Count to 10

```
print( "1" )
print( "2" )
print( "3" )
print( "4" )
.
.
.
print( "10" )
```

While Loop

```
while condition:
  body
```

While Loop

```
while condition:
  body
```

an expression that evaluates to True/False

While Loop

```
while condition:
  body
```

one or more lines of code
(indented, just like a function body)

Conditional Operators

```
x = 5
x < 4       -> False
```

Conditional Operators

```
x = 5
x < 6       -> True
```

Conditional Operators

```
<  less than
>  greater than
== equal to
>= greater than or equal to
<= less than or equal to
!= not equal to
```

Conditional Operators

```
= != ==

x = 5

5 == x
True

5 = x
Error
```

Count to 10

```
while ???:
    print(n)
    ???
```

Count to 10

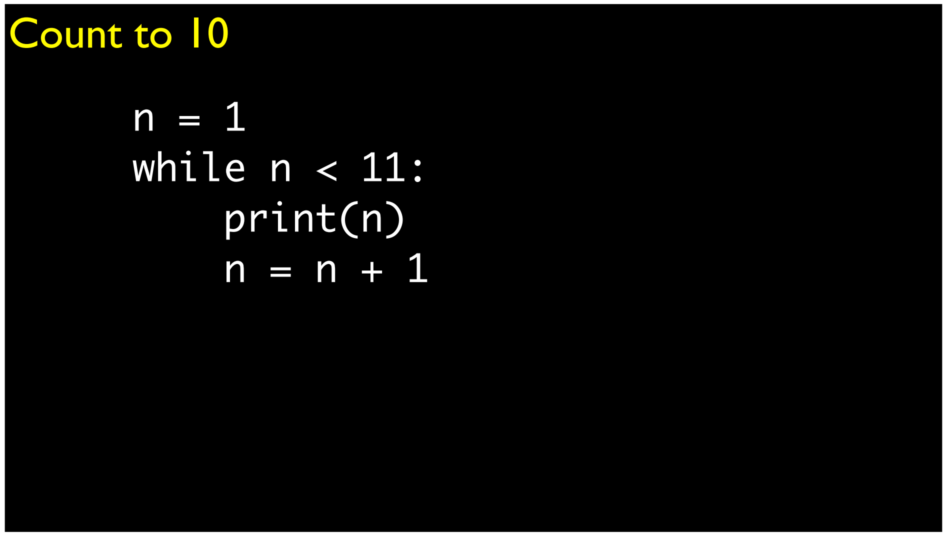
```
n = 1
while ???:
    print(n)
    ???
```

Count to 10

```
n = 1
while ???:
    print(n)
    n = n + 1
```

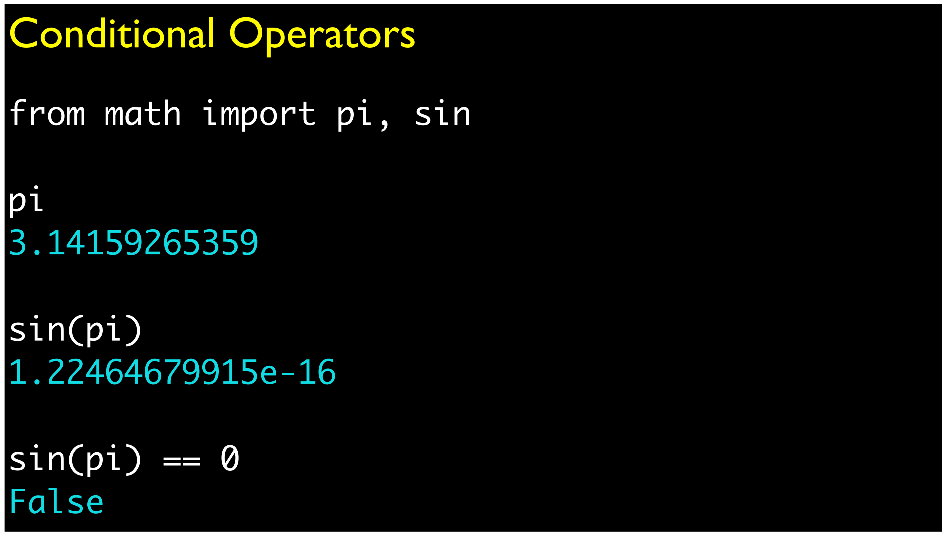
Count to 10

```
n = 1
while n <= 10:
    print(n)
    n = n + 1
```



