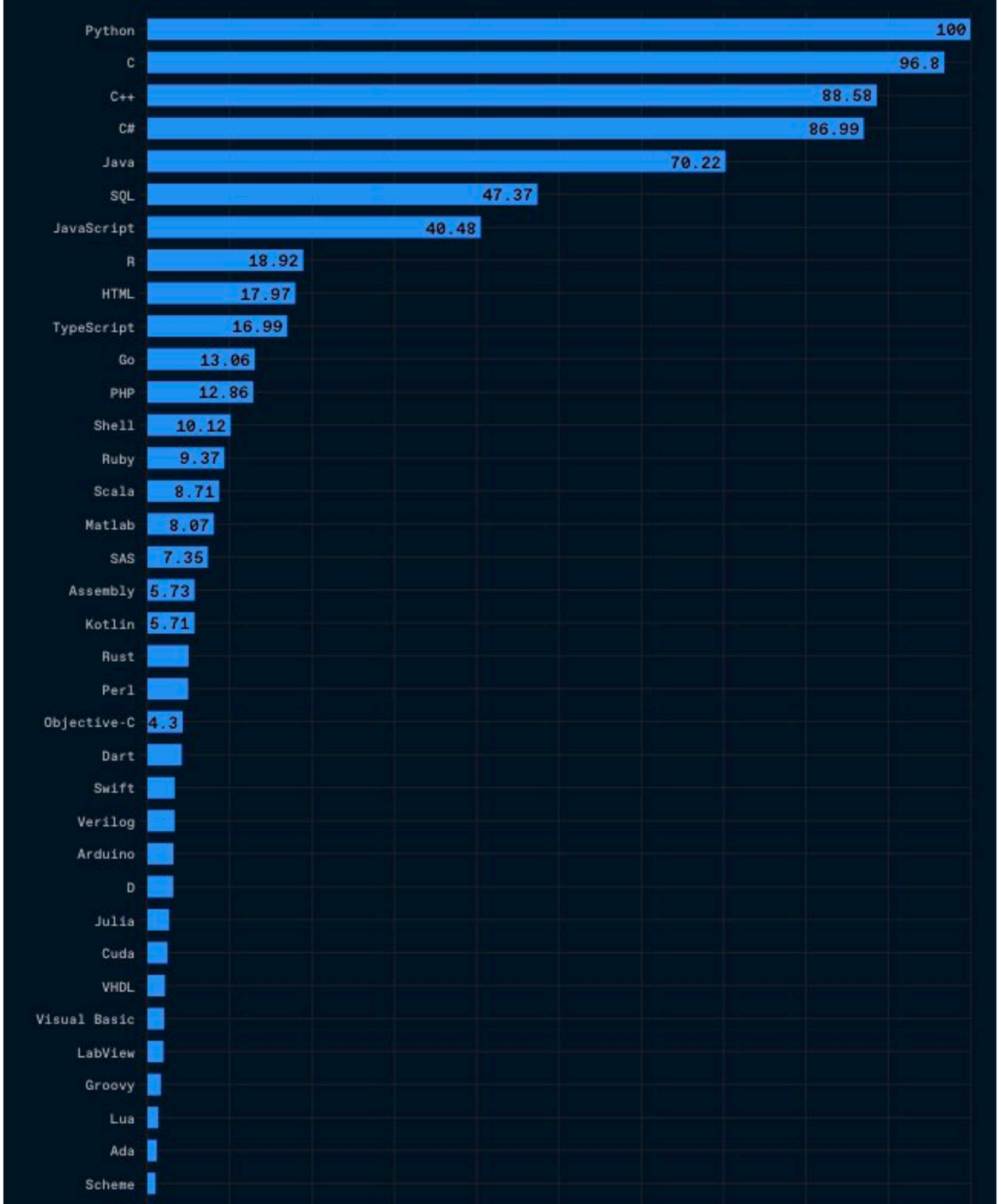
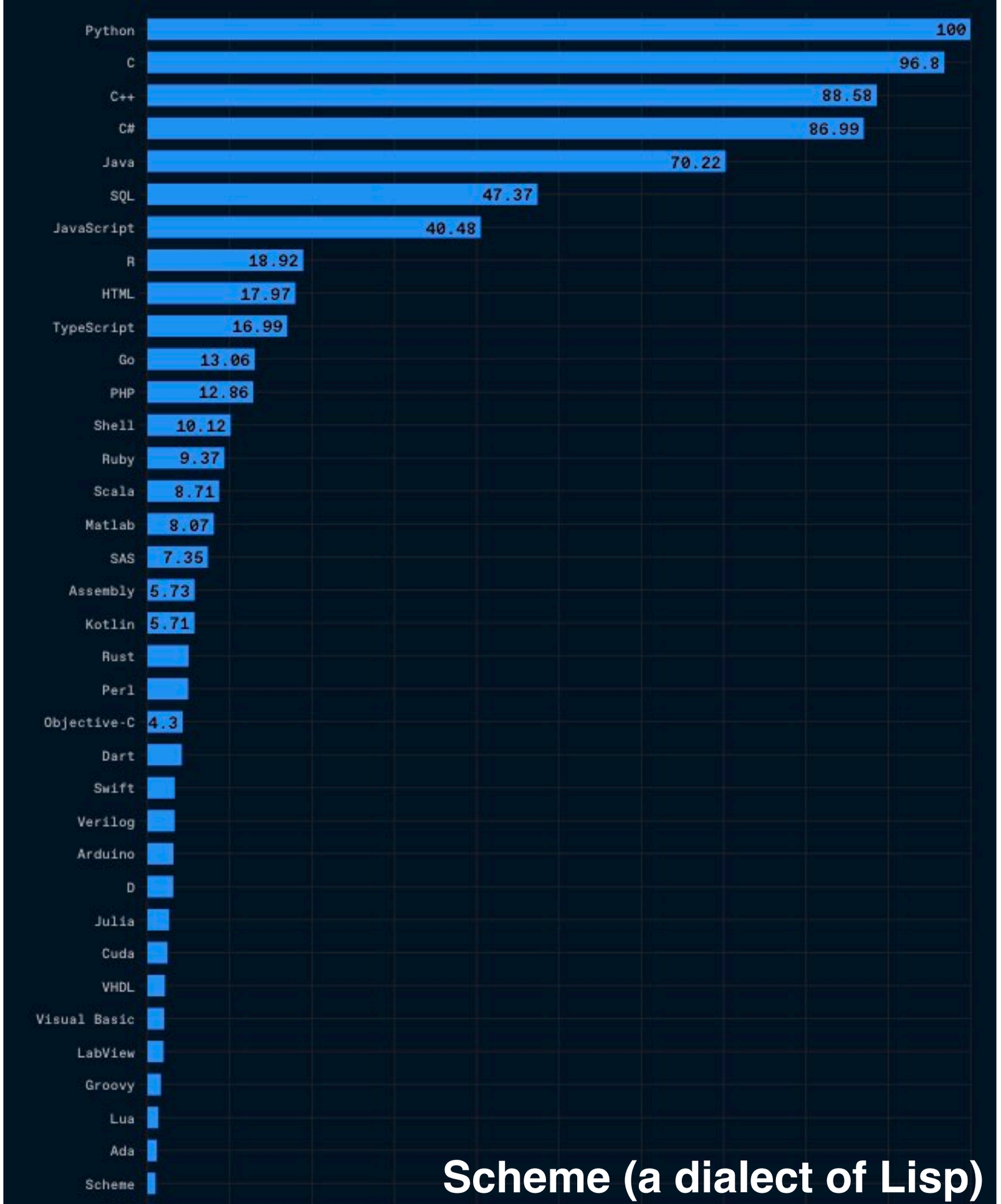
