

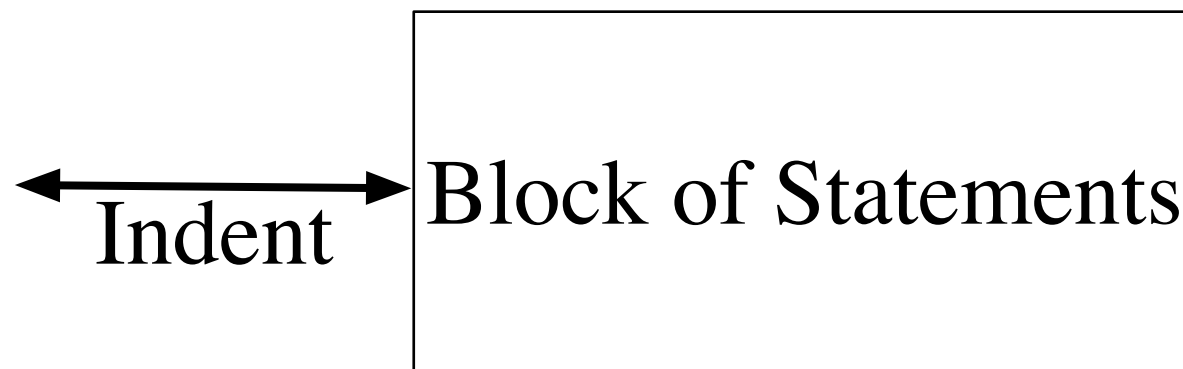
Python

for-loops

Repetition

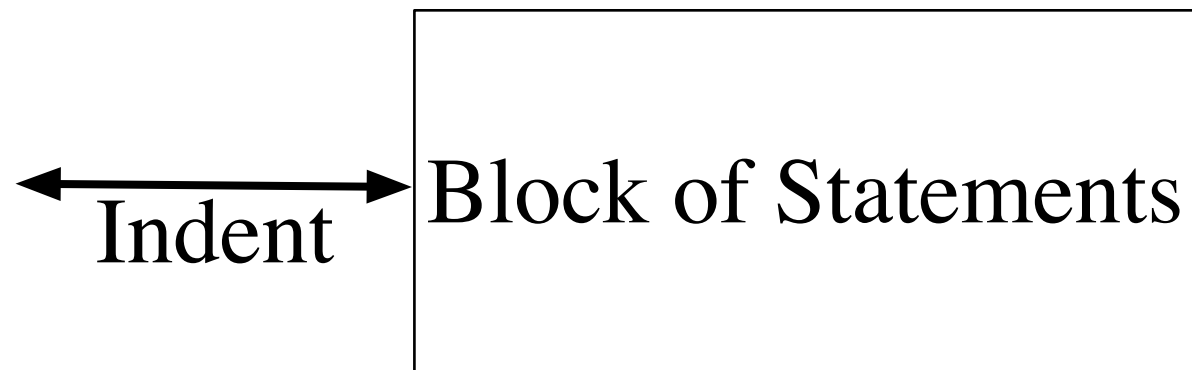
- Python allows the same block of statements to be repeatedly executed.
- Python iterates over a list such as the range of integers from 0 to $n-1$.
- For loop prototype is

```
for i in range (n) :
```