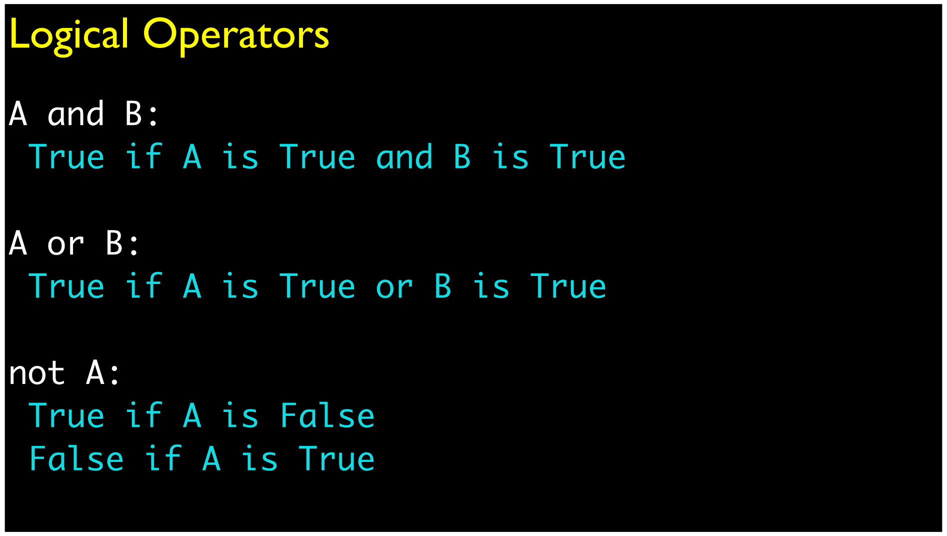
Count to 10

```
n = 1
while n < 11:
    print(n)
    n = n + 1
```



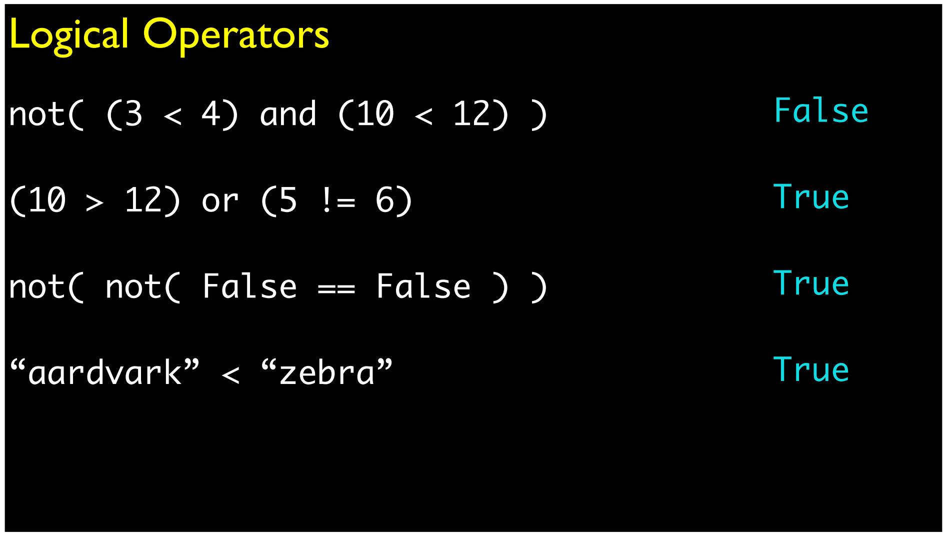
Conditional Operators

```
from math import pi, sin

pi
3.14159265359

sin(pi)
1.22464679915e-16

sin(pi) == 0
False
```



Logical Operators

```
A and B:
 True if A is True and B is True

A or B:
 True if A is True or B is True

not A:
 True if A is False
 False if A is True
```



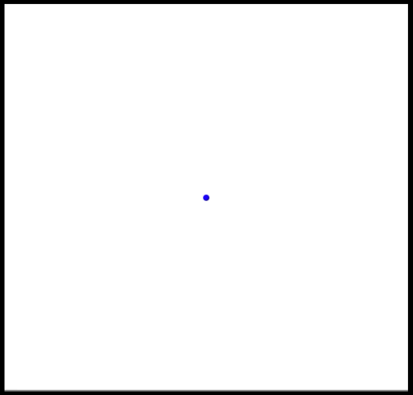
Logical Operators

```
not( (3 < 4) and (10 < 12) )            False

(10 > 12) or (5 != 6)                   True

not( not( False == False ) )            True

“aardvark” < “zebra”                    True
```