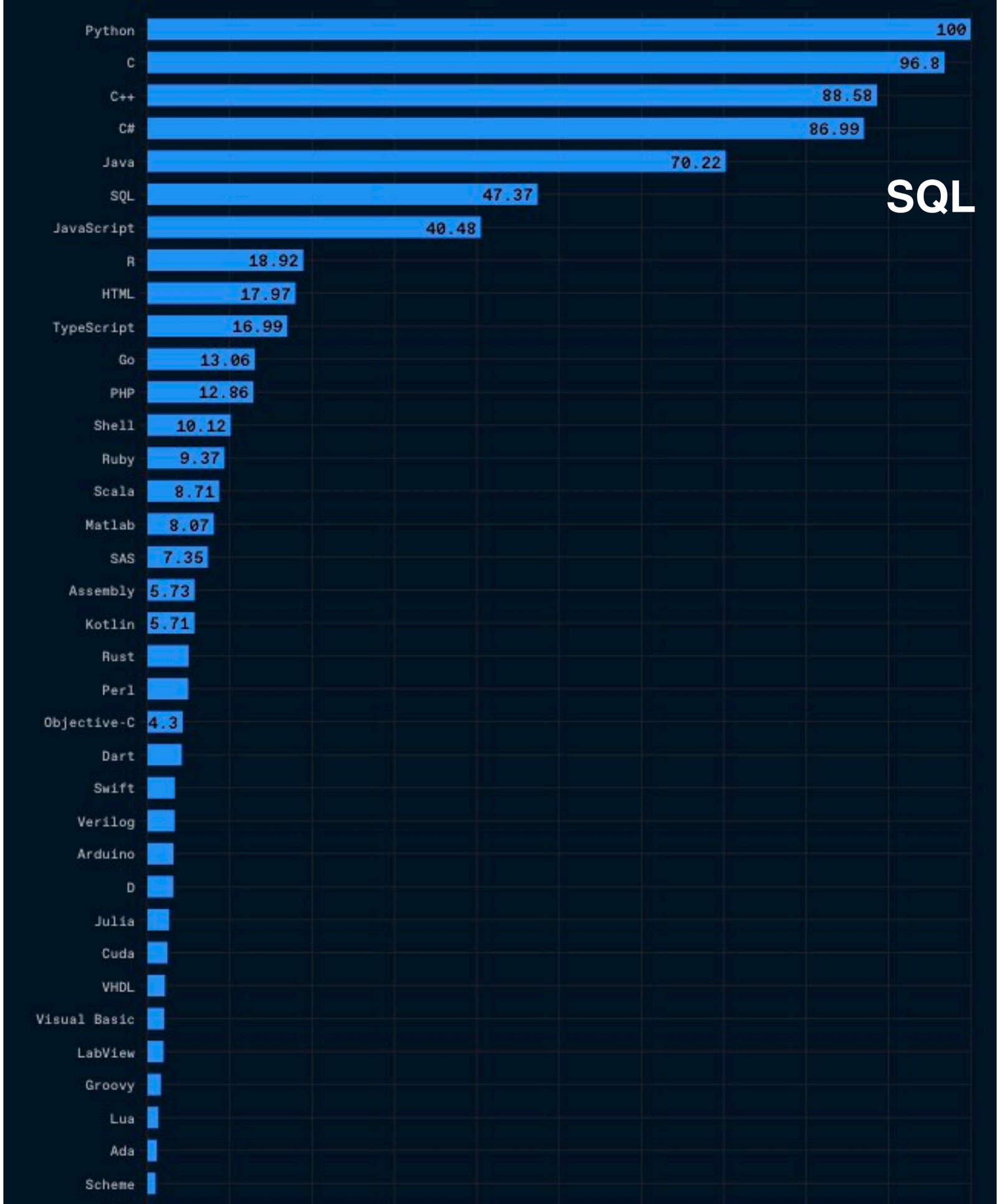
Programming Languages