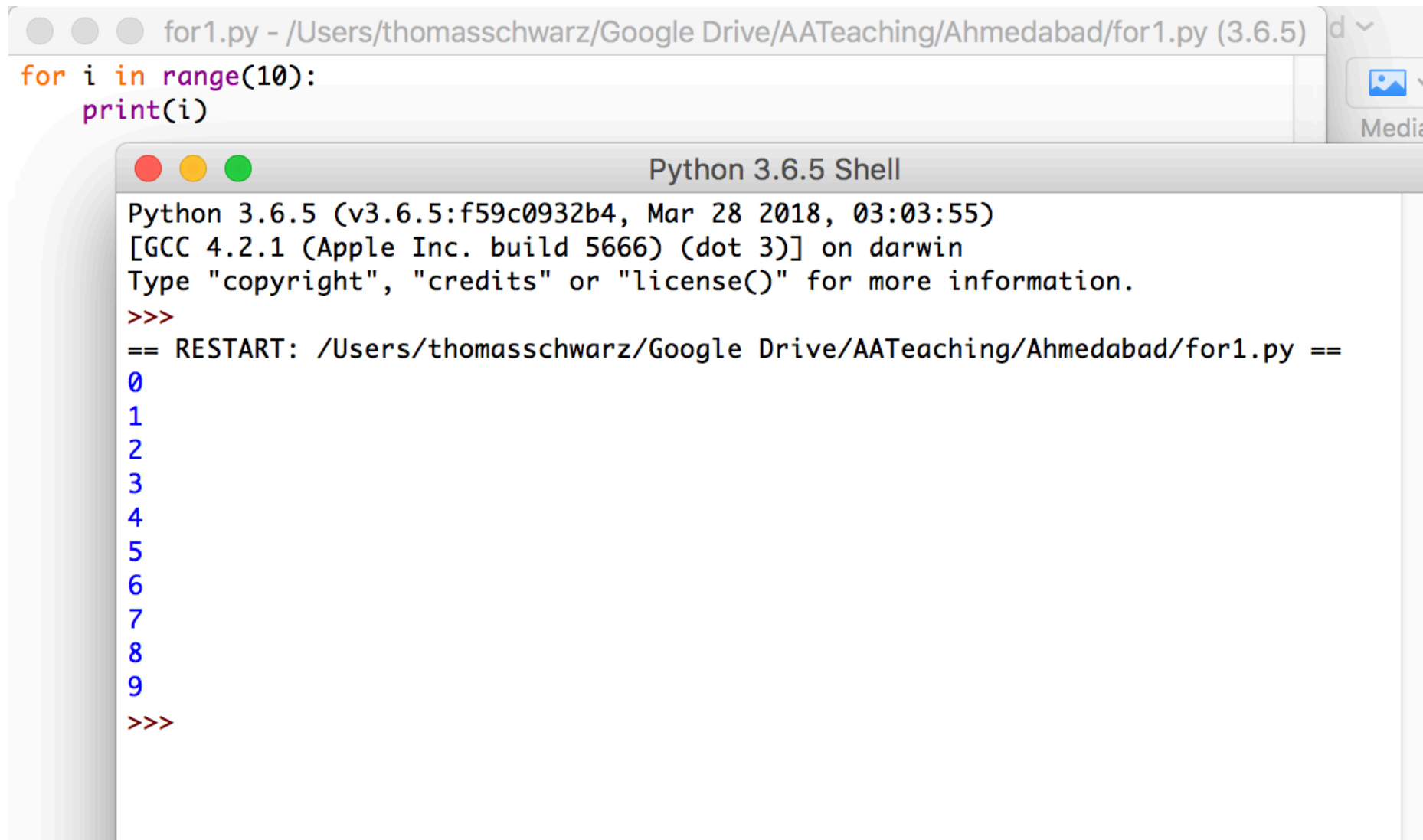
Repetition

```
for i in range (n) :
```



- Keywords for, in, :
- `range (n)` is a short cut for a list `0, 1, 2, ..., n-1`
- `i` is a variable
 - First time through block, `i` is 0, then `i` is 1, ...
- Indented block of statements

Example



The image shows a code editor window with a Python script and a terminal window showing its execution. The code editor window has a title bar with three colored circles (red, yellow, green) and the text "for1.py - /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/for1.py (3.6.5)". The code in the editor is:

