Stepping through a ‘for’ loop
The for loop

• **for loop**: Repeats a set of statements over a group of values.

-Syntax:

```python
for variableName in groupOfValues:
    statements
```

• We indent the statements to be repeated with tabs or spaces.
• `variableName` gives a name to each value, so you can refer to it in the `statements`.
• `groupOfValues` can be a range of integers, specified with the `range` function.

-Example:

```python
for x in range(1, 6):
    print x, "squared is", x * x
```

Output:

1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
The `range` function specifies a range of integers:

- `range(start, stop)` - the integers between `start` (inclusive) and `stop` (exclusive)
  
- It can also accept a third value specifying the change between values.
  - `range(start, stop, step)` - the integers between `start` (inclusive) and `stop` (exclusive) by `step`

Example:
```python
for x in range(5, 0, -1):
    print x
print "Blastoff!"
```

Output:
```
5
4
3
2
1
Blastoff!
```
Cumulative loops

• Some loops incrementally compute a value that is initialized outside the loop. This is sometimes called a cumulative sum.

```python
sum = 0
for i in range(1, 11):
    sum = sum + (i * i)
print "sum of first 10 squares is", sum
```

Output:
sum of first 10 squares is 385

• Exercise: Write a Python program that computes the factorial of an integer.
At the beginning - run module

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name 
print "The end."
```
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello,", name  
print "The end."
Look at the list

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello,", name
print "The end."
```
Is it empty? No. Start with the first object

for name in ['Andrew', 'Tsanwani', 'Arno', 'Tebogo']:
    print "Hello,", name
print "The end."
Assign the first object to the variable ‘name’

```
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name 
print "The end."
```
Then start the first line of the code block

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```

Variable Names

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Objects

<table>
<thead>
<tr>
<th>“Andrew”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
print the string object "Hello," and the value of the variable with name ‘name’

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```

Hello, Andrew
The print statement is finished. Python is at the end of the code block...

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello,", name
print "The end."
```
... so go to the ‘for’ statement again

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name  
print "The end."
```
Move the list pointer forward by one

for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name 
print "The end."
Is it past the end? No, it’s on the second item.

for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name 
print "The end."
Assign the second object to the variable ‘name’

```
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello,", name  
print "The end."
```

Create the string object "Tsanwani" and assign it to the variable named ‘name’
Then start the first line of the code block

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," + name
print "The end."
```

Variable Names

- name

Objects

- "Andrew"
- "Tsanwani"
print the string object "Hello," and the value of the variable with name 'name'

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```

Variable Names

- `name`

Objects

- "Andrew"
- "Tsanwani"

Hello, Andrew
Hello, Tsanwani
The print statement is finished. Python is at the end of the code block...

```
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name  
print "The end."
```
... so go to the ‘for’ statement again

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello,", name
print "The end."
```
Move the list pointer forward by one

for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello,,", name  
print "The end."
Is it past the end? No, it’s on the third item.

for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
Assign the third object to the variable ‘name’

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," \n    name
print "The end."
```

Create the string object “Arno” and assign it to the variable named ‘name’
Then start the first line of the code block

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name  
print "The end."
```
print the string object “Hello,” and the value of the variable with name ‘name’

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```

**Variable Names**
- name

**Objects**
- "Andrew"
- "Tsanwani"
- "Arno"

Hello, Andrew
Hello, Tsanwani
Hello, Arno
The print statement is finished. Python is at the end of the code block...

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello,", name
print "The end."
```
... so go to the ‘for’ statement again

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```
Move the list pointer forward by one

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```

Variable Names

- `name`

Objects

- "Andrew"
- "Tsanwani"
- "Arno"
Is it past the end? No, it’s on the fourth item.

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```
Assign the fourth object to the variable ‘name’

```
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name  
print "The end."
```

Create the string object "Tebogo" and assign it to the variable named ‘name’
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
print the string object “Hello,” and the value of the variable with name ‘name’

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello,", name

print "The end."
```

**Variable Names**
- name

**Objects**
- "Andrew"
- "Tsanwani"
- "Arno"
- "Tebogo"

Hello, Andrew
Hello, Tsanwani
Hello, Arno
Hello, Tebogo
The print statement is finished. Python is at the end of the code block...

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```
... so go to the ‘for’ statement again

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name
print "The end."
```
Move the list pointer forward by one

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name 
print "The end."
```
Is it past the end? Yes!

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```
So skip past the code block to the next statement.

```
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello,", name 
print "The end."
```
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," + name
print "The end."
It prints the string “The end.”

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:
    print "Hello," , name
print "The end."
```

Hello, Andrew
Hello, Tsanwani
Hello, Arno
Hello, Tebogo
The end.
Python looks for the next statement...

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
    print "Hello," , name 
print "The end."
```
... but there isn’t any, so Python stops.

```python
for name in ["Andrew", "Tsanwani", "Arno", "Tebogo"]:  
  print "Hello," , name 
print "The end."
```
Final Program Output

Hello, Andrew
Hello, Tsanwani
Hello, Arno
Hello, Tebogo
The end.