

UNIT 3C

Algorithmic thinking continues

Announcements

- PS2 is due Today Friday Feb 1 in class.
- PA3 (2/5) and PS3 (2/8) are out now
 - Topics covered
 - conditionals
 - Iterations and use of \ and %
 - ASCII art – nested loops
 - Counting and while loops
 - Tracing code
 - From algorithm to code
 - Flow charts

Return Statement

- A return statement is just like a print statement TRUE/FALSE
- What is the purpose of having a return statement in a function? to return a value back to calling program
- Can we have more than one return statement in a function?

yes but can return
only one

```
if (x > y) then  
    return x  
else  
    return y  
end
```

Return Statement ctd..

- Can we return more than once from a function?

```
X = 1  
return X →  
X = X + 2  
return X ← unreachable code
```

- Can we have a return statement inside a for or while loop?

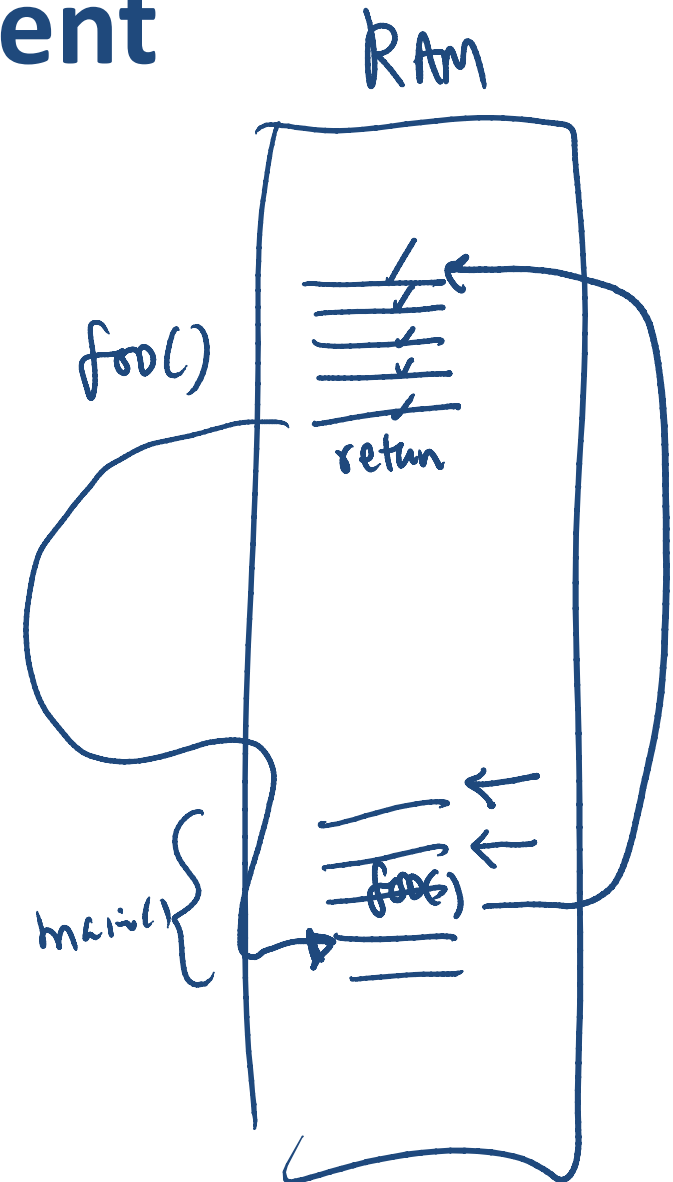
~~No~~
maybe

```
for i in 1..10 do  
  print i  
  return i  
end →
```

Return statement

- How does it work?

```
def main()  
    ||||  
    foo()  
    ||||  
end
```



Questions??

- What is the difference between PRINT and PUTS statements in Ruby?

