



## UNIT 2B

# An Introduction to Programming

15110 Principles of Computing,  
Carnegie Mellon University - CORTINA

1

## for Loop

```
for loop_variable in start .. end do  
    loop body  
end
```

*start* and *end* are integers.

For the first iteration, *loop\_variable* is set to *start*.

For each subsequent iteration, the *loop\_variable* increases by 1.

In the last iteration, *loop\_variable* is equal to *end*.

15110 Principles of Computing,  
Carnegie Mellon University - CORTINA

2

## for Loop

```
for loop_variable in start .. end do
  loop body
end
```

The loop body is one or more instructions that you want to repeat. (We usually indent the body for readability.)

If  $\text{start} \leq \text{end}$ , the for loop repeats the loop body  $\text{end} - \text{start} + 1$  times.

If  $\text{start} > \text{end}$ , the entire loop is skipped.

## for Loop Example

```
for i in 1..5 do
  print "hello world\n"
end
```

```
hello world
hello world
hello world
hello world
hello world
```

## for Loop Example

```
for i in 1..5 do
  print i
  print "\n"
end
```

```
1
2
3
4
5
```

## for Loop Example

```
for i in 1..5 do
  print i
end
```

```
12345
```

```
for i in 1..5 do
  print i
  print " "
end
```

```
1 2 3 4 5
```

## for Loop Example

```
for i in 1..10 do
  print i*2
  print " "
end
```

```
2 4 6 8 10 12 14 16 18 20
```

## Danger!

```
for i in 1..5 do
  print i
  print " "
  i = 10
end
```

```
1 2 3 4 5
```

```
for i in 1..5 do
  i = 10
  print i
  print " "
end
```

```
10 10 10 10 10
```

If you modify the loop variable inside of the `for` loop, the loop will reset the loop variable to its next expected value in the next iteration.

Programming suggestion:  
Do NOT modify the loop variable inside a `for` loop.



## Assignment Statements

An assignment statement has two parts:

*variable* = *expression*

The *expression* on the right side of the equals is evaluated and the result is stored in the *variable* shown on the left side of the equals (overwriting the previous contents of that *variable*).

	x	y
x = 150	150	?
y = x * 10	150	1500
y = y + 1	150	1501
x = x + y	1651	1501

15110 Principles of Computing,  
Carnegie Mellon University - CORTINA

9

## A function using a for loop

```
def sum()  
  # sums the first 5 positive integers  
  sum = 0  
  for i in 1..5 do  
    sum = sum + i  
  end  
  return sum  
end  
  
sum()  
=> 15
```

i	sum
?	0
1	1
2	3
3	6
4	10
5	15

15110 Principles of Computing,  
Carnegie Mellon University - CORTINA

10

## Generalizing our solution

```
def sum(n)
  # sums the first n positive integers
  sum = 0
  for i in 1..n do
    sum = sum + i
  end
  return sum
end
```

```
sum(6)           => 21
sum(100)         => 5050
sum(15110)       => 114163605
```

## An epidemic

```
def compute_sick(n)
  # computes total sick after n days
  newly_sick = 1
  total_sick = 1
  for day in 2..n do
    # each iteration represents one day
    newly_sick = newly_sick * 2
    total_sick = total_sick + newly_sick
  end
  return total_sick
end
```

Each newly infected person  
infects 2 people the next day.

## An epidemic (cont'd)

```
compute_sick(1) => 1      compute_sick(17) => 131071
compute_sick(2) => 3      compute_sick(18) => 262143
compute_sick(3) => 7      compute_sick(19) => 524287
compute_sick(4) => 15     compute_sick(20) => 1048575
compute_sick(5) => 31     compute_sick(21) => 2097151
compute_sick(6) => 63
compute_sick(7) => 127
compute_sick(8) => 255
compute_sick(9) => 511
compute_sick(10) => 1023
compute_sick(11) => 2047
compute_sick(12) => 4095
compute_sick(13) => 8191
compute_sick(14) => 16383
compute_sick(15) => 32767
compute_sick(16) => 65535
```

In just three weeks, over  
2 million people are sick!  
(This is what Blown To Bits  
means by *exponential growth*.  
We will see important  
computational problems that  
get exponentially "harder" as  
the problems gets bigger.)

## Countdown!

```
def countdown()
  for i in 1..10 do
    print 11-i
    print " "
    sleep 1      # pauses for 1 sec.
  end
end

countdown()
=> 10 9 8 7 6 5 4 3 2 1
```

Why can't we just use 10..1  
here and print i instead?