What do we know so far?

- Class – lecture, lab, Refactor, Quick Checks, R&R
- Solve problems, computers useful, user vs. programmer
- Sequence of instructions, algorithm is step-by-step
- Python is 3rd most popular language, core principles
- Syntax, runtime, & logic errors, testing & debugging
- Hardware vs. software
- flow – step-by-step, function call, conditional, loop
- IDLE shell, editor, install Python, Hello World
- Interpreter, compiler, Python Standard Library
- Variables, assignment, numeric expr., precedence
- Print function, Strings, concatenation, indexes, in, *
- Interactive programs, if, if-else, if-elif-else, int, float
- Boolean expressions: ==, !=, <, <=, >, >=, not, and, or
- Input function, comparing strings, programming conventions
- variable & function names lowercase, CONSTANTS, indent

The while Statement

A while statement has the following syntax:

```python
while condition:
    statement
```

- If the condition is true, the statement is executed
- Then the condition is evaluated again, and if it is still true, the statement is executed again
- The statement is executed repeatedly until the condition becomes false

Logic of a while Loop

An example of a while statement:

```python
count = 1
while count <= 5:
    print(count)
    count += 1
```

If the condition of a while loop is false initially, the statement is never executed. Statements in the while execute zero or more times.

The for Statement

A for statement has the following syntax:

```python
for variable in sequence:
    statement
```

A variable that is assigned each item in the sequence one at a time
A sequence of items, such as the characters in a string
The statement that is executed for each variable in the sequence
Logic of a for loop

try to get the next item in the sequence

got an item

true

false

statement

true

false

The range Function

The built-in `range` function produces a sequence of values in a specified range, usually used in a for loop:

```python
for i in range(5):
    print(i)
```

When run, this for loop produces:

```
0
1
2
3
4
```

The for loop will iterate as many times as there are items in the sequence.

Augmented Assignments

One way to increment a value is:

```python
value = value + 1
```

An augmented assignment does the same thing with less:

```python
value += 1
```

It's a way to accomplish the same operation using less code. It works with strings, too:

```python
name = 'Joe'
name += ' '
name += 'Blow'
print(name)
```

Augmented Assignments

Compare these regular and augmented assignments:

```python
value = value * 100
value *= 100
value = value / 7
value /= 7
mystr = 'Hello'
mystr = mystr + ', ' + 'World!'
mystr += ', ' + 'World!'
mystr = 'Hello'
```

Example: Palindromes

```python
while True:
    check_me = input('enter a string (b to quit): ')
    if check_me == 'b':
        break
    left = 0
    right = len(check_me) - 1
    palindrome = True
    while left < right and palindrome:
        if check_me[left] != check_me[right]:
            palindrome = False
        else:
            left += 1
            right -= 1
    if palindrome:
        print('"' + check_me + '" is a palindrome."
    else:
        print('"' + check_me + '" is not a palindrome."
```

Turtle Graphics

Turtle Graphics is a framework that lets you easily create graphics

The star, of course, is the turtle:

In your program, you can tell the turtle:

- how far to move
- which direction to turn
- what color to draw with as it moves
Turtle Graphics Example

Here is a basic program:

```python
import turtle
turtle.forward(100)
turtle.left(90)
turtle.forward(70)
```

And this is what you see in the output window when it runs:

---

Turtle Graphics Pen Color & Size

Use `pencolor` and `pensize` to change the color and size of lines

```python
import turtle
turtle.pensize(3)
turtle.pencolor('orange')
turtle.forward(150)
turtle.pencolor('blue')
turtle.left(90)
turtle.forward(70)
turtle.pencolor('red')
turtle.left(90)
turtle.forward(150)
turtle.pencolor('green')
turtle.left(90)
turtle.forward(70)
```

---

Turtle Graphics Set Heading

Use `setheading` to make the turtle go in a specific direction

```python
import turtle
turtle.pensize(2)
turtle.pencolor('purple')
turtle.setheading(45)
turtle.forward(100)
turtle.setheading(180)
turtle.forward(100)
turtle.setheading(20)
turtle.forward(100)
turtle.setheading(160)
turtle.forward(130)
turtle.setheading(270)
turtle.forward(100)
```

---

Turtle Graphics Dots & Circles

Use `dot` and `circle` to draw dots and circles

```python
import turtle
turtle.dot()
turtle.forward(30)
turtle.dot()
turtle.forward(50)
turtle.circle(30)
```

---

Turtle Graphics Setting Position

Use `goto` to put the turtle in a specific position

```python
import turtle
turtle.dot()  
turtle.goto(50, 50)  
turtle.goto(-20, 50)  
turtle.goto(70, -40)  
turtle.goto(-60, -90)  
turtle.goto(100, 80)
```

---

Turtle Graphics Filling Shapes

Use `fillcolor`, `begin_fill` and `end_fill` to fill in shapes with color

```python
import turtle
turtle.penup()  
turtle.goto(-70, 0)  
turtle.pendown()  
turtle.pensize(3)
turtle.fillcolor('orange')  
turtle.begin_fill()  
turtle.circle(50)  
turtle.end_fill()
```
Turtle Graphics Drawing Speed

Use `speed` to speed up or slow down the turtle (default is 3)

```python
import turtle
turtle.speed(6)
turtle.speed('fast')
```

Speeds can be:
- Any number 0 to 10
- 1 is slowest
- 10 is fastest
- 0 turns off animation
- 'fastest' 0
- 'fast' 10
- 'normal' 6
- 'slow' 3
- 'slowest' 1

Use `hideturtle` to go faster!

```python
import turtle
turtle.hideturtle()
```