Logical Operators

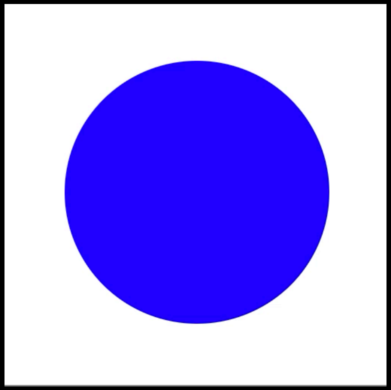
`not( (3 < 4) and (10 < 12) )`	False
`(10 > 12) or (5 != 6)`	True
`not( not( False == False ) )`	True
`“aardvark” < “zebra”`	True
`True > False`	True

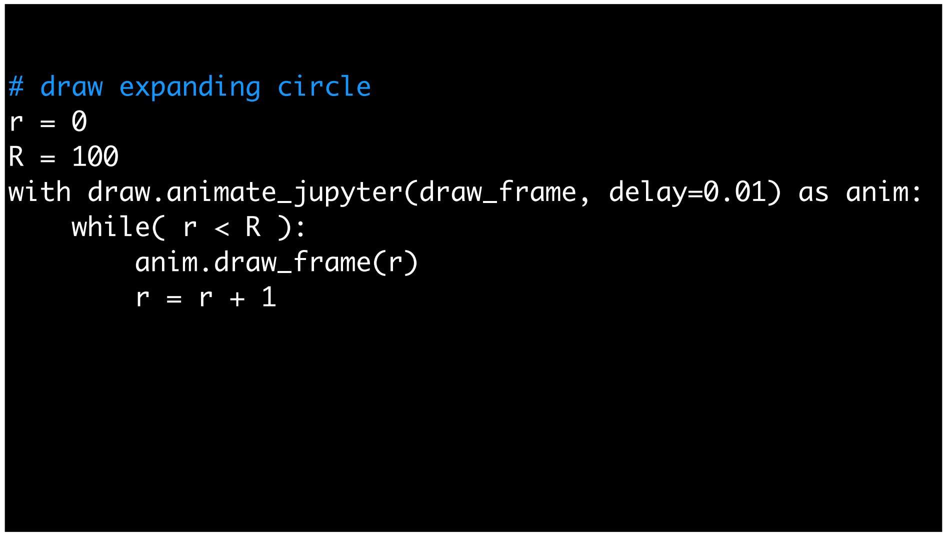
```
# --- DRILL ---
# write some code that generates the following
```



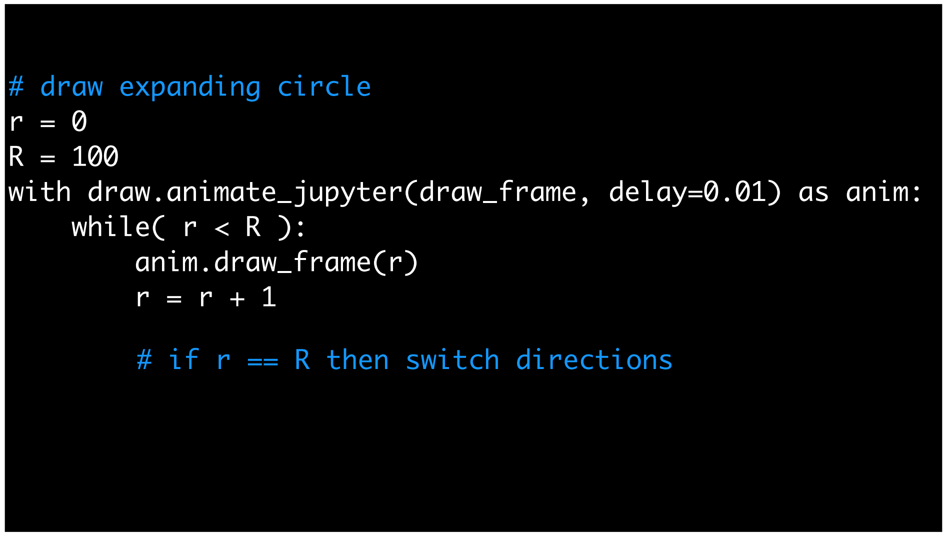
```
import drawSvg as draw

# draw expanding circle
r = 0
R = 100
with draw.animate_jupyter(draw_frame, delay=0.01) as anim:
    while( r < R ):
        anim.draw_frame(r)
        r = r + 1
```

```
# --- DRILL ---
# write some code that generates the following
```