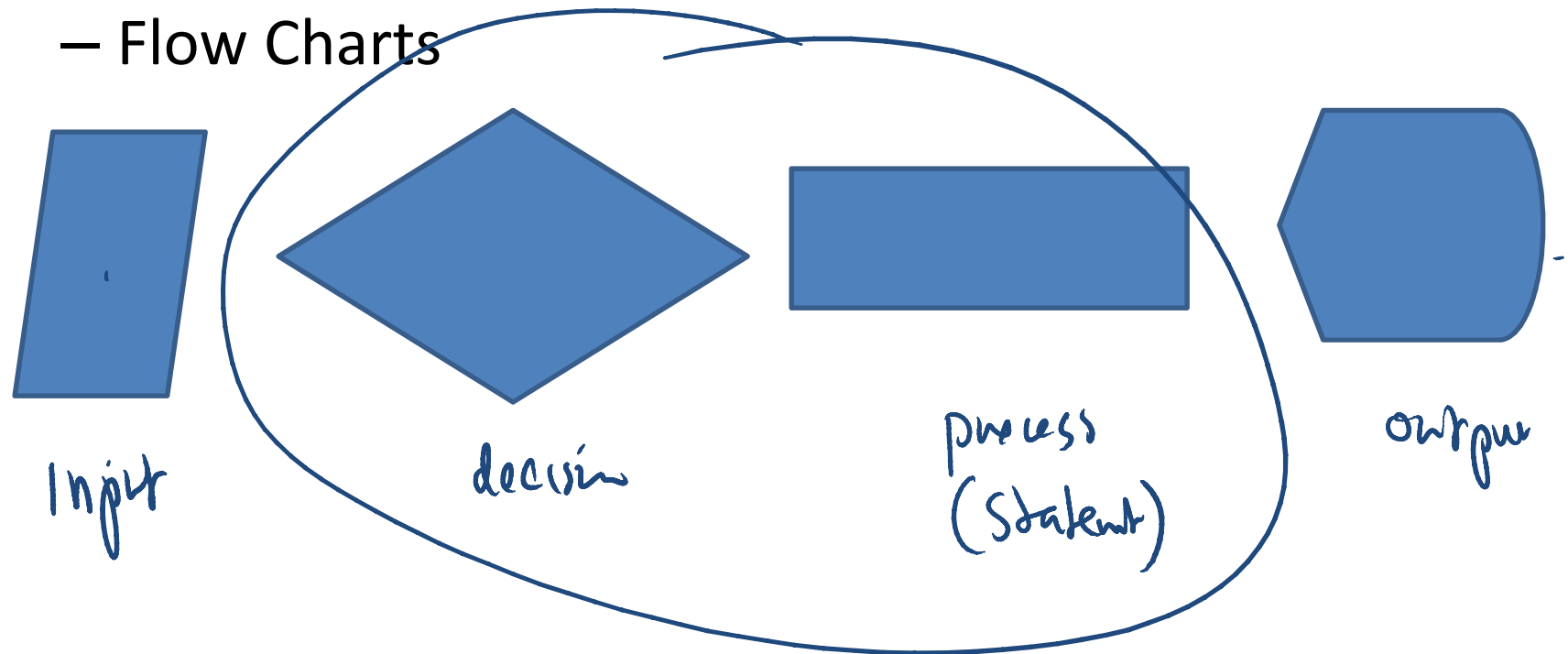
`print "B"` → B ^{Cursor} ↓

`puts "B"` → B `"\n"` ← cursor

Tools for Developing algorithms

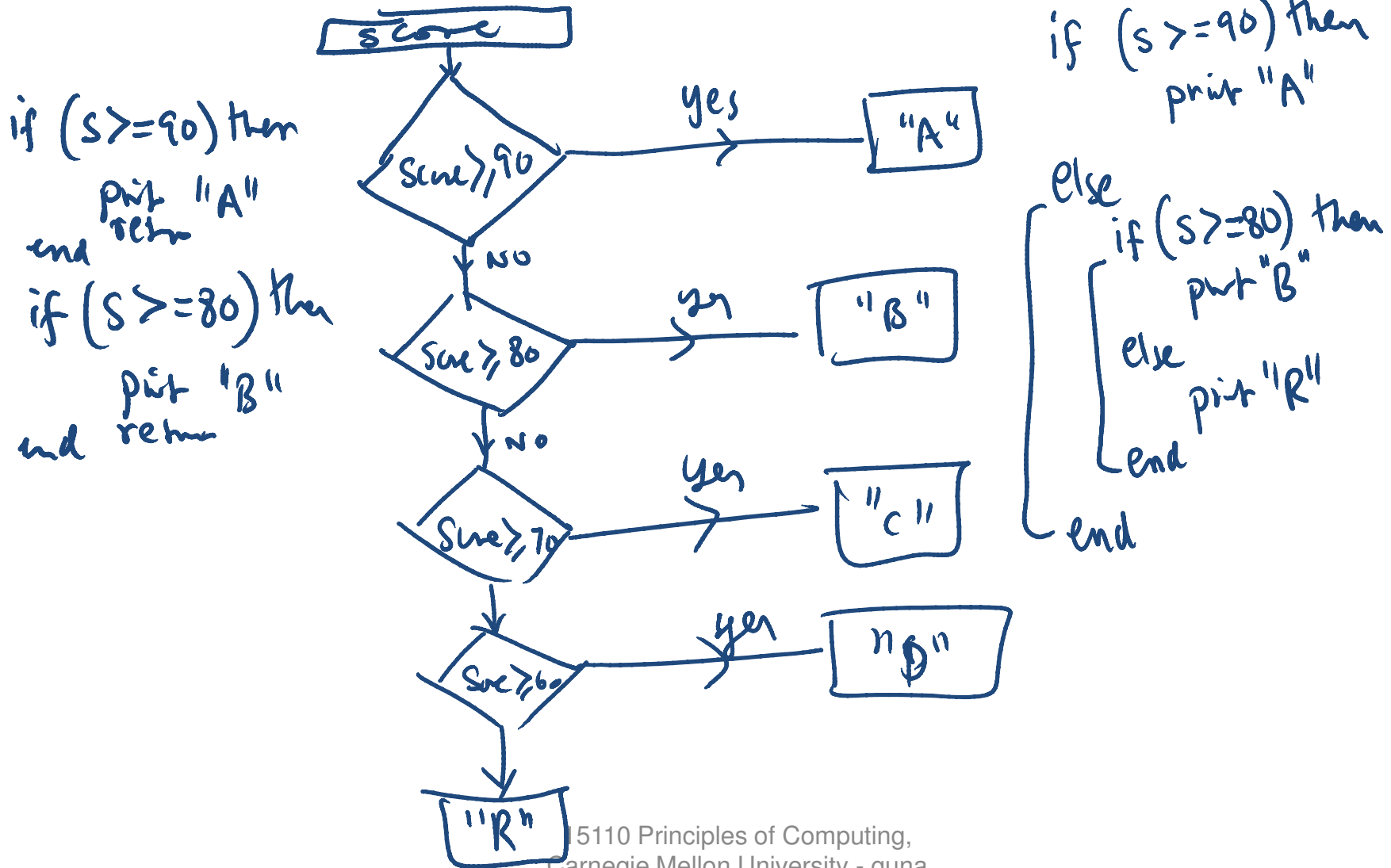
Developing algorithms

- An algorithms must be
 - correct, efficient, tested
- Tools for developing algorithms
 - Flow Charts



Draw a flow chart $Score \leq S$

- Given a score, print a grade



Tools for Implementing algorithms

Truth Tables

$$\neg (x \geq 3 \text{ and } x < 5) \longleftrightarrow \neg (x \geq 3) \text{ or } \neg (x < 5)$$

$\neg (x \geq 3) \text{ or } \neg (x < 5)$
 $\text{or } x < 3 \text{ or } x \geq 5$

AND, OR, NOT tables

AND	T	F
T	T	F
F	F	F

OR	T	F
T	T	T
F	F	F

NOT	T	F
T	F	T
F	T	F

$$\neg (x \geq 3) \longleftrightarrow x < 3$$

$$\neg \neg (x < 3) \text{ (circled)}$$

- DeMorgans Law

$$\neg (A \text{ and } B) = \neg A \text{ or } \neg B$$

$$\neg (A \text{ or } B) = \neg A \text{ and } \neg B$$

Relational Operators

- If we want to compare two integers to determine their relationship, we can use these relational operators:

