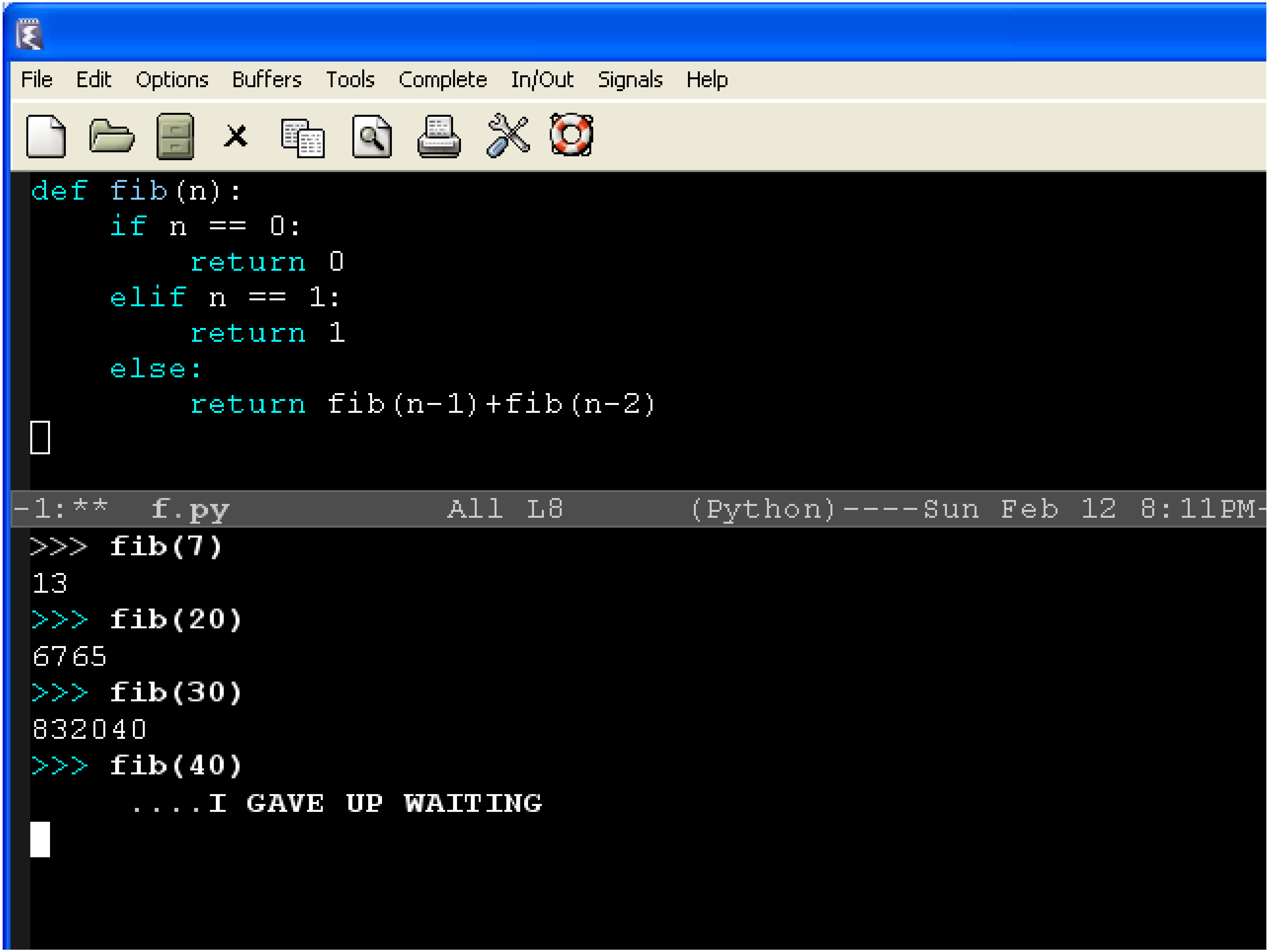
```
def bin(n):
    if n == 0: return ‘0’
    elif n==1: return ‘1’
    else:
        return bin(n // 2) + bin(n % 2)
```


File Edit Options Buffers Tools Complete In/Out Signals Help
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1)+fib(n-2)
-1:** f.py All L8 (Python)----Sun Feb 12 8:11PM-
>>> fib(7)
13
>>> fib(20)
6765
>>> fib(30)
832040
>>> fib(40)
....I GAVE UP WAITING

Framework for abstraction

	Procedures	Data
Primitives	+, *, ==, …	numbers, strings
Means of combination	if, while, … composition, e.g., can write 3*(4+7)	lists, dictionaries
Means of abstraction	def	
Means of capturing common patterns		