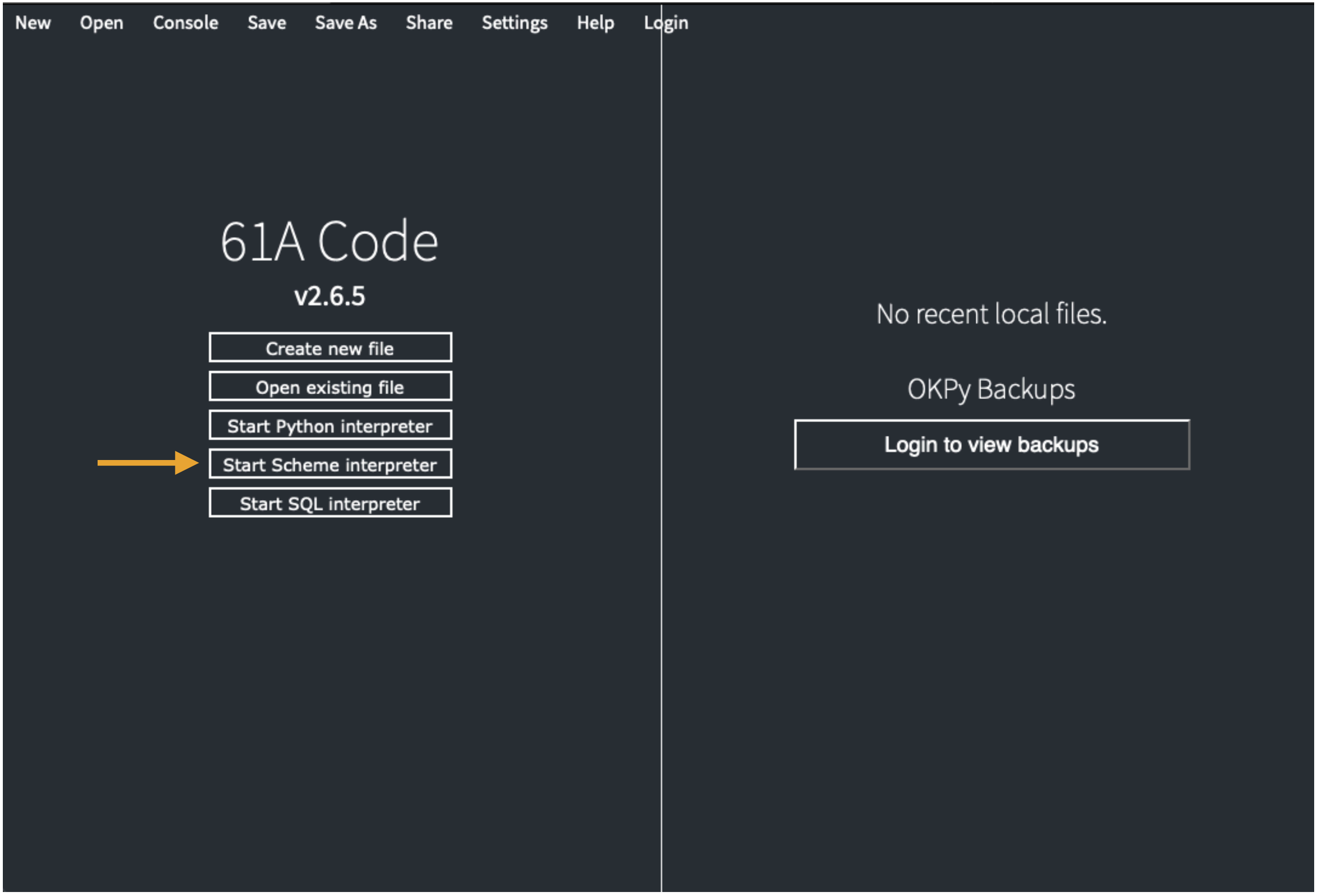


Programming Languages



Programming Languages




New
Open
Console
Save
Save As
Share
Settings
Help
Login
61A Code
v2.6.5
Create new file
Open existing file
Start Python interpreter
Start Scheme interpreter
Start SQL interpreter
No recent local files.
OKPy Backups
Login to view backups