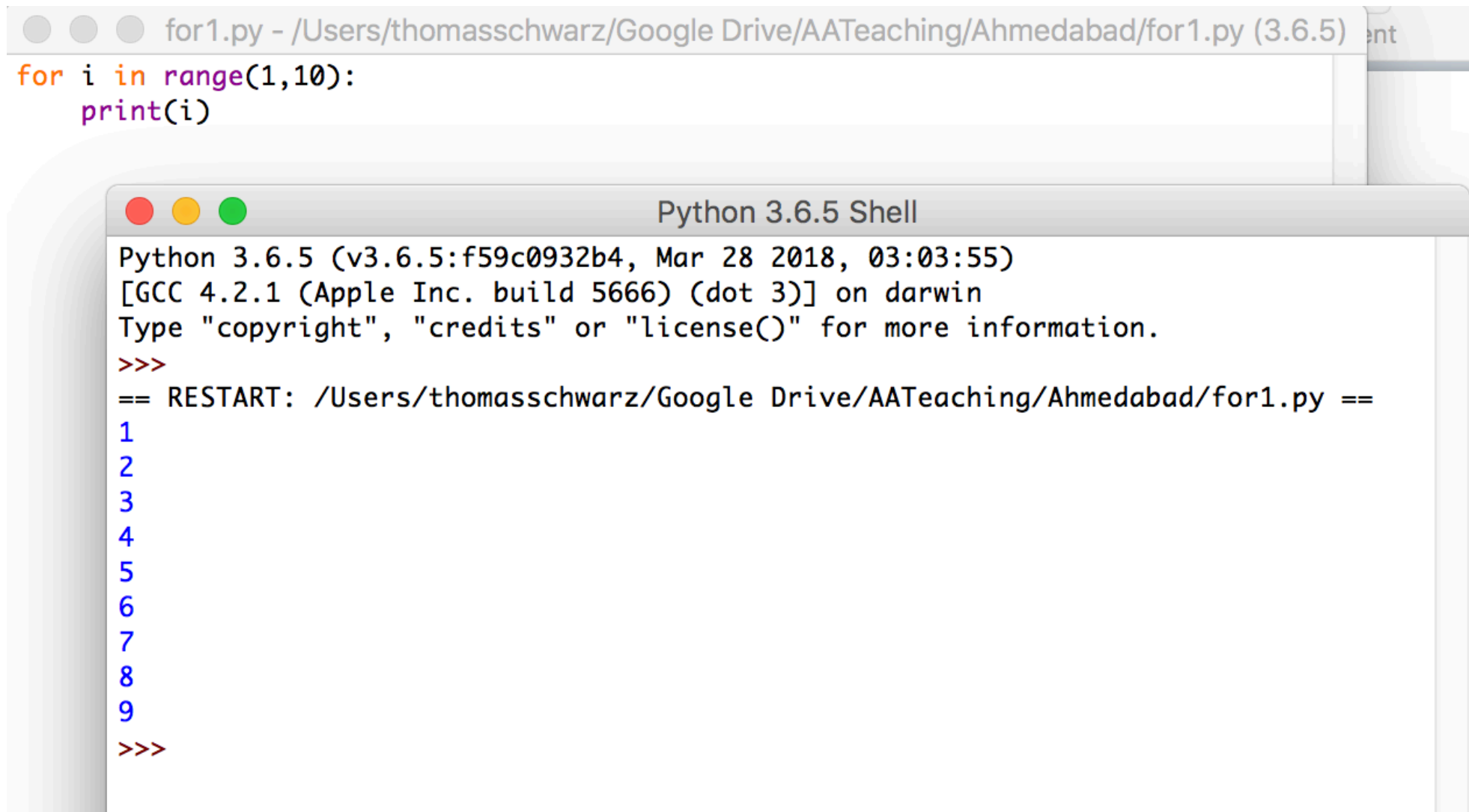
```
for i in range(10):  
    print(i)
```

The terminal window has a title bar with three colored circles (red, yellow, green) and the text "Python 3.6.5 Shell". The output in the terminal is:

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55)  
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin  
Type "copyright", "credits" or "license()" for more information.  
>>>  
== RESTART: /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/for1.py ==  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
>>>
```

Range

- Range allows you to specify a start in addition to the stop value.
- `range(1, 10)` : start value 1, stop value 10



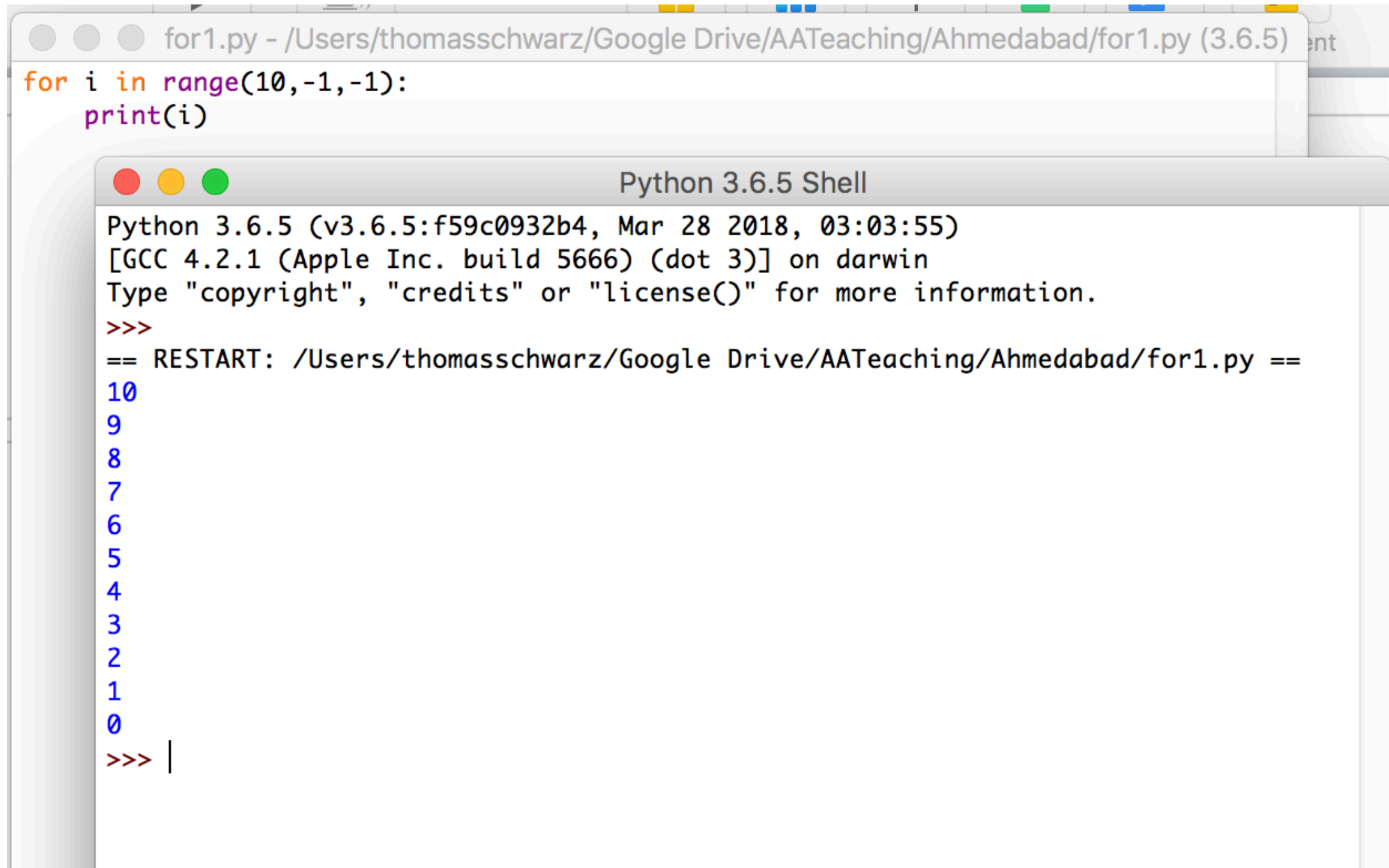
```
for1.py - /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/for1.py (3.6.5) ent
for i in range(1,10):
    print(i)