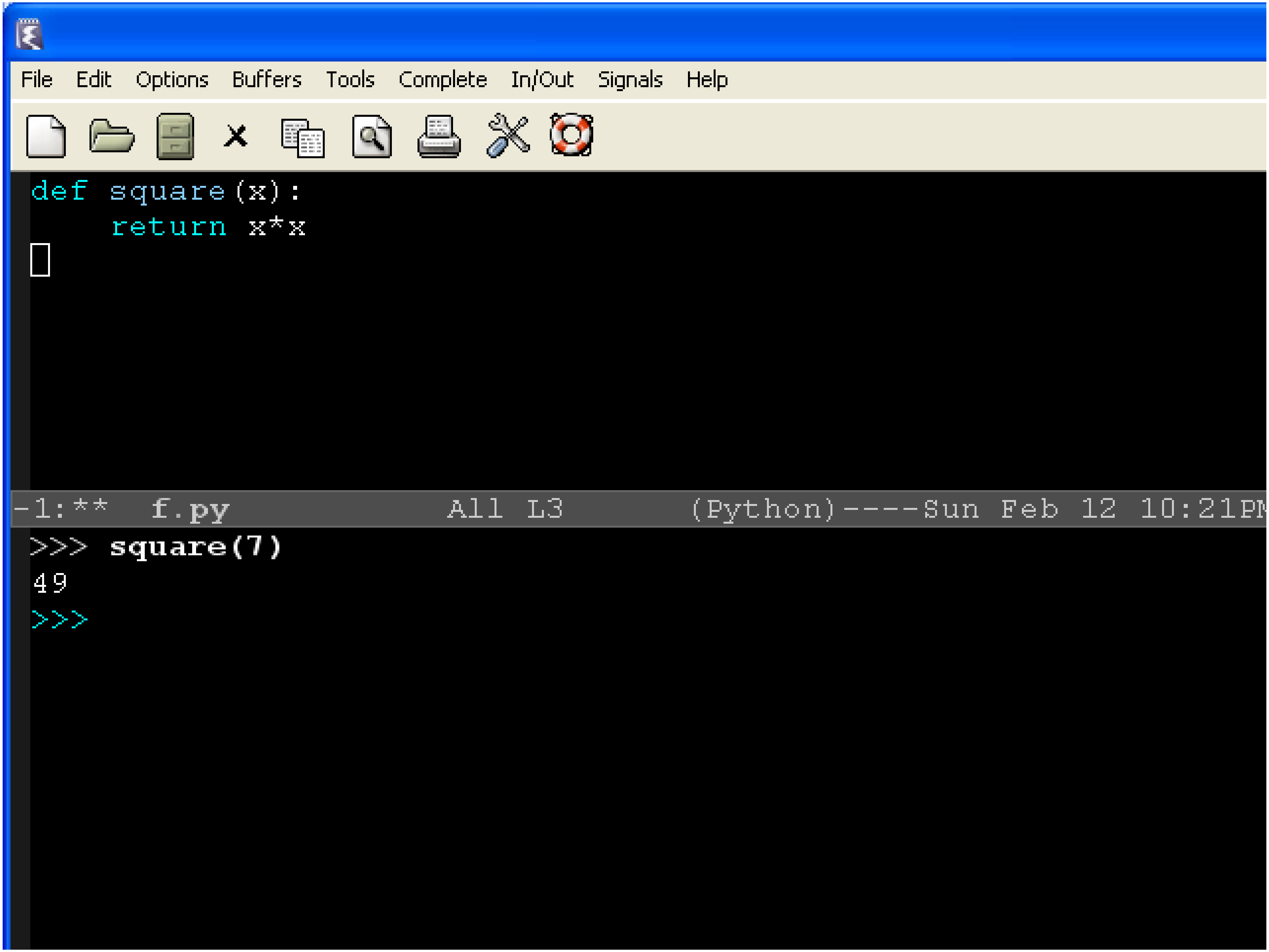
Python dictionaries

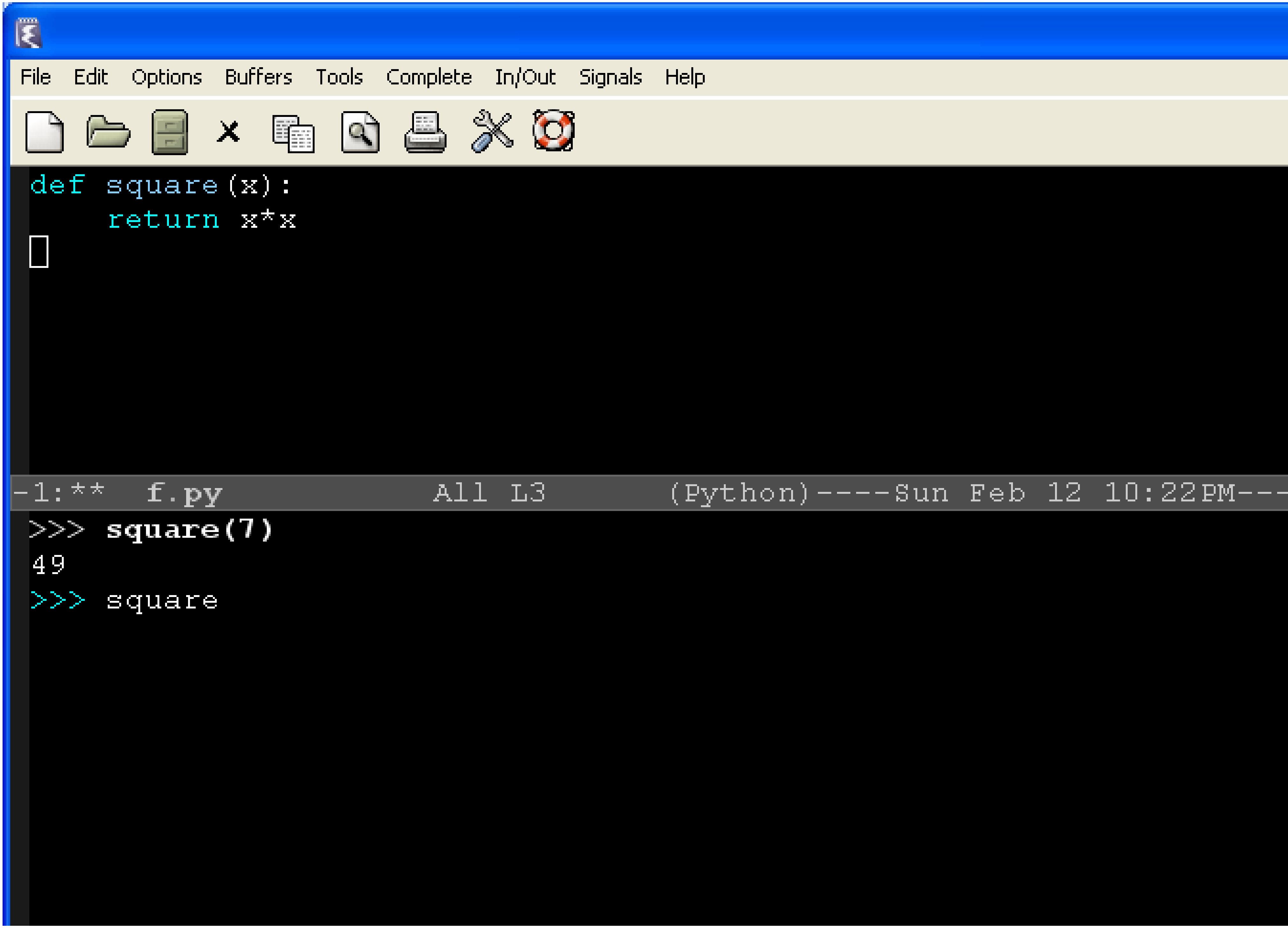
- A dictionary is a table where you can store values under keys.
- The keys can be anything. The values can be anything.

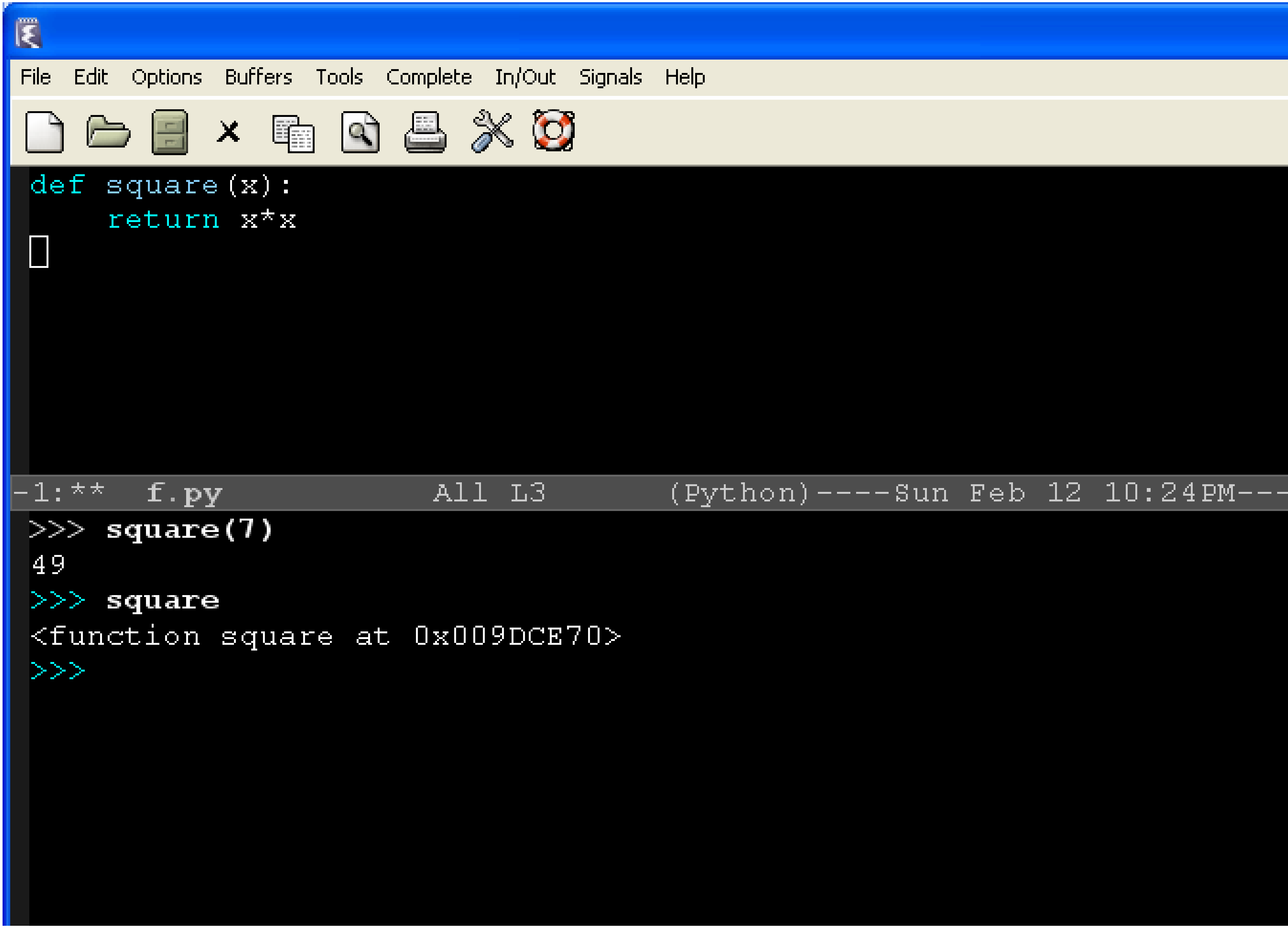
```
> d={ }                 #make a new dictionary
> d[17]='hello'         #store 'hello' under the key 17
> d['a']='apple'        #store 'apple' under the key 'a'
> print d[17]+d['a']    # retrieve the stored values
'helloapple'
```