Scheme Expressions

Python	Scheme
`1 + 2` `3 + 4 + 5 + 6`	`(+ 1 2)` `(+ 3 4 5 6)`

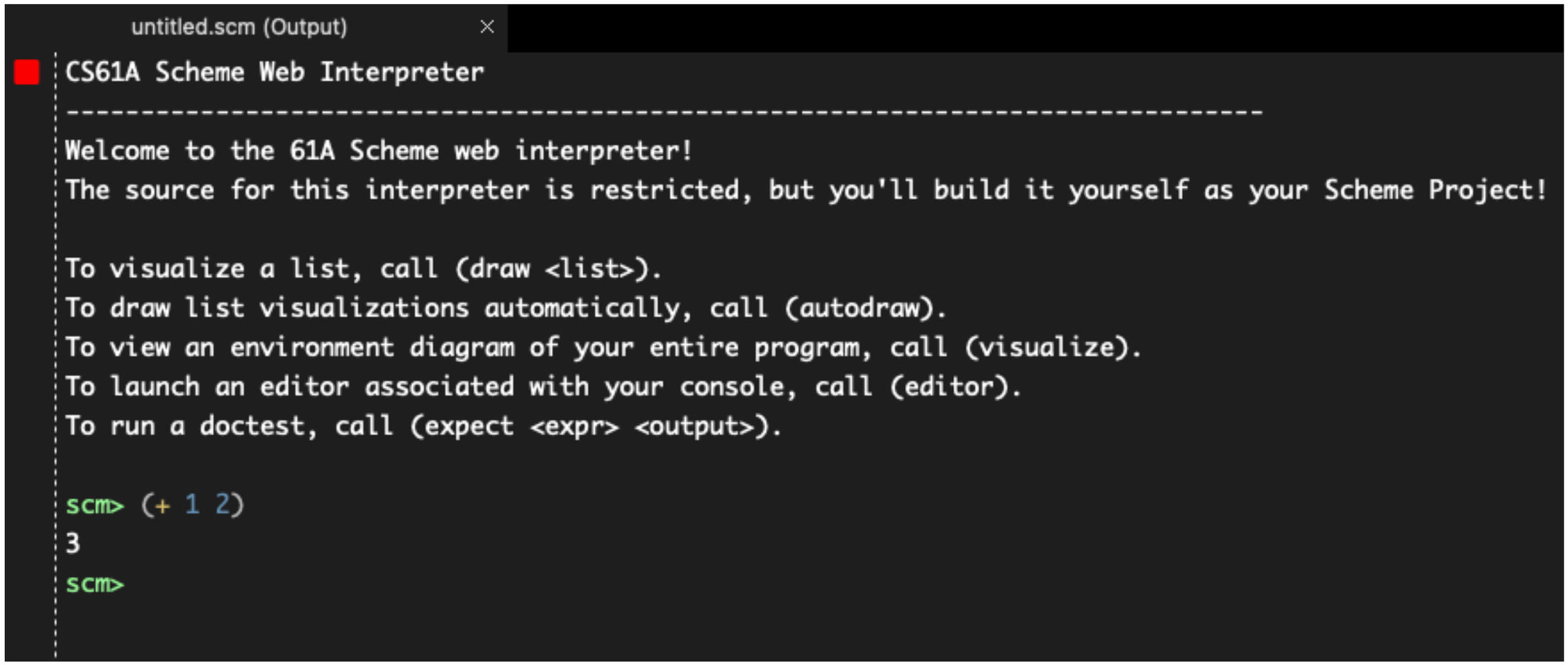
Scheme Expressions

Python	Scheme
`1 + 2` `3 + 4 + 5 + 6`	`(+ 1 2)` `(+ 3 4 5 6)`
`(2 * 4) + (3 + 5)`	`(+ (* 2 4) (+ 3 5))`

Scheme Expressions

Python	Scheme
`1 + 2` `3 + 4 + 5 + 6`	`(+ 1 2)` `(+ 3 4 5 6)`
`(2 * 4) + (3 + 5)`	`(+ (* 2 4) (+ 3 5))`
`"string"` `'string'`	`"string"`

Scheme Expressions



Scheme Special Forms

Python	Scheme
`x = 3`	`(define x 3.14)`

Scheme Special Forms

Python	Scheme
`x = 3`	`(define x 3.14)`
`if x < 0:` `    print(-x)` `else:` `    print(x)`	`(if (< x 0) (- x) x)`

Scheme Special Forms

Python	Scheme
`x = 3`	`(define x 3.14)`
`if x < 0:` `  print(-x)` `else:` `  print(x)`	`(if (< x 0) (- x) x)` `(if (< x 0) (print (- x)) (print x))`

Scheme Special Forms

Python	Scheme
`x = 3`	`(define x 3.14)`
`if x < 0:` `  print(-x)` `else:` `  print(x)`	`(if (< x 0) (- x) x)` `(if (< x 0) (print (- x)) (print x))` `(print (if (< x 0) (- x) x))`

Scheme Special Forms

Python	Scheme
`x = 3`	`(define x 3.14)`
`if x < 0:` `  print(-x)` `else:` `  print(x)`	`(if (< x 0) (- x) x)` `(if (< x 0) (print (- x)) (print x))` `(print (if (< x 0) (- x) x))`
`def abs(x):` `  if x < 0:` `  return -x` `  else:` `  return x`	`(define (abs x)` `  (if (< x 0)` `  (- x)` `  x))`

Scheme Special Forms

Python	Scheme
`x = 3`	`(define x 3.14)`
`if x < 0:` `  print(-x)` `else:` `  print(x)`	`(if (< x 0) (- x) x)` `(if (< x 0) (print (- x)) (print x))` `(print (if (< x 0) (- x) x))`
`def abs(x):` `  if x < 0:` `  return -x` `  else:` `  return x`	`(define (abs x)(if (< x 0)(- x) x))` unlike python, white space doesn't matter

Scheme Special Forms

Python	Scheme
`x = -1` `if x < 0:` `    print(x)` `    print("done")`	`(define x -1)` `(if (< x 0)` `    (begin (print x)` `           (print "done")` `     ))`

Scheme Special Forms

Python	Scheme
`` if(x < 0): print(-1) elif(x > 0): print(1) else: print(0) ```	``` (cond ((< x 0) -1) ((> x 0) 1) (else 0)) ```

Scheme Special Forms

Python	Scheme
`x = 3` `(x > 1) and (x < 10)` `true`	`(define x 3)` `(and (> x 1) (< x 10))` `#t`