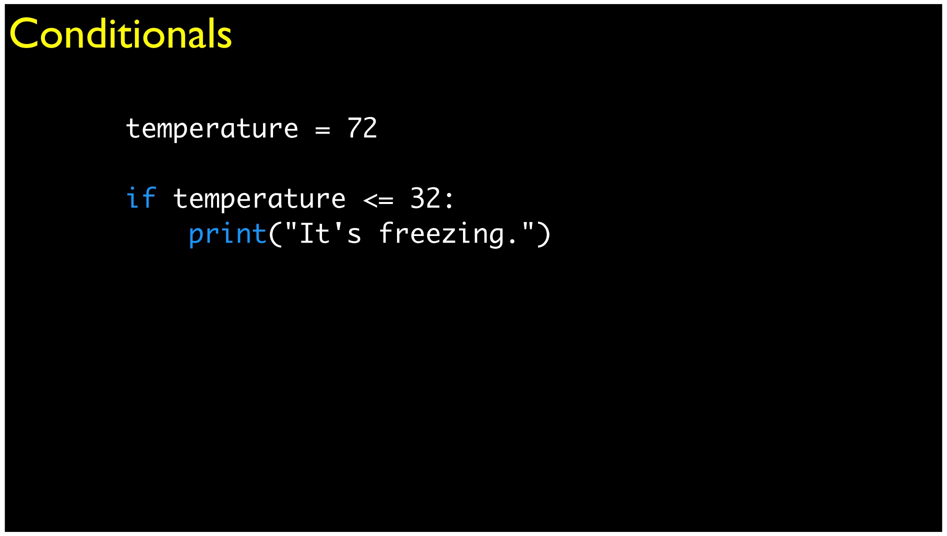




```python
# draw expanding circle
r = 0
R = 100
with draw.animate_jupyter(draw_frame, delay=0.01) as anim:
    while( r < R ):
        anim.draw_frame(r)
        r = r + 1
```

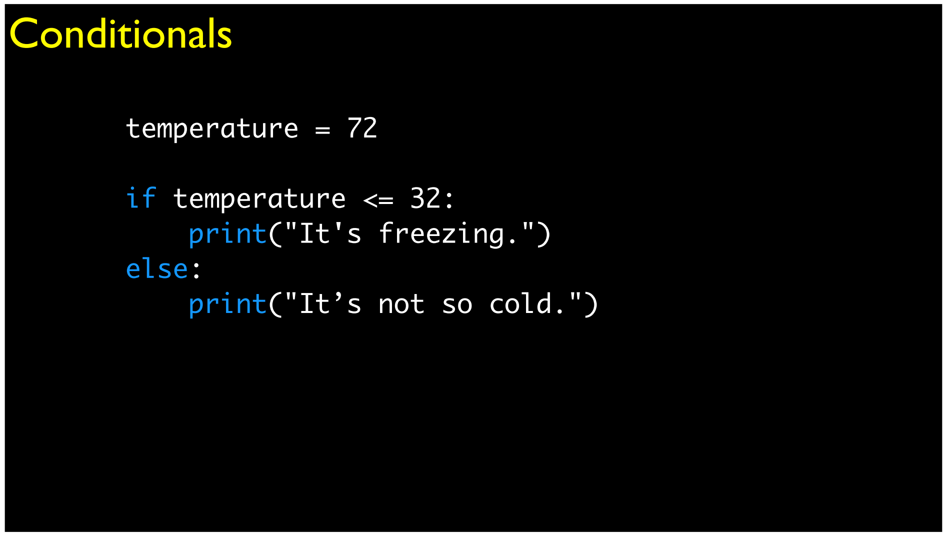


```python
# draw expanding circle
r = 0
R = 100
with draw.animate_jupyter(draw_frame, delay=0.01) as anim:
    while( r < R ):
        anim.draw_frame(r)
        r = r + 1

        # if r == R then switch directions
```



Conditionals

```python
temperature = 72

if temperature <= 32:
    print("It's freezing.")
```



Conditionals

```python
temperature = 72

if temperature <= 32:
    print("It's freezing.")
else:
    print("It’s not so cold.")
```

