NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 2, Lecture 6

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Some examples

- Find all factors of a number n
- Factors must lie between 1 and n

```
def factors(n):
  factorlist = []
  for i in range(1,n+1):
    if n%i == 0:
      factorlist = factorlist + [i]
  return(factorlist)
```

Primes

- Prime number — only factors are 1 and itself
- `factors(17)` is `[1,17]`
- `factors(18)` is `[1,2,3,6,9,18]`

```
def isprime(n):
  return(factors(n) == [1,n])
```

- 1 should not be reported as a prime
  - `factors(1)` is `[1]`, not `[1,1]`

Primes upto n

- List all primes below a given number

```
def primesupto(n):
  primelist = []
  for i in range(1,n+1):
    if isprime(i):
      primelist = primelist + [i]
  return(primelist)
```

First n primes

- List the first n primes

```
def nprimes(n):
  (count,i,plist) = (0,1,[])
  while(count < n):
    if isprime(i):
      (count,plist) = (count+1,plist+[i])
    i = i+1
  return(plist)
```

`for` and `while`

- `primesupto()`
  - Know we have to scan from `1` to `n`, use `for`
- `nprimes()`
  - Range to scan not known in advance, use `while`

for and while

- Can use while to simulate for

```
for n in range(i,j):
  statement
```

```
n = i
while n < j:
  statement
  n = n+1
```

```
for n in l:
  statement
```

```
i = 0
while i < len(l):
  n = l[i]
  statement
  i = i+1
```

`for` and `while`

- Can use `while` to simulate `for`
- However, use `for` where it is natural
  - Makes for more readable code
- What makes a good program?
  - Correctness and efficiency — algorithm
  - Readability, ease of maintenance — style
  - What you say, and how you say it