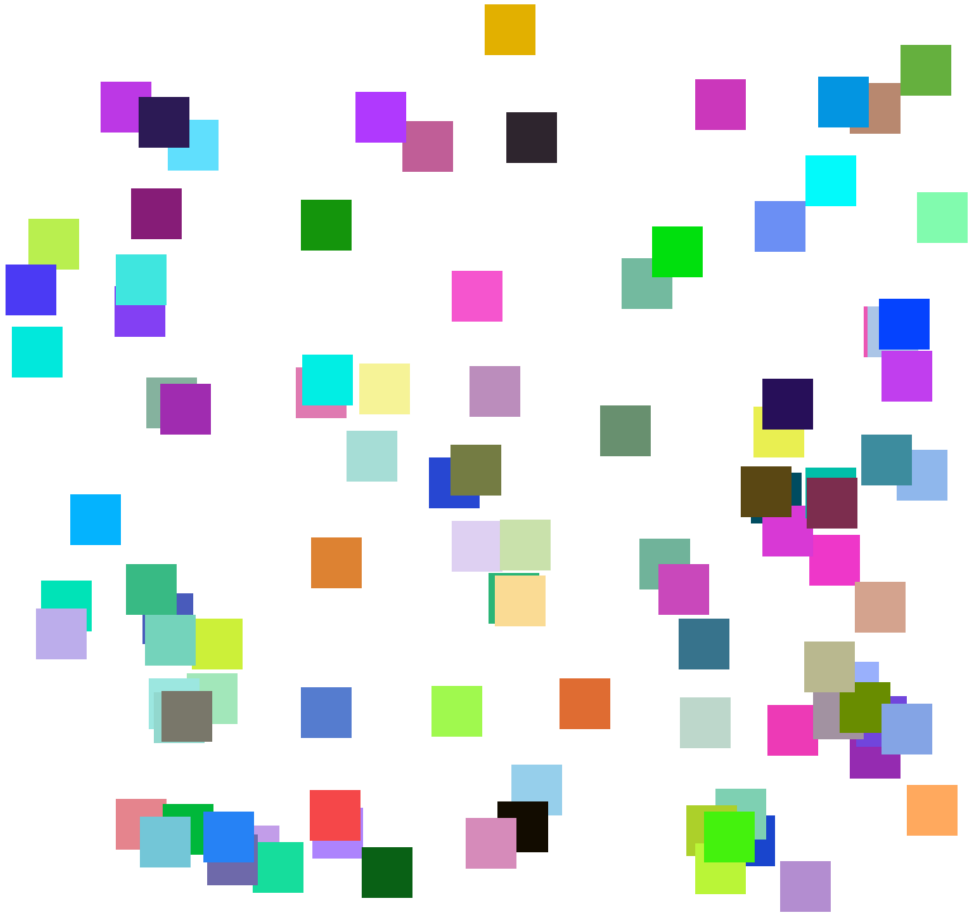
What are key concepts in Spiro.py?

```
8   import turtle
9
10  def draw(turt):
11      colors = ['red', 'purple', 'blue', 'green', 'yellow', 'orange']
12      turt.speed(0)
13      for x in range(360):
14          turt.pencolor(colors[x % 6])
15          turt.width(x/100 + 1)
16          turt.forward(x)
17          turt.left(59)
18
19  if __name__ == '__main__':
20      win = turtle.Screen()
21      t = turtle.Turtle()
22      draw(t)
23      win.exitonclick()
```

Useful turtle functions

- **`forward(n)/backward(n)`** – move turtle n pixels
- **`left(n)/right(n)`** – turn turtle n degrees
- **`pendown()/pendup()`** – whether actually drawing
- **`setposition(x, y)`** – puts turtle in this (x,y) coordinate (a.k.a. **`goto`**, **`setpos`**)
- **`sethead(n)`** – points turtle in this direction (n=0 is east)
- Many more in documentation!
  - https://docs.python.org/3/library/turtle.html

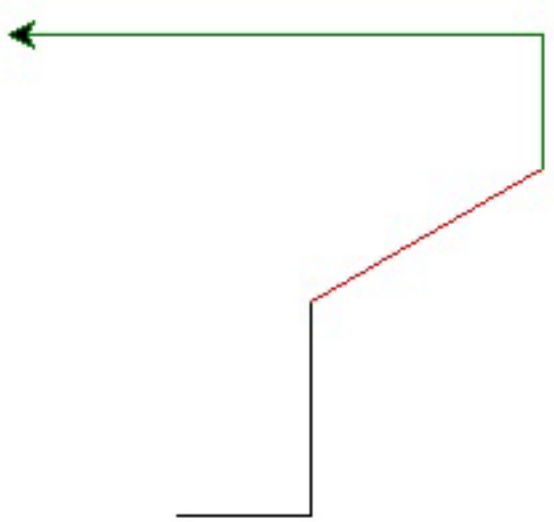
ColorMyWorld.py



Turtle Concepts

- Create a screen so you can ..
  - Exit On Click
  - Some other Screen Functions
- Create a turtle so you can …
  - Move and draw using the turtle
- Drawing Concepts
  - Pen [up and down]
  - Fill
  - Color
  - Position

WOTO-2 - Turtles
http://bit.ly/101s22-0203-2



Looping over Sequences

- **Let's explore this:**
  - Given a sentence:
    - "Duke Computer Science is so much fun!"
  - How do we create this sentence?
    - "Dk Cmptr Scnc s s mch fn!"
  - Input is sentence. Output has vowels removed

Step 3 and 4: Algorithm: input is string **phrase**

create a new empty string **answer**

for each character **ch** in the **phrase**

    if **ch** is not a vowel

        add **ch** to the end of **answer**

answer is the result

Step 4: Test this algorithm on “Computer”

Step 5: Now ready to translate this to code!

Step 5: Translate to Code

- **Which function do we implement first?**
  - Translate sentence?
  - Is a vowel?
  - Reasons to prefer one to the other?

- **How do we verify that our function is correct?**
  - Reproducible testing not detailed here
  - Testing is really, really important

WOTO-3
http://bit.ly/101s22-0203-3

- Is it a vowel?

Testing IsVowelFunctions

```
if __name__ == '__main__':
    print("Testing isVowel functions")
    lstIsVowel = [isVowel1, isVowel2, isVowel3]

    for v in lstIsVowel:
        print("\nTesting: ", v)
        print(v('a') == True)
        print(v('E') == True)
        print(v('b') == False)
        print(v('Z') == False)
```

Accumulator Pattern: NoVowels

- "For each character, if it's not a vowel add it to the output string"
- Accumulator pattern: change a variable in a loop
  - Accumulate a value while iterating through loop

```
20  def noVowels(phrase):
21      ret = ""
22      for ch in phrase:
23          if not isVowel1(ch):
24              ret = ret + ch
25      return ret
```