Scheme Special Forms

Python	Scheme
`x = 3` `(x > 1) and (x < 10)` `true`	`(define x 3)` `(and (> x 1) (< x 10))` `#t` `(or (= x 3) (< (/ x 0) 1))` `#t`

Scheme Special Forms

Python	Scheme
`x = 3` `(x > 1) and (x < 10)` `true`	`(define x 3)` `(and (> x 1) (< x 10))` `#t` `(or (= x 3) (< (/ x 0) 1))` `#t` `(define x 2)` `(or (= x 3) (< (/ x 0) 1))` `Error: division by zero`

Scheme Lambda Expressions

Python	Scheme
`f = lambda x,y,z: x+y+abs(z)` `f(1,2,3)`	`(define f (lambda (x y z)` `(+ x y (abs z))))` `(f 1 2 3)`

Scheme Let Expressions

Python	Scheme
a = 3 b = 4 c = math.sqrt(a*a + b*b) # a and b remain bound	(define c (let ((a 3)(b 4)) (sqrt (+ (* a a) (* b b))))) ; a and b are not bound here

Scheme Let Expressions

Python	Scheme
	; inline comment ;; short comment on own line ;;; longer explanatory comment

Scheme

```
(number? 3)
#t

(zero? 2)
#f

(integer? 2.2)
#f

(equal? 2 2) ; see also eq?
#t

(string? "foo")
#t
```

Python	Scheme
`for x in range(10):` `  print(x)` `x = 0` `while(x < 10):` `  print(x)` `  x = x + 1`	

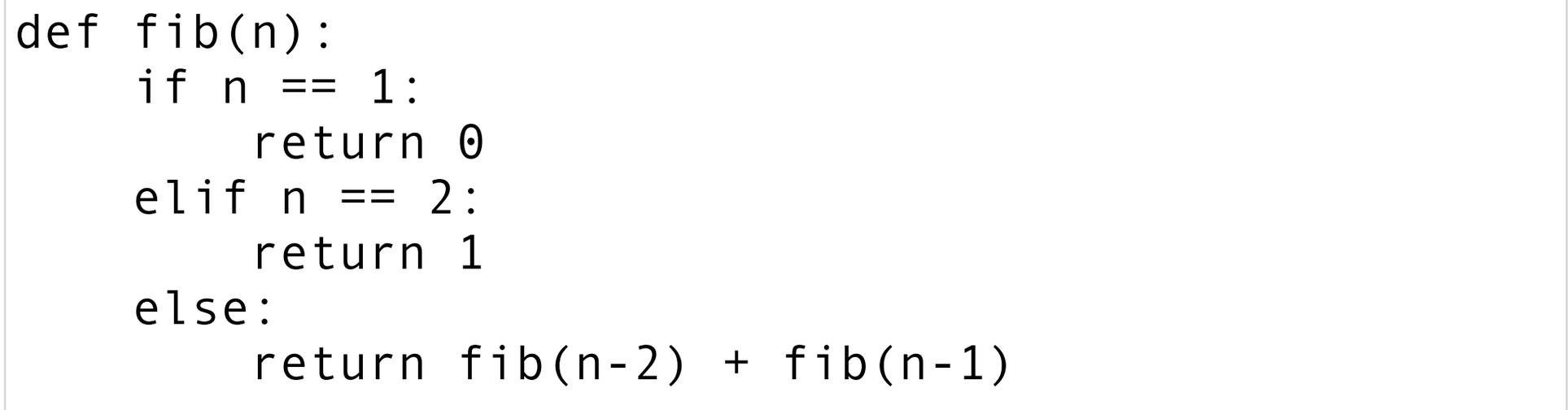
Python	`def factorial(n):` `    if n == 0:` `        return 1` `    else:` `        return n * factorial(n-1)`
Scheme	

Python	`def factorial(n):` `  if n == 0:` `  return 1` `  else:` `  return n * factorial(n-1)`
Scheme	`(define (factorial n)` `  (if (= n 0)` `  1` `  (* n (factorial (- n 1)))` `  )) `