< less than

<= less than or equal to

> greater than

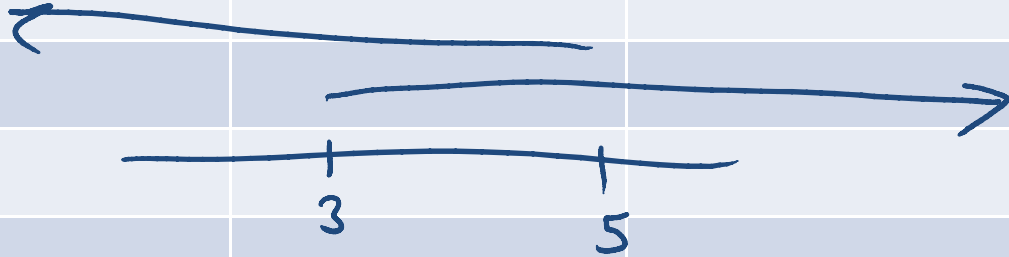
>= greater than or equal to

== equal to

!= not equal to

Examples

- Classify following statements as “always false” “always true” or if “sometimes true”, then provide a value(s).

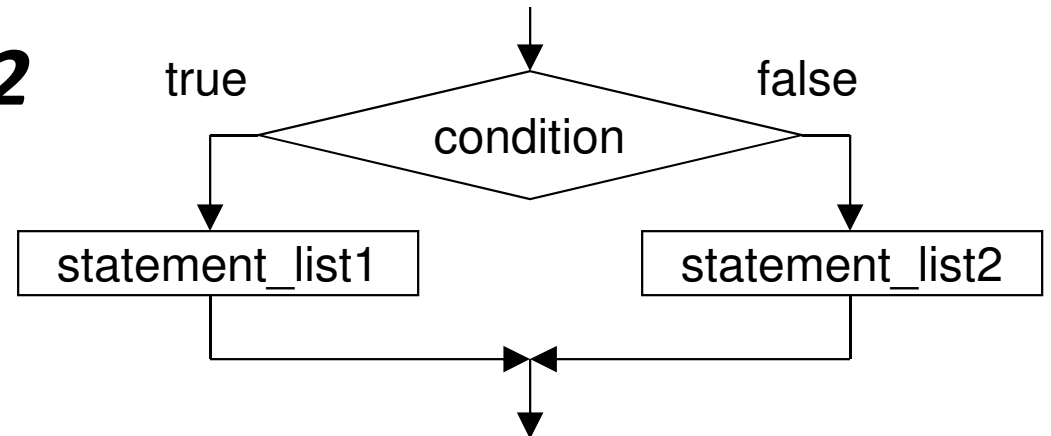
statement	Always false	Always true	Sometimes true
$3 == 5$	✓		
$X \geq 1$ or $X < 1$		✓	
$X > 1$ and $X < 1$	✓		
$X \geq 1$ and $X \leq 1$			$X = 1$
$X > 3$ and $X < 5$		✓	
			

Branching

if/else statement

Format:

```
if bool_condition then  
    statement_list1  
else  
    statement_list2  
end
```



