Iteration

James Gain
Department of Computer Science
University of Cape Town
jgain@cs.uct.ac.za

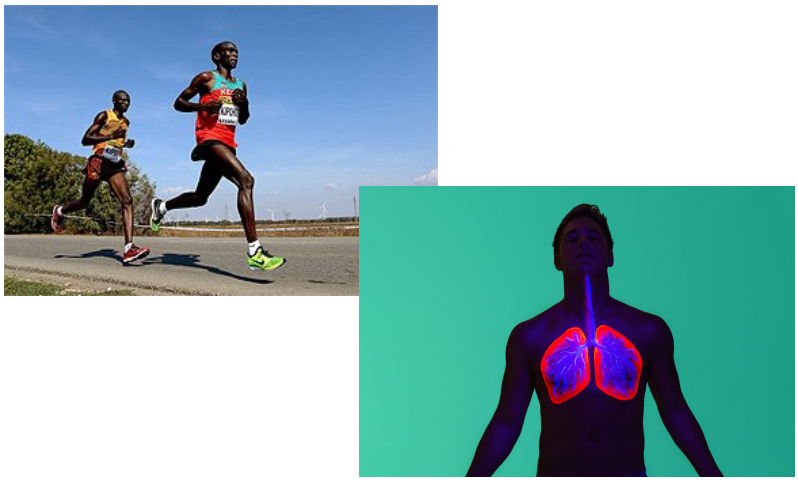
Problem 1 Introduction

- Output the 7x table for each of the values 1-12.

What is Iteration?

- Executing the same basic task or set of statements multiple times.
  - e.g., print the 7x table (from 1 to 12)
- Allows repetition in a program without having to type out the same statements multiple times.
  - Also enables variation with each repetition

Analogies



Counter-controlled Loops

- Counter-controlled loops execute for a fixed number of times.
- A special counter variable is assigned a different value each time and may be referred to within the loop.
- Python provides the "for" statement as a counter-controlled loop.

The "for" statement

In general

```
for <variable> in <something that creates a sequence>:
   <statement1>
   <statement2>
   ...
```

Typical form

```
for <variable> in range (<start>, <stop>, [<step>]):
   <statement1>
   <statement2>
   ...
```

Example

```
for x in range (1,10):
```

The `range` function

- `range` is a special function that generates a list of integers.
- Three different forms:
- `range(stop)`
  - begins at 0; ends just before stop; increments by 1 each time
- `range(start, stop)`
  - begins at start; ends just before stop; increments by 1 each time
- `range(start, stop, step)`
  - begins at start; ends just before stop; increments by step each time

The `range` function

- Examples:
- `range(10)`
  - [0,1,2,3,4,5,6,7,8,9]
- `range(5,12)`
  - [5,6,7,8,9,10,11]
- `range(3,9,2)`
  - [3, 5, 7]
- `range(8,1,-1)`
  - [8,7,6,5,4,3,2]

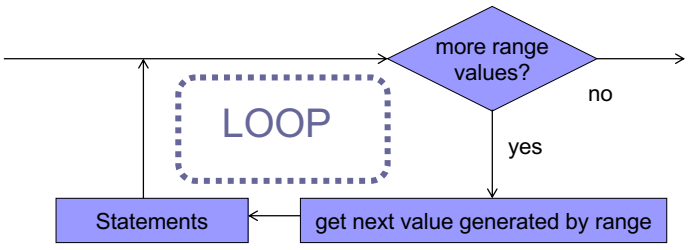
Problem 1 Solution

```
for n in range (1, 13):
     print (n,"x 7 =",(n*7))
```

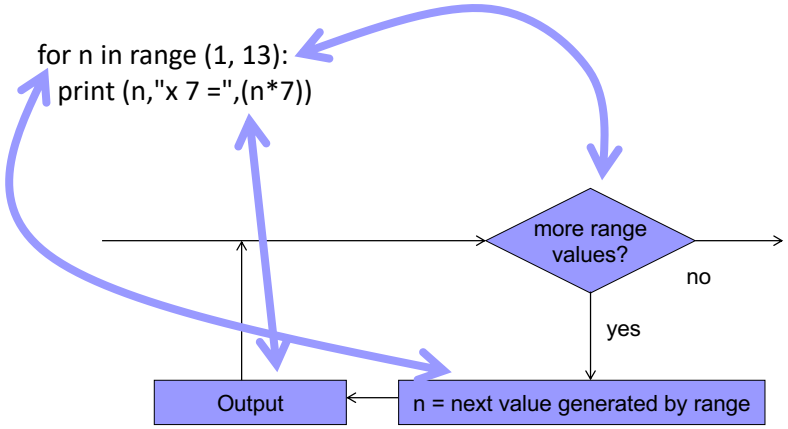
- Output:

```
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
...
```