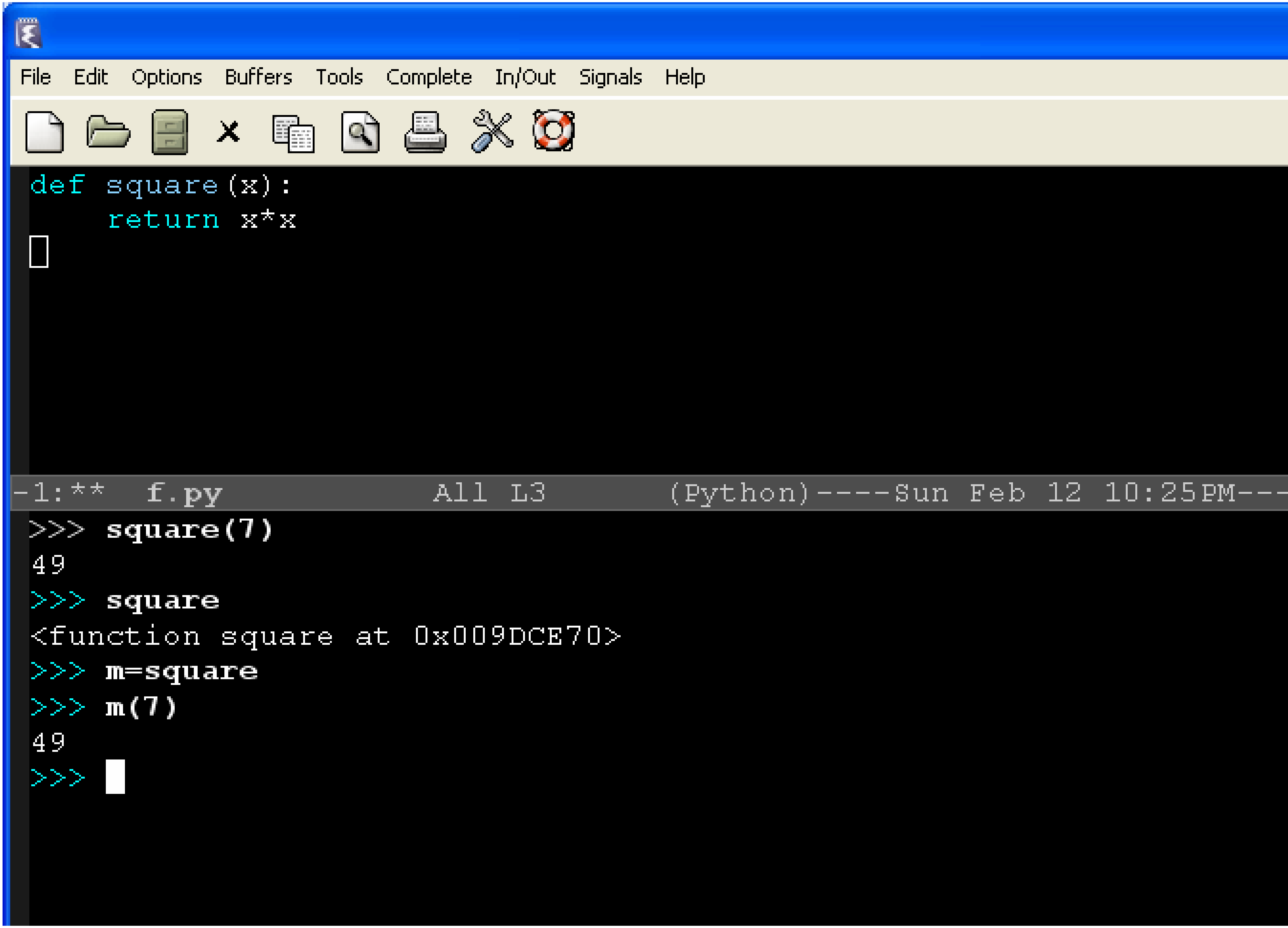
Framework for abstraction

	Procedures	Data
Primitives	+, *, ==, …	numbers, strings
Means of combination	if, while, … composition, e.g., can write 3*(4+7)	lists, dictionaries
Means of abstraction	def	
Means of capturing common patterns	?????	


File Edit Options Buffers Tools Complete In/Out Signals Help
def square(x):
    return x*x
-1:** f.py All L3 (Python)----Sun Feb 12 10:21PM
>>> square(7)
49
>>>


File Edit Options Buffers Tools Complete In/Out Signals Help
def square(x):
    return x*x
-1:** f.py All L3 (Python)----Sun Feb 12 10:22PM---
>>> square(7)
49
>>> square


File Edit Options Buffers Tools Complete In/Out Signals Help
def square(x):
    return x*x
-1:** f.py All L3 (Python)----Sun Feb 12 10:24PM---
>>> square(7)
49
>>> square
<function square at 0x009DCE70>
>>>