**Write a function to print the grade
given a score**

skip

iteration

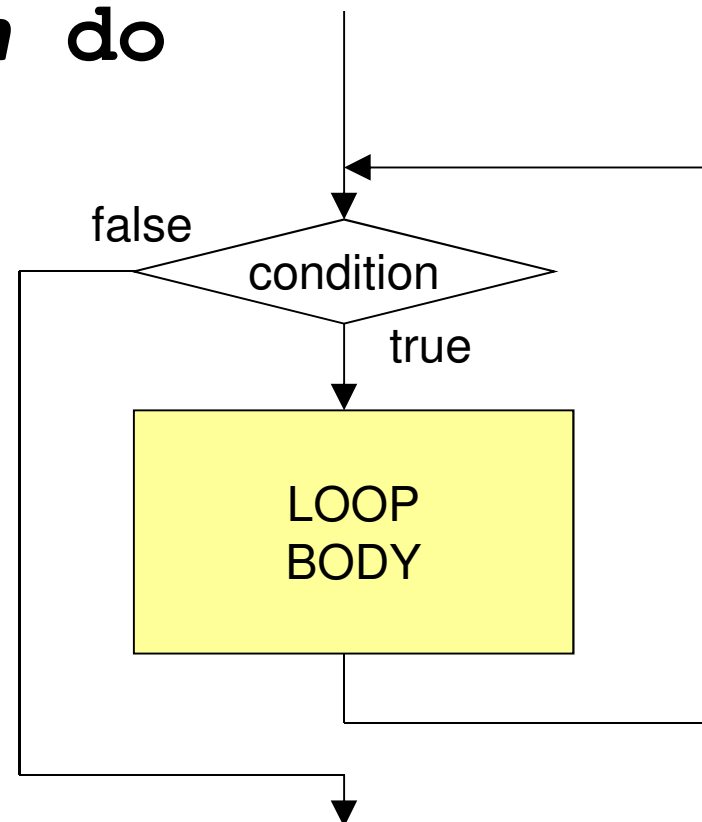
while loop

Format:

```
while bool_condition do  
    loop body  
end
```

one or more instructions
to be repeated

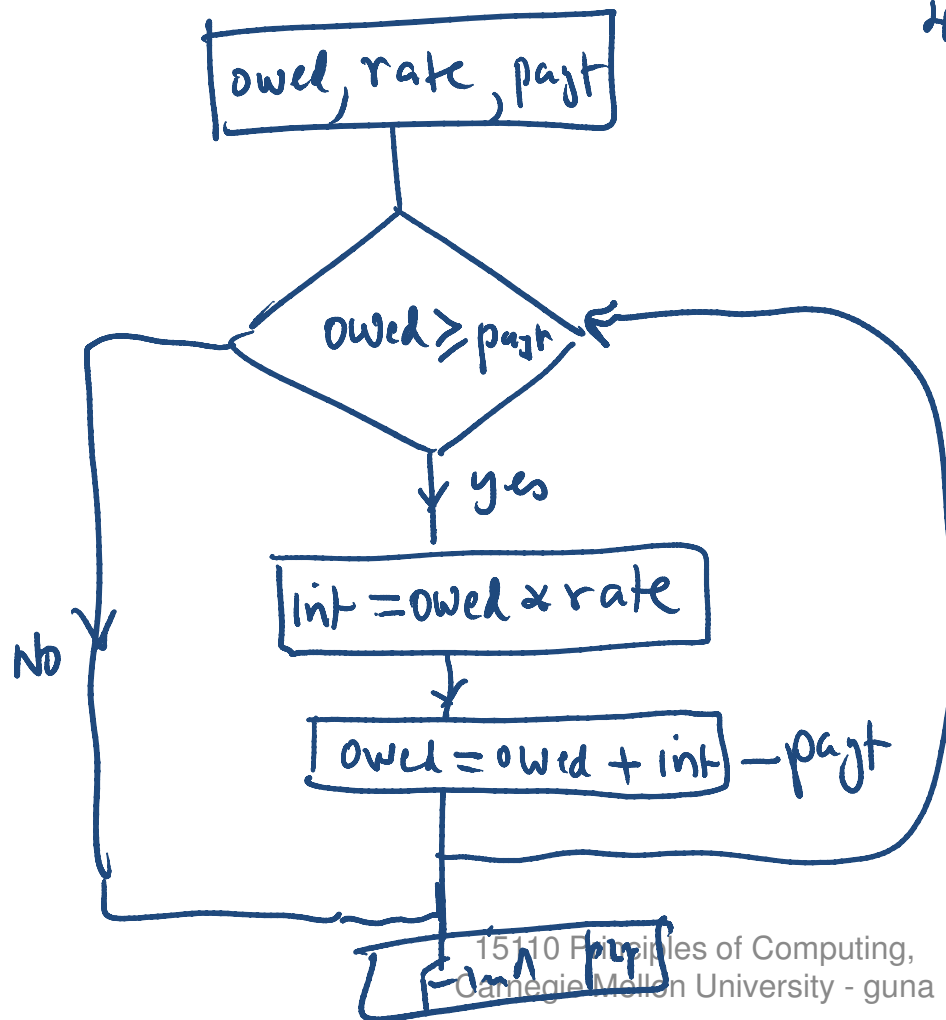
If the loop condition becomes false during the loop body, the loop body still runs to completion before we exit the loop and go on with the next step.