Python 3.6.5 Shell
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/for1.py ==
1
2
3
4
5
6
7
8
9
>>>
```

Range

- You can also specify the stride
 - `range(10, -1, -1)`
 - start with 10
 - stop before -1 (i.e. with 0)
 - change by going down by one
 - `range(0, 10, 2)`
 - start with 0
 - go up in twos : 0, 2, 4, 6, 8
 - stop when stepping on or over 10

Example

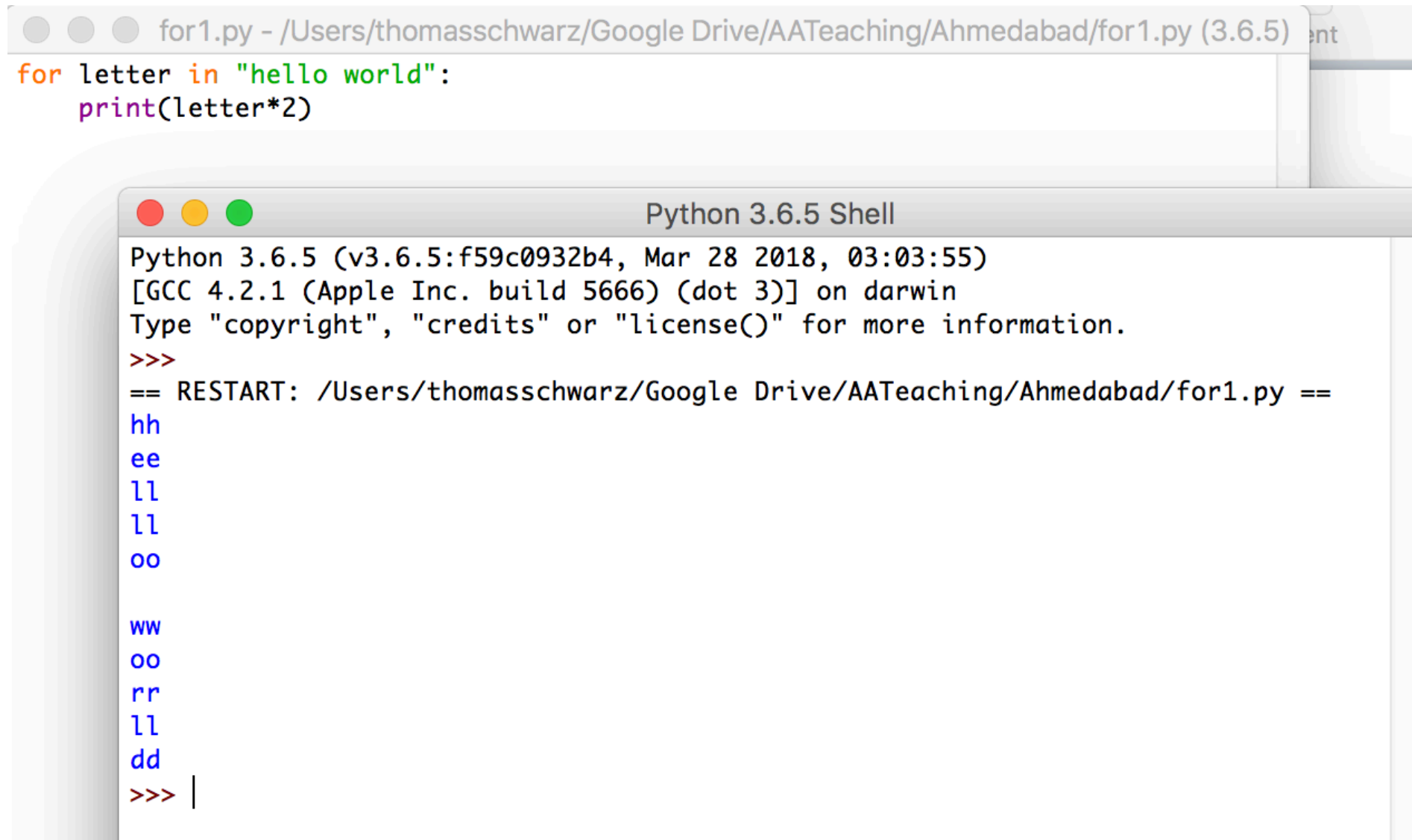