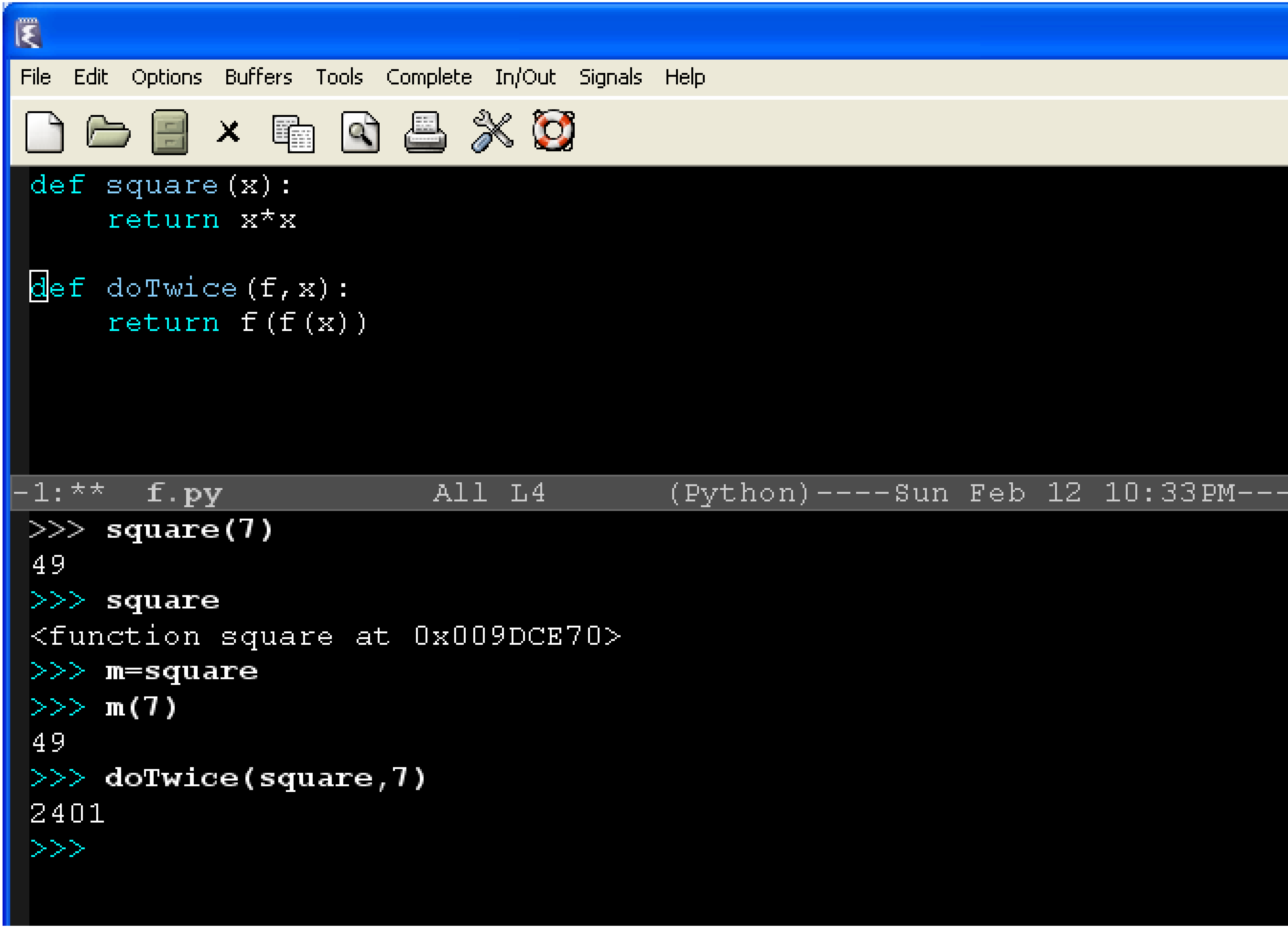

File Edit Options Buffers Tools Complete In/Out Signals Help
def square(x):
    return x*x
-1:** f.py All L3 (Python)----Sun Feb 12 10:25PM---
>>> square(7)
49
>>> square
<function square at 0x009DCE70>
>>> m=square
>>> m(7)
49
>>>

```
def square(x):
    return x*x

def doTwice(f,x):
    return f(f(x))
```

```
-1:**  f.py          All L4      (Python)----Sun Feb 12 10:31PM---
```

```
>>> square(7)
49
>>> square
<function square at 0x009DCE70>
>>> m=square
>>> m(7)
49
>>> doTwice(square,7)
```


File Edit Options Buffers Tools Complete In/Out Signals Help
def square(x):
    return x*x
def doTwice(f,x):
    return f(f(x))
-1:** f.py All L4 (Python)----Sun Feb 12 10:33PM---
>>> square(7)
49
>>> square
<function square at 0x009DCE70>
>>> m=square
>>> m(7)
49
>>> doTwice(square,7)
2401
>>>

```python
def sumint(low,high):
    s=0
    x=low
    while x < high:
        s = s + x
        x = x + 1
    return s

def sumsquares(low,high):
    s=0
    x=low
    while x < high:
        s = s + x**2
        x = x + 1
    return s

def piSum(low,high):
    s=0
    x=low
    while x < high:
        s = s + 1.0/x**2
        x = x + 2
    return s
```

```
>>> sumint(1,101)
5050
>>> sumsquares(1,101)
338350
>>> piSum(1,10000)
1.2336505501363413
>>>
>>> math.pi**2/8
1.2337005501361697
>>>
```

lambda creates procedures without naming them

- `lambda x: x+1`
  - the procedure that adds 1 to its argument
- `lambda x,y: x+ 2 * y`
  - the procedure that adds its first argument to twice its second argument
- Note that you do *not* use return
- lambda must be a single expression, not a block

```
def sumint(low,high):
    return sum(low,
               high,
               lambda x: x,
               lambda x: x+1)

def sumsquares(low,high):
    return sum(low,
               high,
               lambda x: x**2,
               lambda x: x+1)

def piSum(low,high):
    return sum(low,
               high,
               lambda x: 1.0/x**2,
               lambda x: x+2)
```

Framework for abstraction

	Procedures	Data
Primitives	+, *, ==, …	numbers, strings
Means of combination	if, while, … composition, e.g., can write 3*(4+7)	lists, dictionaries
Means of abstraction	def	
Means of capturing common patterns	higher-order procedures	

Computing square roots

- To compute an approximation to the square root of x:
  - Let g be a guess for the answer
  - Compute an improved guess by taking the average of g and x/g
  - Keep improving the guess until it's good enough. Where good enough means that g-squared is close to x.