Examples

Interest calculation (flow chart)

- The problem:** Given an total amount “owed” and a monthly “rate”, find how many “payments” can be deducted from the total. Return the value

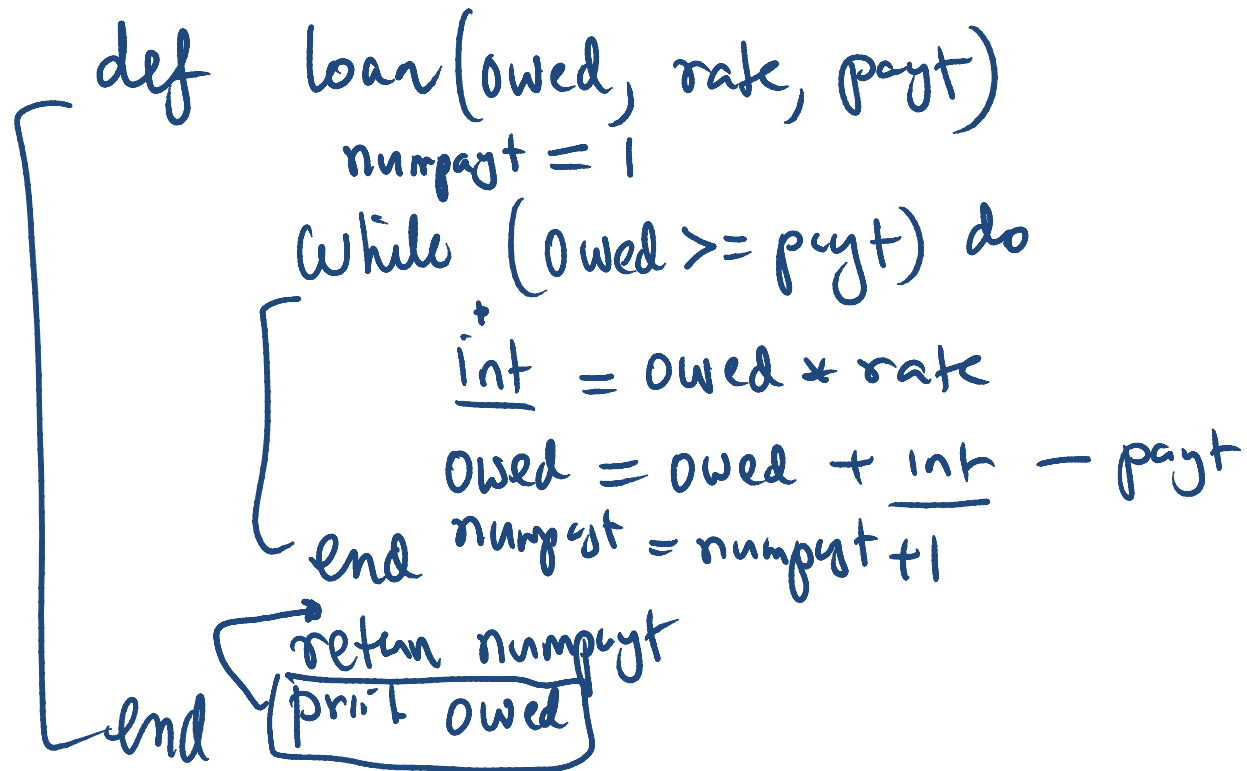


40,000 at 12% (year)

$$\begin{array}{r} 40\,000 + 400 = 40\,400 \\ \underline{-1\,000} \end{array}$$

Interest calculation (code)

```
def loan(owed, rate, payt)
    numpayt = 1
    while (owed >= payt) do
        +
        int = owed * rate
        owed = owed + int - payt
        numpayt = numpayt + 1
    end
    return numpayt
    print owed
end
```



**Write a function to check if a
number is prime**

**Use the isPrime function to print all primes
between any two numbers**

Nested Loops

ASCII ART

- How would you draw a skyscraper?
- How would you combine them to create a skyline?

Next week

- Arrays
- Algorithms on Arrays
- Complexity of algorithms