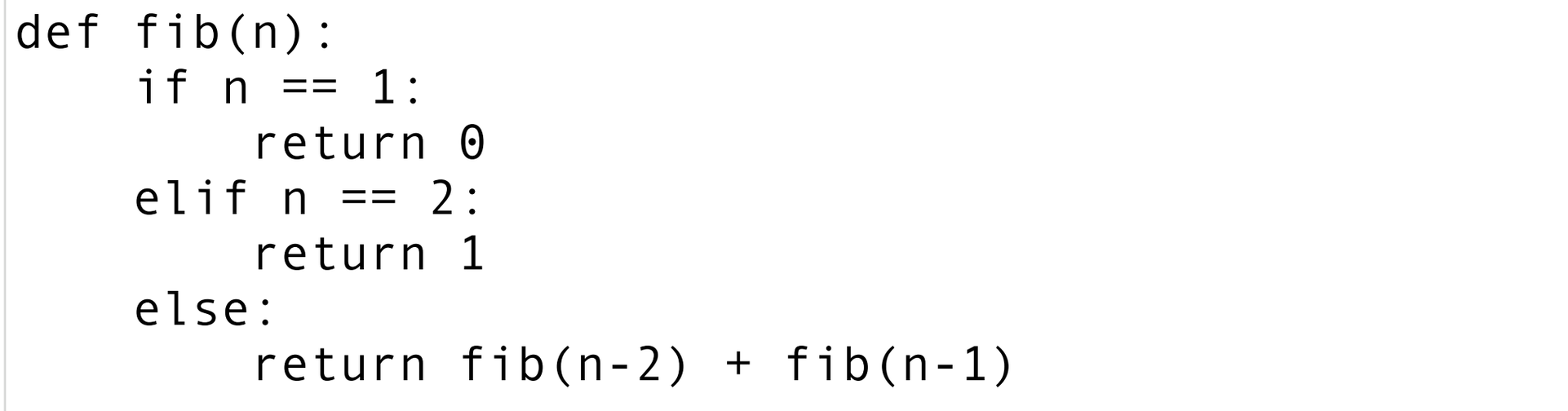
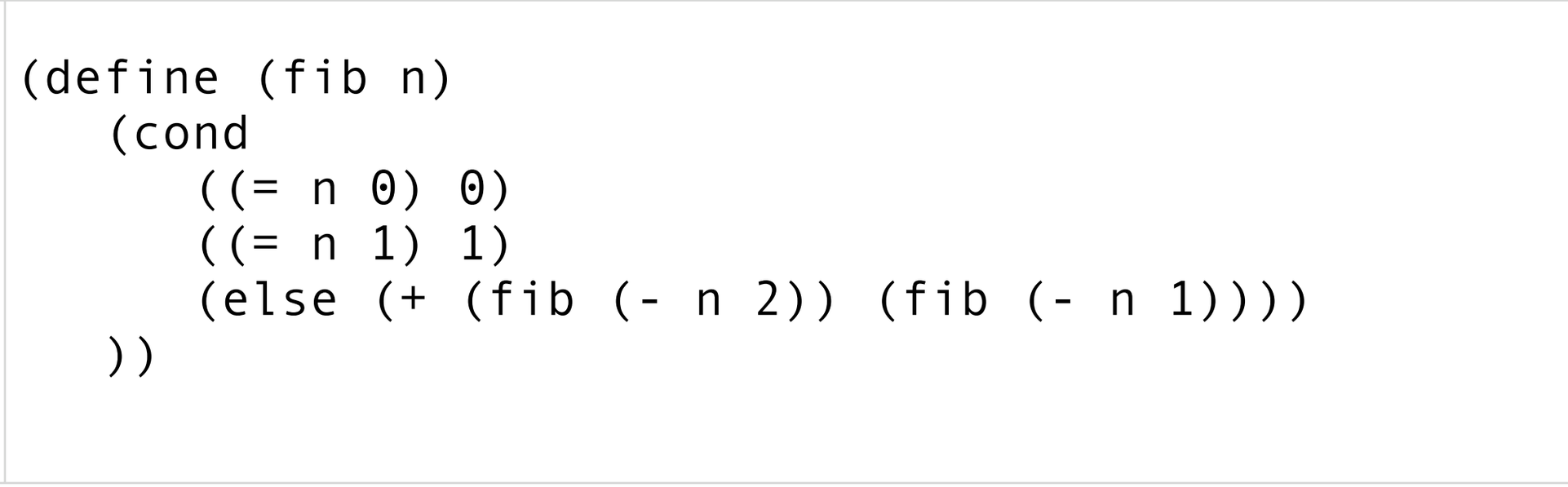
Python	`def perfectSquare(x,i=0):` `  if i > sqrt(x):` `  return False` `  else:` `  return i*i==x or perfectSquare(x,i+1)`
Scheme	

Python	`def perfectSquare(x,i=0):` `  if i > sqrt(x):` `  return False` `  else:` `  return i*i==x or perfectSquare(x,i+1)`
Scheme	`(define (perfectSquare n i)` `  (if (> i (sqrt n))` `  #f` `  (or (= (* i i) n) (perfectSquare n (+ i 1)))` `  ))` `(perfectSquare 16 0)`

Python	`def perfectSquare(x,i=0):` `  if i > sqrt(x):` `  return False` `  else:` `  return i*i==x or perfectSquare(x,i+1)`
Scheme	`(define (perfectSquare n i)` `  (if (> i (sqrt n))` `  #f` `  (or (= (* i i) n) (perfectSquare n (+ i 1)))` `  ))` `(perfectSquare 16 0)`
Scheme	`(define (perfectSquare n)` `  (define (psHelper n i)` `  (if (> i (sqrt n))` `  #f` `  (or (= (* i i) n) (psHelper n (+ i 1)))` `  ))` `  (psHelper n 0))` `(perfectSquare 16)`