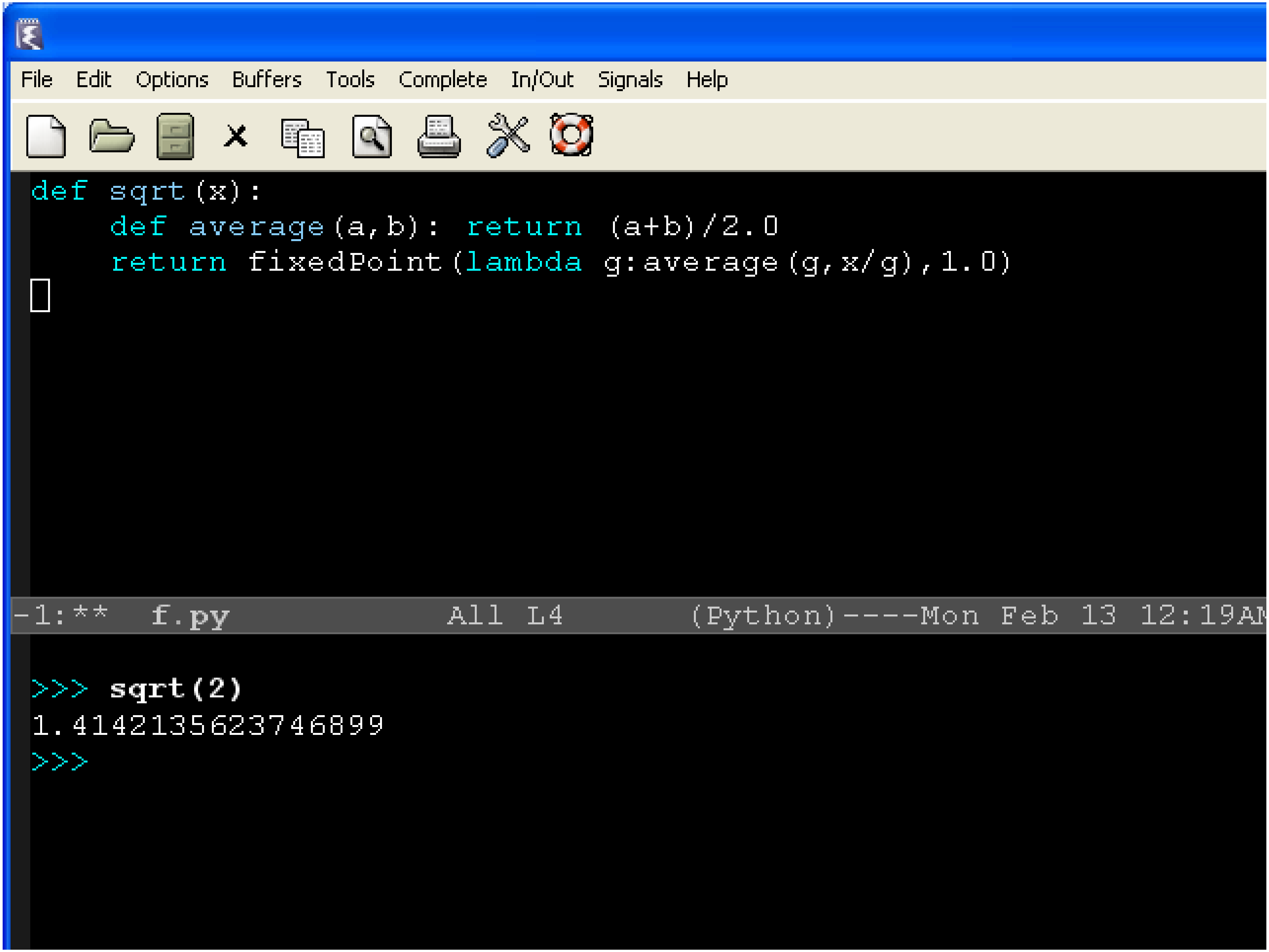
Computing square roots

- To compute an approximation to the square root of x:
  - Let g be a guess for the answer
  - Compute an improved guess by taking the average of g and x/g
  - Keep improving the guess until it's good enough. Where good enough means that g is close to x/g

Computing fixed points

- To compute fixed point of a function f
  - Start with a guess
  - Keep applying f over and over until the result doesn't change very much

```
def fixedPoint(f,firstGuess):
    def close(g1,g2):
        return abs(g1-g2)<.0001
    def iter(guess,next):
        while True:
            if close(guess, next):
                return next
            else:
                guess=next
                next=f(next)
    return iter(firstGuess,f(firstGuess))
```

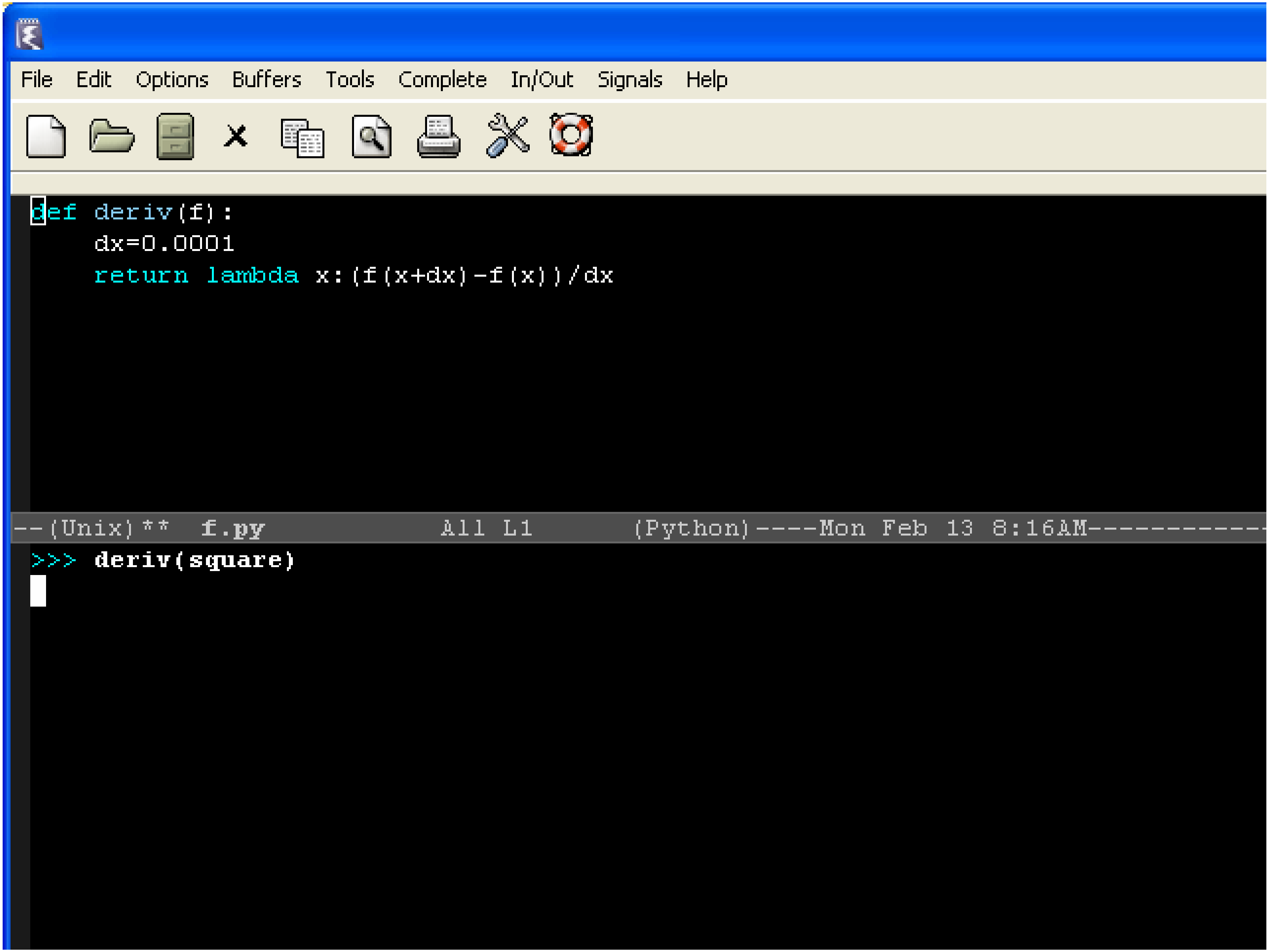

File Edit Options Buffers Tools Complete In/Out Signals Help
def sqrt(x):
def average(a,b): return (a+b)/2.0
return fixedPoint(lambda g:average(g,x/g),1.0)
-1:** f.py All L4 (Python)----Mon Feb 13 12:19AM
>>> sqrt(2)
1.4142135623746899
>>>