```
for1.py - /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/for1.py (3.6.5) ent
for i in range(10,-1,-1):
    print(i)

Python 3.6.5 Shell
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/for1.py ==
10
9
8
7
6
5
4
3
2
1
0
>>> |
```

Other iterations

- If you use for on a string, you walk through all of the letters of the string:



```
for1.py - /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/for1.py (3.6.5) ent
for letter in "hello world":
    print(letter*2)

Python 3.6.5 Shell
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/for1.py ==
hh
ee
ll
ll
oo

ww
oo
rr
ll
dd
>>> |
```


Calculating Sums

- For loops are handy to calculate mathematical sums
 - Geometric series:
 - Calculate $\frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \dots + \frac{1}{2^{10}}$
 - Determine iterator needs to run from 0 to 10 (inclusive)
 - `for i in range(11):`
 - Need to accumulate fractions in a sum
 - Just don't call it "sum", because "sum" has another meaning

Calculating Sums

geometric.py - /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/Solu...

```
accu = 0
for i in range(11):
    accu += 1/2**i
print(accu)
```

Python 3.6.5 Shell

Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55)

[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: /Users/thomasschwarz/Google Drive/AATeaching/Ahmedabad/Solutions/geometric.py

1.9990234375

>>>

Calculating Sums

- Admittedly, we could have used Mathematics instead
 - The sum is 1.1111111111 in binary.
 - Add $1/2^{*}10$ or 0.0000000001 in binary and we get 2.
 - Thus, the sum is $2 - 1/2^{*}10$

Drawing Pictures

- We can use the index in a for loop in order to draw contours
- The trick is to use string repetition instead of drawing each line separately.

```
for2.py - /Users/thomasschwarz/Google Drive/AATeac
for i in range(0,6):
    print((5-i)*" "+2*i*"*"+"*")
for i in range(5,-1,-1):
    print((5-i)*" "+2*i*"*"+"*")
```

```
Python 3.6.5 S
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018,
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on
Type "copyright", "credits" or "license()" for
>>>
RESTART: /Users/thomasschwarz/Google Drive/AA
PY
    *
   ***
  *****
 *****
*****
*****
*****
*****
 *****
  *****
   ***
    *
>>> |
```


While Loops

Python

While Loops

- Form of the while loop:

```
while condition :
```

←————→ Statement Block
Indent

- Keyword is while
- Condition needs to evaluate to either True or False
 - Condition is a boolean

While Loop Conditions

- Statement block is executed as long as condition is valid.
 - Allows the possibility of infinite loops

Apple Inc.
One Infinite Loop
Cupertino, CA 95014
(408) 606-5775

`while` condition :

←→ Statement Block
Indent

An Infinite Loop

```
while True:  
    print("Hello World")
```

If this happens to you, your
process.

Example: Interest Payments

- Calculate the interest on an outstanding loan
 - amount
 - rate (percentage per year, compounded monthly)
 - `round(rate / (12 * 100) * amount, 2)`

Example: Interest Payments

- Payment plan:
 - You take out an amount
 - You pay every month a monthly payment:
 - interest on the outstanding amount
 - You pay a repayment

Example: Interest Payments

- Let's print out a payment plan
- Use a bunch of variables:
 - ```
amount = 100000
monthly = 1000
rate = 5
count = 0
```
  - amount -- what's left to pay
  - monthly payment
  - count : count of months

# Example: Interest Payments

- We calculate the new outstanding amount and current interest and repayment

```
while (amount > monthly):
 interest = round(amount*rate/100/12,2)
 repayment = round(monthly - interest,2)
 amount = round(amount-repayment,2)
 count += 1
 print(count, amount, interest, repayment)
```