Conditionals

```python
temperature = 72

if temperature <= 32:
    print("It's freezing.")
elif temperature <= 50:
    print("It's cool.")
elif temperature <= 75:
    print("It's warm.")
else:
    print("It's hot.")
```

Conditionals

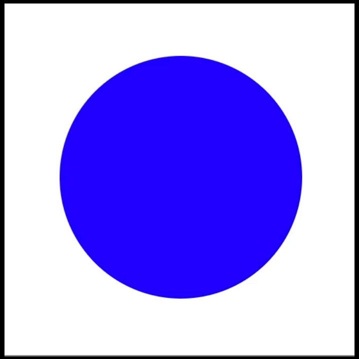
```python
x = 1
if x > 0:
    print("positive")
    x = -1 * x
elif x < 0:
    print("negative")
else:
    print("zero")

print( x )
```

Conditionals

```python
x = 1
if x > 0:
    print("positive")
    x = -1 * x
elif x < 0:
    print("negative")
else:
    print("zero")

print( x )
```

```
positive
-1
```

```python
# --- DRILL ---
# write some code that generates the following
```

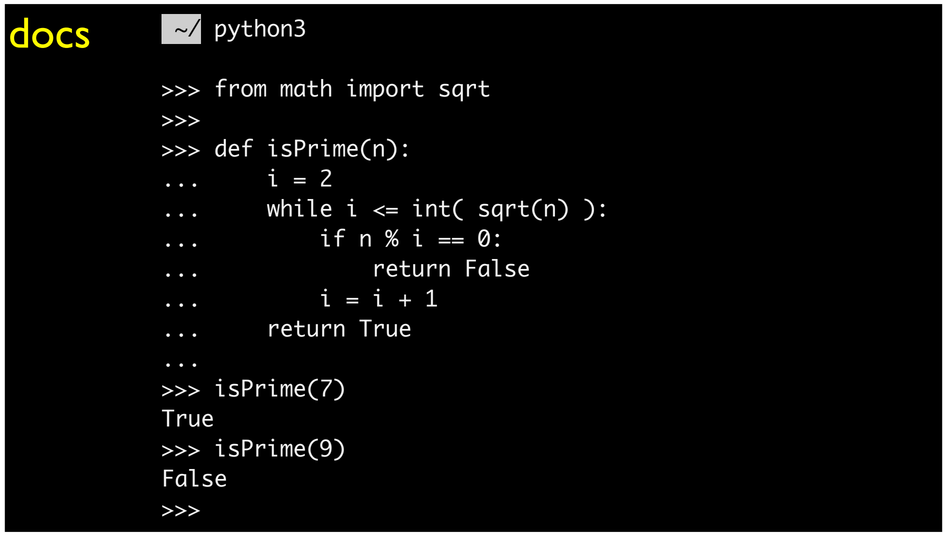


```
# Draw expanding/contracting circle
r = 0 # current radius
R = 100 # maximum radius
sign = 1 # direction (1: expand; -1: contract)
with draw.animate_jupyter(draw_frame, delay=0.01) as anim:
    while ???:
        anim.draw_frame(r)
        if sign == 1:
            # expand circle
        else:
            # contract circle

        if circle is fully expanded or contracted:
         reverse direction
```

```
# Draw expanding/contracting circle
r = 0 # current radius
R = 100 # maximum radius
sign = 1 # direction (1: expand; -1: contract)
with draw.animate_jupyter(draw_frame, delay=0.01) as anim:
    while ???:
        anim.draw_frame(r)
        if sign == 1):
            r = r + 1
        else:
            r = r - 1

        if circle is fully expanded or contracted:
         reverse direction
```