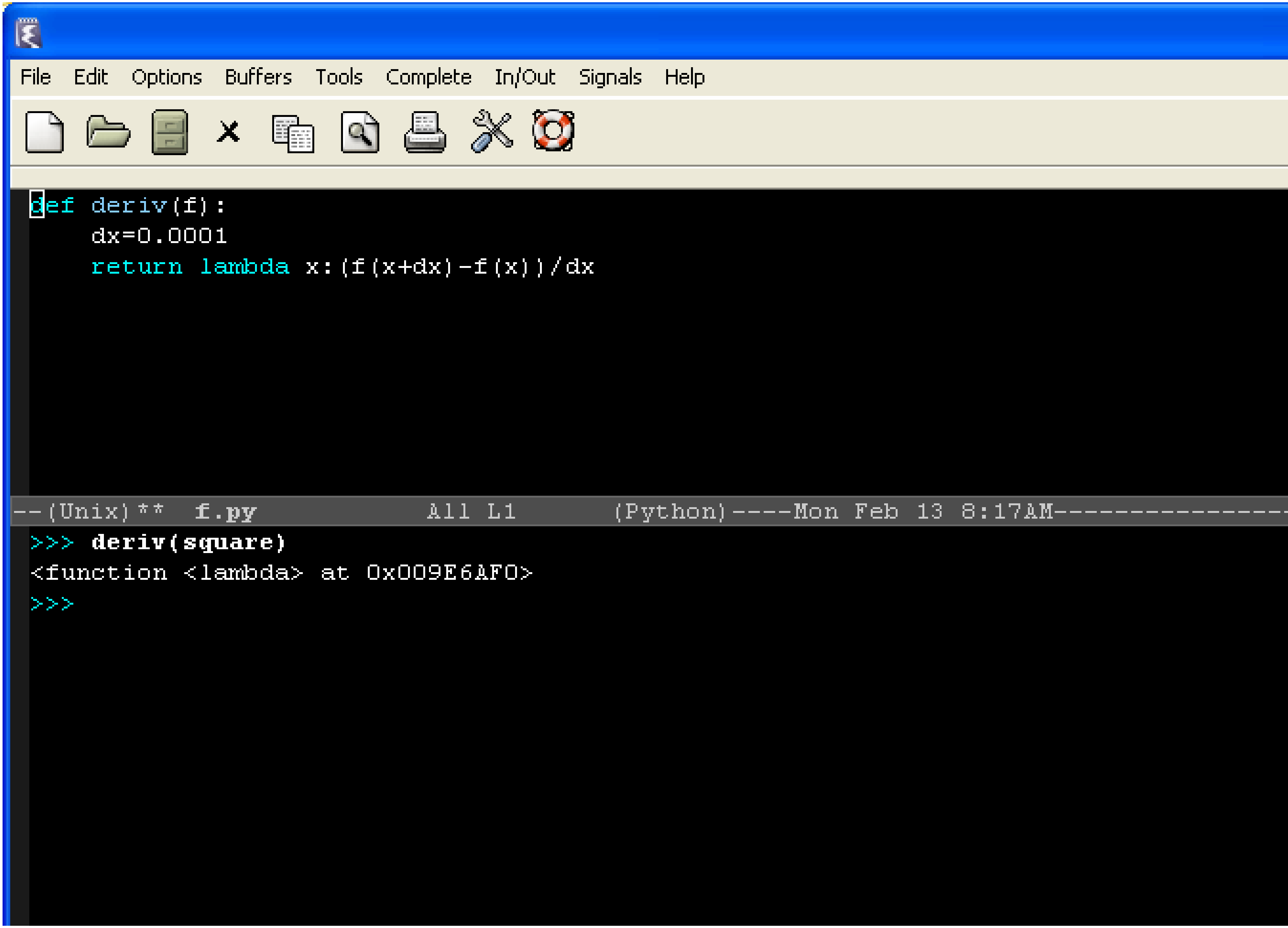
Solving f(y)=0 by Newton's Method

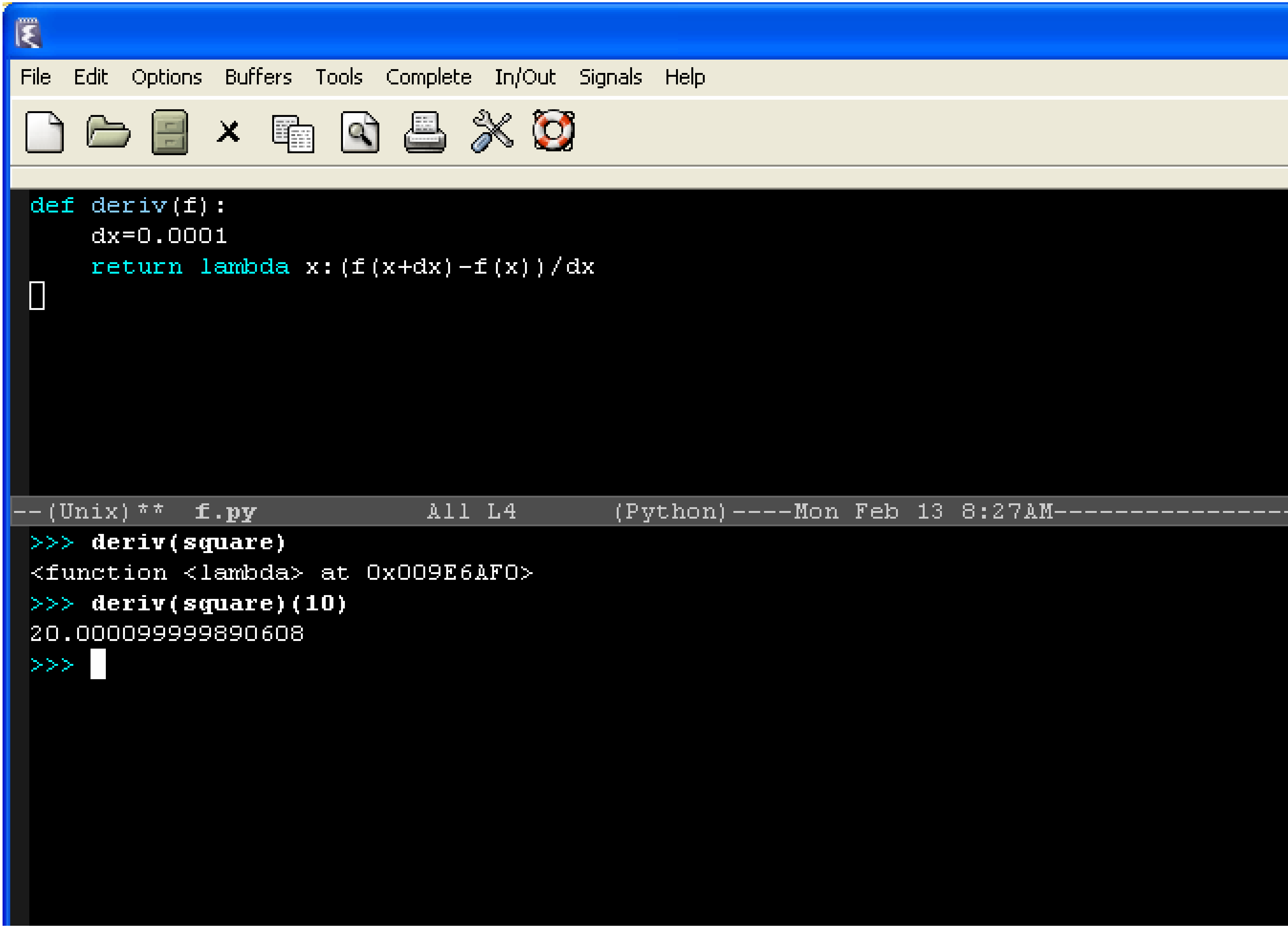
- To compute a solution of f(y)=0
  - Let g be a guess for the answer
  - Compute an improved guess as