Python	
Scheme	

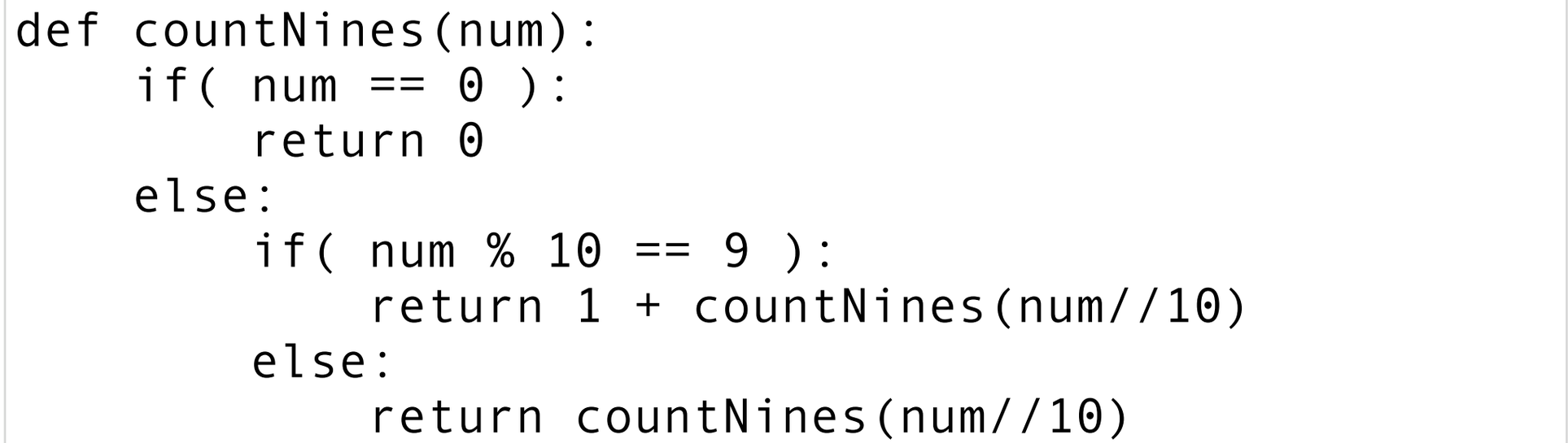
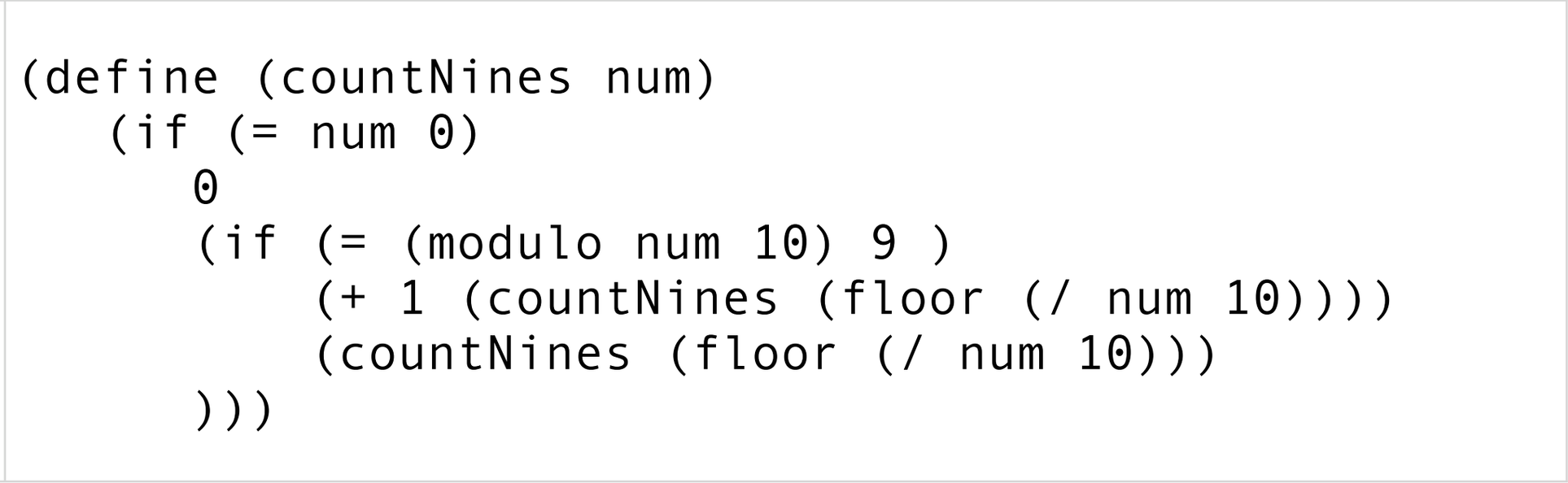
```
def fib(n):
    if n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fib(n-2) + fib(n-1)
```

Python	
Scheme	

```
def fib(n):
    if n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fib(n-2) + fib(n-1)
```

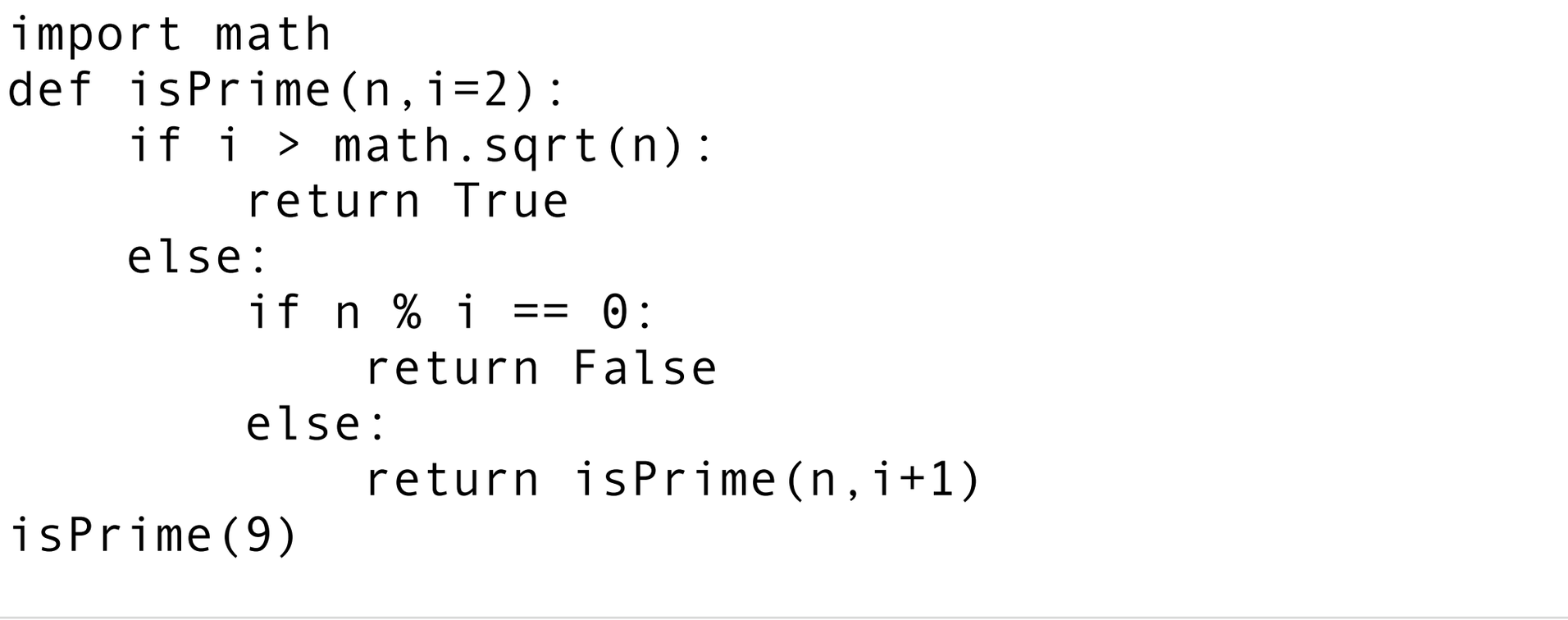
```
(define (fib n)
   (cond
      ((= n 0) 0)
      ((= n 1) 1)
      (else (+ (fib (- n 2)) (fib (- n 1))))
   ))
```