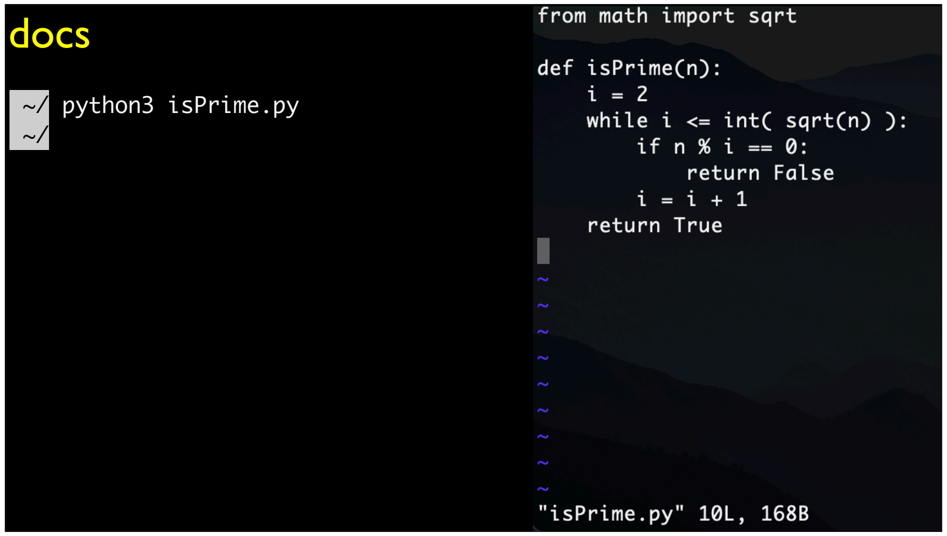
```
# Draw expanding/contracting circle
r = 0 # current radius
R = 100 # maximum radius
sign = 1 # direction (1: expand; -1: contract)
with draw.animate_jupyter(draw_frame, delay=0.01) as anim:
    while( ??? ):
        anim.draw_frame(r)
        if( sign == 1 ):
            r = r + 1
        else:
            r = r - 1

        if r > R or r < 0:
            sign = -1 * sign
```

```
# Draw expanding/contracting circle
r = 0 # current radius
R = 100 # maximum radius
sign = 1 # direction (1: expand; -1: contract)
with draw.animate_jupyter(draw_frame, delay=0.01) as anim:
    while True:
        anim.draw_frame(r)
        if sign == 1:
            r = r + 1
        else:
            r = r - 1

        if r > R or r < 0:
            sign = -1 * sign
```



docs

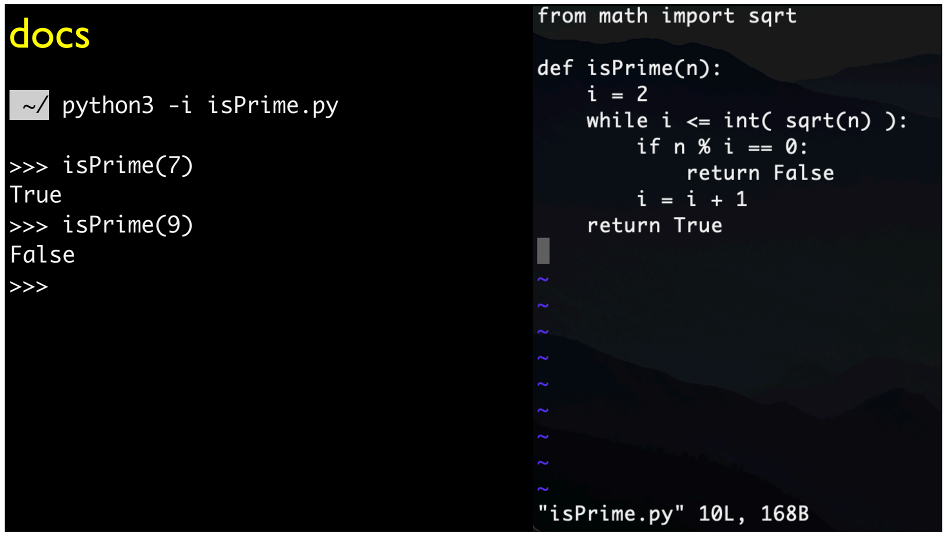
```
~/ python3

>>> from math import sqrt
>>>
>>> def isPrime(n):
...     i = 2
...     while i <= int( sqrt(n) ):
...         if n % i == 0:
...             return False
...         i = i + 1
...     return True
...
>>> isPrime(7)
True
>>> isPrime(9)
False
>>>
```