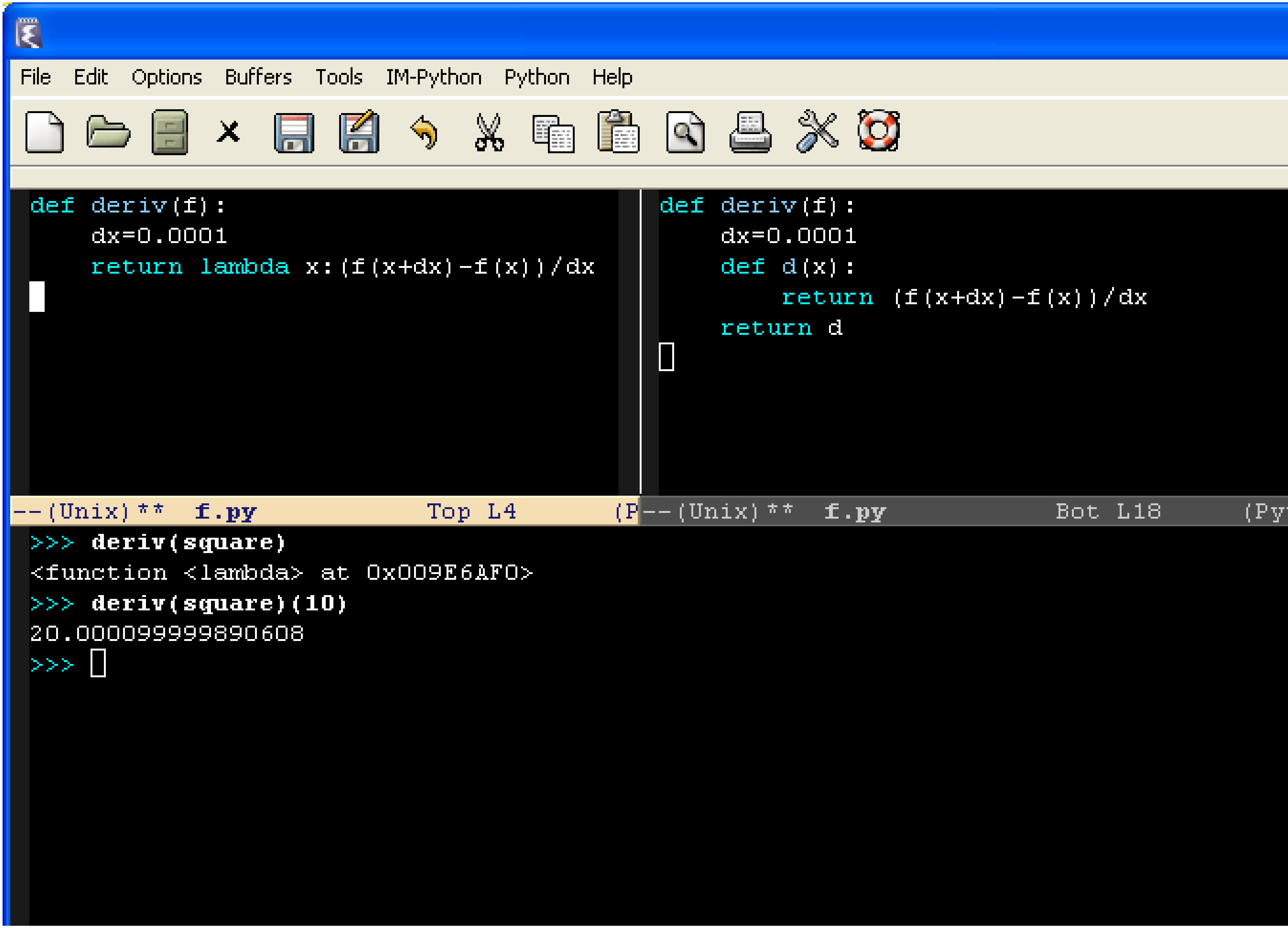
    $$g - f(g)/Df(g)$$

    where Df is the derivative of f
  - Keep improving the guess until it's good enough.


File Edit Options Buffers Tools Complete In/Out Signals Help
def deriv(f):
dx=0.0001
return lambda x:(f(x+dx)-f(x))/dx
--(Unix)** f.py All L1 (Python)----Mon Feb 13 8:16AM----------
>>> deriv(square)


File Edit Options Buffers Tools Complete In/Out Signals Help
def deriv(f):
dx=0.0001
return lambda x:(f(x+dx)-f(x))/dx
--(Unix)** f.py All L1 (Python)----Mon Feb 13 8:17AM
>>> deriv(square)
<function <lambda> at 0x009E6AF0>
>>>


File Edit Options Buffers Tools Complete In/Out Signals Help
def deriv(f):
    dx=0.0001
    return lambda x:(f(x+dx)-f(x))/dx
--(Unix)** f.py All L4 (Python)----Mon Feb 13 8:27AM----------------
>>> deriv(square)
<function <lambda> at 0x009E6AF0>
>>> deriv(square)(10)
20.000099999890608
>>>


File Edit Options Buffers Tools IM-Python Python Help
def deriv(f):
dx=0.0001
return lambda x:(f(x+dx)-f(x))/dx
def deriv(f):
dx=0.0001
def d(x):
return (f(x+dx)-f(x))/dx
return d
--(Unix)** f.py Top L4 (P
--(Unix)** f.py Bot L18 (Py
>>> deriv(square)
<function <lambda> at 0x009E6AF0>
>>> deriv(square)(10)
20.000099999890608
>>>