Python	`def countNines(num):` `    if( num == 0 ):` `        return 0` `    else:` `        if( num % 10 == 9 ):` `            return 1 + countNines(num//10)` `        else:` `            return countNines(num//10)`
Scheme	

Python	
Scheme	

```
def countNines(num):
    if( num == 0 ):
        return 0
    else:
        if( num % 10 == 9 ):
            return 1 + countNines(num//10)
        else:
            return countNines(num//10)
```

```
(define (countNines num)
   (if (= num 0)
       0
       (if (= (modulo num 10) 9 )
           (+ 1 (countNines (floor (/ num 10))))
           (countNines (floor (/ num 10)))
       )))
```

Python	
Scheme	

Python

```
import math
def isPrime(n,i=2):
    if i > math.sqrt(n):
        return True
    else:
        if n % i == 0:
            return False
        else:
            return isPrime(n,i+1)
isPrime(9)
```

Scheme

```
(define (isPrime n i)
   (if (> i (sqrt n))
       #t
       (if (= (modulo n 2) 0)
          #f
          (isPrime n (+ i 1)))
    ))
(isPrime 9 2)
```

Python

```
import math
def isPrime(n,i=2):
    if i > math.sqrt(n):
        return True
    else:
        if n % i == 0:
            return False
        else:
            return isPrime(n,i+1)
isPrime(9)
```

Scheme

```
(define (isPrime n i)
   (cond ((> i (sqrt n)) #t)
         ((= (modulo n 2) 0) #f)
         (else (isPrime n (+ i 1)))
    ))
(isPrime 9 2)
```

```
(define (single-digit x)
    ; return true if x>=0 and x<10)

(single-digit 5)
#t

(single-digit 12)
#f
```

```
(define (single-digit x)
   (and (>= x 0) (< x 10)))

(single-digit 5)
#t

(single-digit 12)
#f
```

```
(define (single-digit x)
   (and (>= x 0) (< x 10)))

(single-digit 5)
#t

(single-digit 12)
#f


(if (= 5 (+ 2 3)) 10 20)
```

```
(define (single-digit x)
   (and (>= x 0) (< x 10)))

(single-digit 5)
#t

(single-digit 12)
#f



(if (= 5 (+ 2 3)) 10 20)
10
```

```
(define (single-digit x)
   (and (>= x 0) (< x 10)))

(single-digit 5)
#t

(single-digit 12)
#f



(if (= 5 (+ 2 3)) 10 20)
10

(define (my-max x y)
   ; your code)
(my-max 20 10)
20
```

```
(define (single-digit x)
   (and (>= x 0) (< x 10)))

(single-digit 5)
#t

(single-digit 12)
#f



(if (= 5 (+ 2 3)) 10 20)
10

(define (my-max x y)
   (if (> x y) x y))
(my-max 20 10)
20
```

```
(define (single-digit x)
   (and (>= x 0) (< x 10)))

(single-digit 5)
#t

(single-digit 12)
#f



(if (= 5 (+ 2 3)) 10 20)
10

(define (my-max x y)
   (if (> x y) x y))
(my-max 20 10)
20



(define (my-max3 x y z)
   (if (and ???)
       x
       (if ???
           y
           z)))
(my-max3 4 12 8)
12
```

```
(define (single-digit x)
   (and (>= x 0) (< x 10)))

(single-digit 5)
#t

(single-digit 12)
#f



(if (= 5 (+ 2 3)) 10 20)
10

(define (my-max x y)
   (if (> x y) x y))
(my-max 20 10)
20



(define (my-max3 x y z)
   (if (and (> x y) (> x z))
       x
       (if (> y z)
           y
           z)))
(my-max3 4 12 8)
12
```

```
(define (single-digit x)
   (and (>= x 0) (< x 10)))

(single-digit 5)
#t

(single-digit 12)
#f



(if (= 5 (+ 2 3)) 10 20)
10

(define (my-max x y)
   (if (> x y) x y))
(my-max 20 10)
20



(define (my-max3 x y z)
   (if (and (> x y) (> x z))
       x
       (if (> y z)
           y
           z)))
(my-max3 4 12 8)
12
```

```
(define (my-max3 x y z)
   (cond ((and (> x y) (> x z)) x)
         ((> y z) y)
         (else z)))
```