docs

```
~/ python3 isPrime.py
~/
```

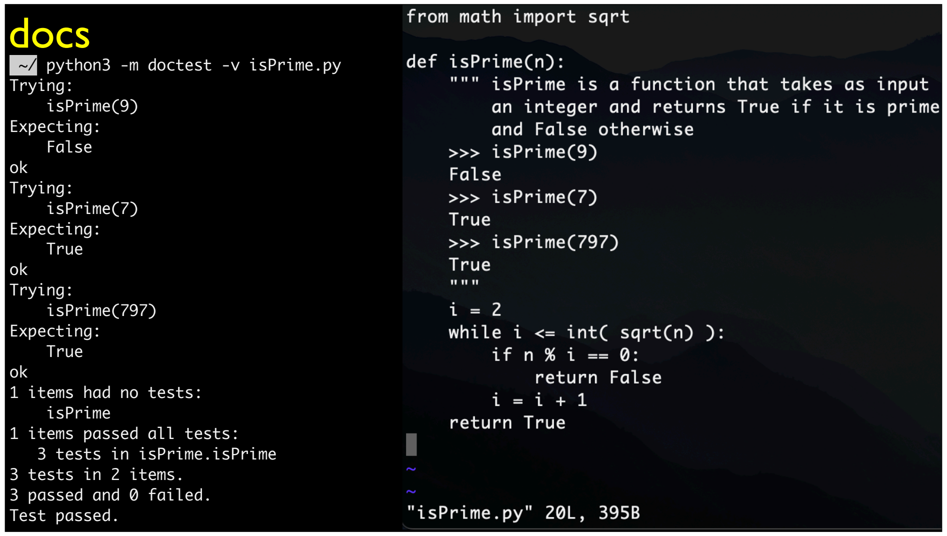
```
from math import sqrt

def isPrime(n):
    i = 2
    while i <= int( sqrt(n) ):
        if n % i == 0:
            return False
        i = i + 1
    return True

~
~
~
~
~
~
~
~
~
"isPrime.py" 10L, 168B
```



docs

```
~/ python3 -i isPrime.py

>>> isPrime(7)
True
>>> isPrime(9)
False
>>>
```

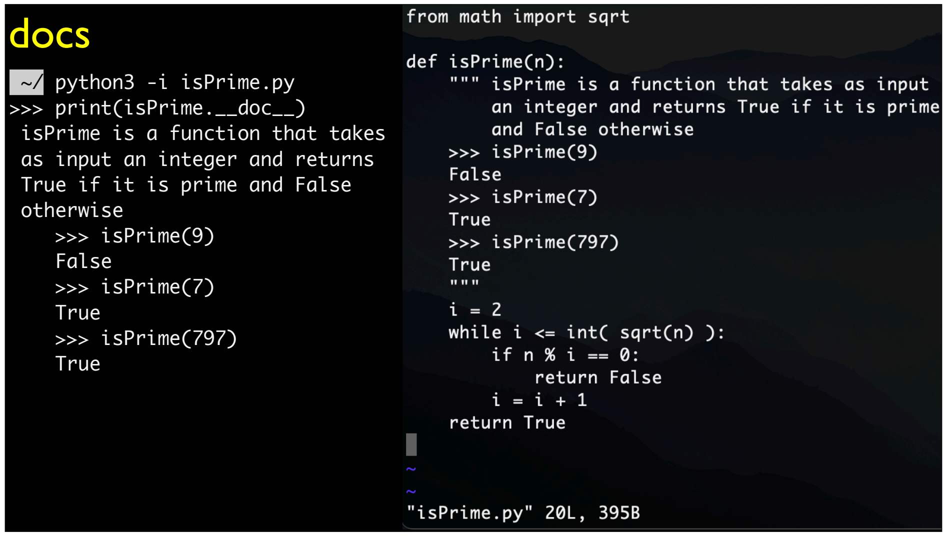
```
from math import sqrt

def isPrime(n):
    i = 2
    while i <= int( sqrt(n) ):
        if n % i == 0:
            return False
        i = i + 1
    return True

~
~
~
~
~
~
~
~
~
"isPrime.py" 10L, 168B
```



docs

```
~/ python3 -m doctest -v isPrime.py
Trying:
    isPrime(9)
Expecting:
    False
ok
Trying:
    isPrime(7)
Expecting:
    True
ok
Trying:
    isPrime(797)
Expecting:
    True
ok
1 items had no tests:
    isPrime
1 items passed all tests:
   3 tests in isPrime.isPrime
3 tests in 2 items.
3 passed and 0 failed.
Test passed.
```