Newton’s method as a fixed point, and computing square roots by Newton’s Method

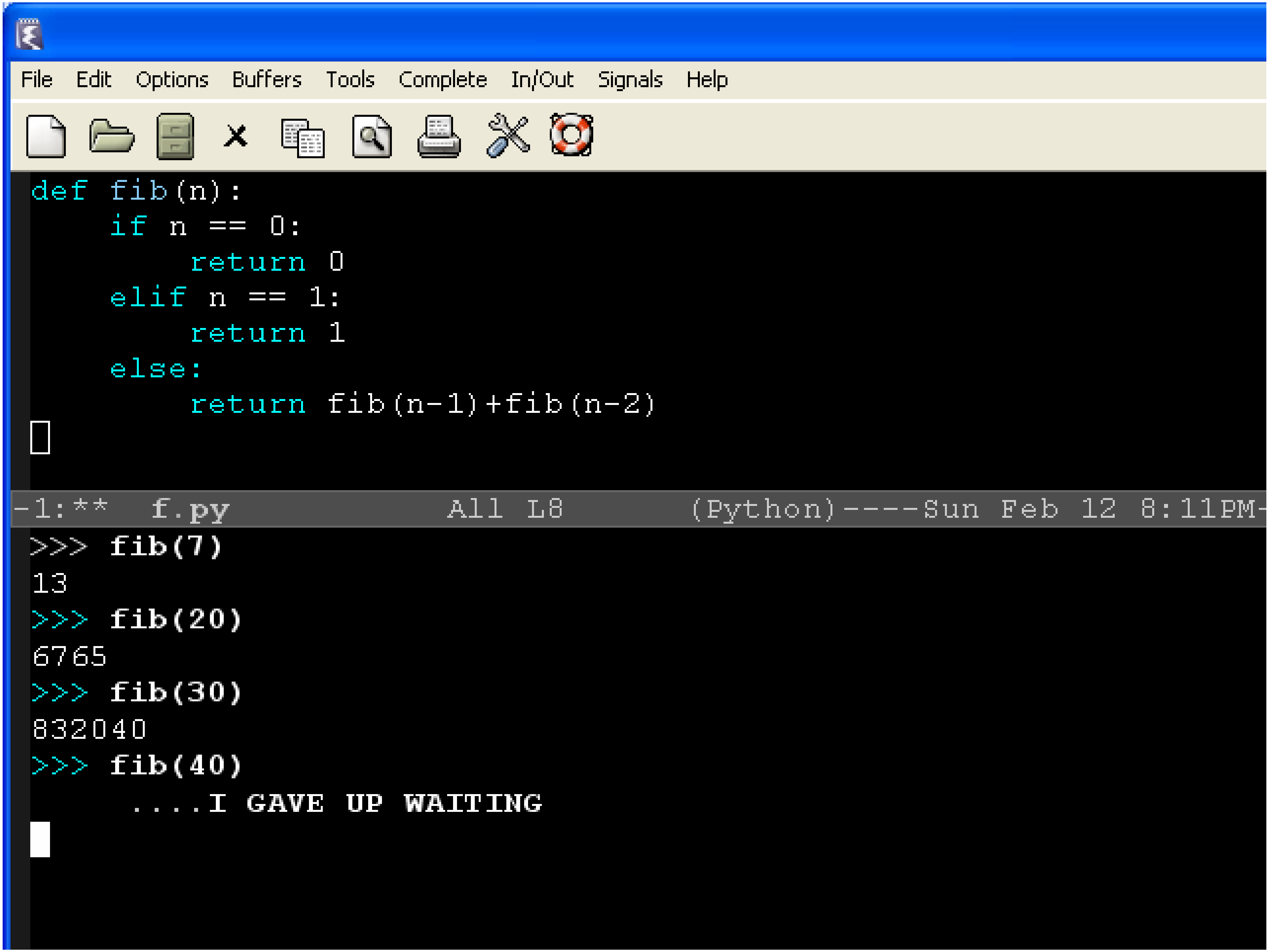
```
def newtonsMethod(f,firstGuess):
    return fixedPoint(
        lambda x: x - f(x)/deriv(f)(x),
        firstGuess)


def sqrt(x):
    return newtonsMethod(
        lambda y: y**2 - x,
        1.0)
```

Rights and privileges of first-class citizens

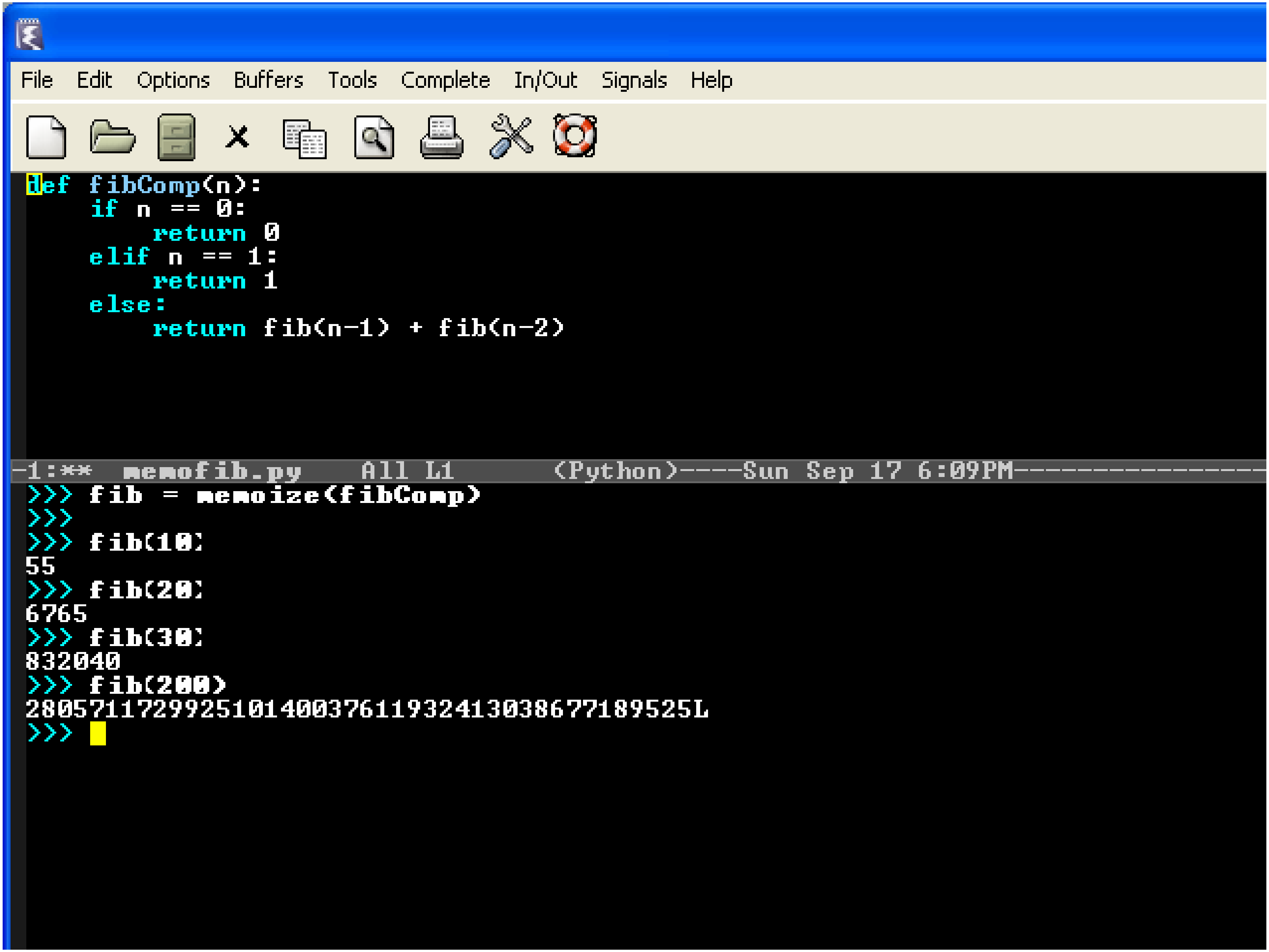
- May be named by variables
- May be passed as arguments to procedures
- May be returned as results of procedures
- May be included in data structures

-- Christopher Strachey (1916-1975)


File Edit Options Buffers Tools Complete In/Out Signals Help
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1)+fib(n-2)
-1:** f.py All L8 (Python)----Sun Feb 12 8:11PM-
>>> fib(7)
13
>>> fib(20)
6765
>>> fib(30)
832040
>>> fib(40)
....I GAVE UP WAITING

Memoization

```
def memoize(f):
    storedResults={}
    def doit(n):
        if storedResults.has_key(n):
            return storedResults[n]
        else:
            value = f(n)
            storedResults[n] = value
            return value
    return doit
```



File Edit Options Buffers Tools Complete In/Out Signals Help
def fibComp(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
-1:** memofib.py All L1 (Python)----Sun Sep 17 6:09PM----------------
>>> fib = memoize(fibComp)
>>>
>>> fib(10)
55
>>> fib(20)
6765
>>> fib(30)
832040
>>> fib(200)
280571172992510140037611932413038677189525L
>>>

END