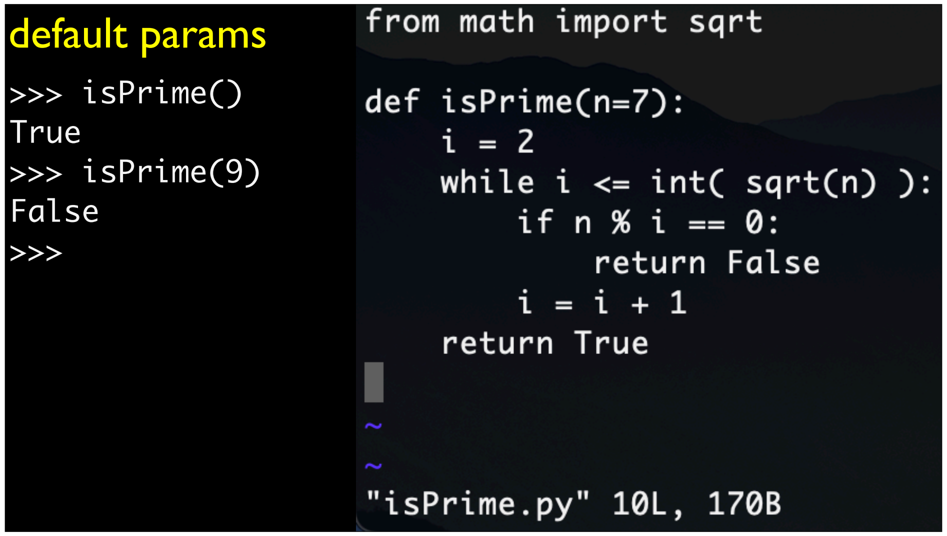
```
from math import sqrt

def isPrime(n):
    """ isPrime is a function that takes as input
        an integer and returns True if it is prime
        and False otherwise
    >>> isPrime(9)
    False
    >>> isPrime(7)
    True
    >>> isPrime(797)
    True
    """
    i = 2
    while i <= int( sqrt(n) ):
        if n % i == 0:
            return False
        i = i + 1
    return True

~
~
"isPrime.py" 20L, 395B
```



docs

```
~/ python3 -i isPrime.py
>>> print(isPrime.__doc__)
 isPrime is a function that takes
 as input an integer and returns
 True if it is prime and False
 otherwise
    >>> isPrime(9)
    False
    >>> isPrime(7)
    True
    >>> isPrime(797)
    True
```

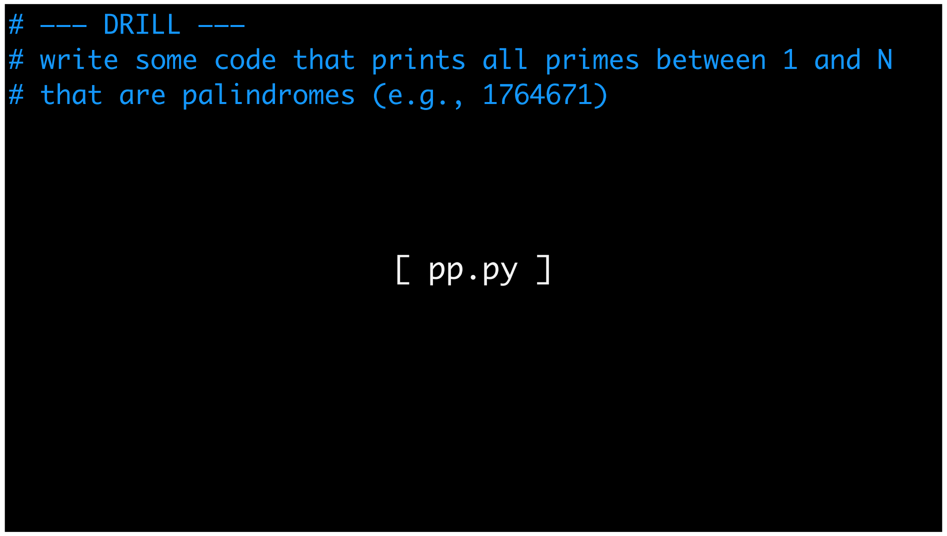
```
from math import sqrt

def isPrime(n):
    """ isPrime is a function that takes as input
        an integer and returns True if it is prime
        and False otherwise
    >>> isPrime(9)
    False
    >>> isPrime(7)
    True
    >>> isPrime(797)
    True
    """
    i = 2
    while i <= int( sqrt(n) ):
        if n % i == 0:
            return False
        i = i + 1
    return True

~
~
"isPrime.py" 20L, 395B
```



default params

```
>>> isPrime()
True
>>> isPrime(9)
False
>>>
```

```
from math import sqrt

def isPrime(n=7):
    i = 2
    while i <= int( sqrt(n) ):
        if n % i == 0:
            return False
        i = i + 1
    return True

~
~
"isPrime.py" 10L, 170B
```



```
# --- DRILL ---
# write some code that prints all primes between 1 and N
# that are palindromes (e.g., 1764671)
```

[pp.py]