General Semantics of "for"



Flowchart vs Python



Problem 2

- Output the `n x table` for any integer value of n.

Problem 2 Solution

```
n = eval (input("Enter the multiple: "))

for i in range (1,13):
   print (i,"x",n,"=",(n*i))
```

Problem 3

- Find the product of the integers from 1..n
  - This is the same as calculating n!.

Problem 4

- Calculate a^b using a **for** loop, assuming that a is a float and b is an integer.

Problem 5 Introduction

- Write programs to generate (on the screen) the following triangles of user-specified height:

```
*          *          ****
**        ***         ***
***      *****         **
****    *******         *
```

Nesting of statements

- **for** and **if** are both statements, therefore they can each appear within the statement body.

```
for i in range(10):
   if i>b:
      max = i

if a<b:
   for i in range(10):
      ...
for i in range(10):
   for j in range(10):
      ...
```

Nested loops

- Where a task is carried out multiple times and a subtask within that is carried out multiple times.
  - Example:
  - Draw a rectangle of arbitrary height/width on the screen, such as (height=4, width=3):

```
***
***
***
***
```

Problem 5

- Write programs to generate (on the screen) the following triangles of user-specified height:

```
*          *          ****
**        ***         ***
***      *****        **
****    *******       *
```

Problem 6 Introduction

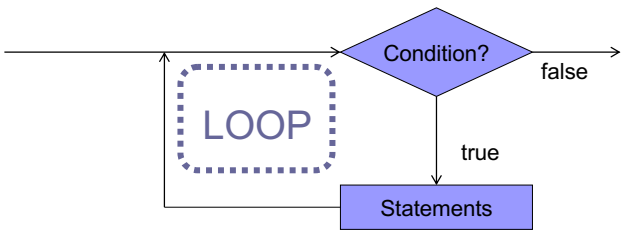
- Approximate the logarithm (with a base of 10) of an integer using repeated division.

Condition-controlled Loops

- If we do not know the number of iterations a priori (in advance), we can use a condition-controlled (or event-controlled) loop
  - Where the loop executes while a condition is true.
- Syntax:

```
while <condition>:
    <statement1>
    <statement2>
    ...
```

"while" Example




```
sum = 0
num = eval(input())
while num != 0:
    sum = sum + num
    num = eval(input())
print(sum)
```

Problem 6

- Approximate the logarithm (with a base of 10) of an integer using repeated division.

Poll: Loop Equivalence

- Which of the following loops end with the same value of `fac`:

```
a) fac = 1
   for i in range(1, n+1):
      fac *= i
b) fac = 1
   for i in range(n, 0, -1):
      fac = fac * i
c) fac = 1
   i = 1
   while i <= n:
      fac *= i
```

- a) and b)
- a) and c)
- All of them
- None of them

Problem 7

- Approximate the logarithm (with a base of 10) of an integer using repeated division.
- Design a user interface where the user can continue to ask for logarithms until the user enters a value of 0.

Problem 8

- Create an interactive menu to select and print out sandwich ingredients.

Menus

- A menu is a list of choices presented to the user, with the means to select one.
  - Example:

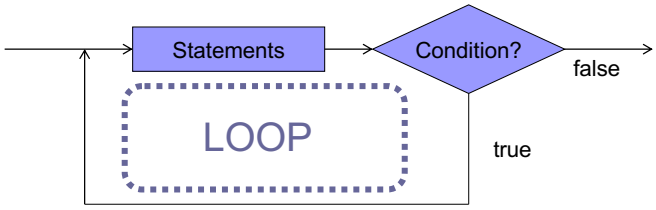
```
Souper Sandwich Menu
1. Chicken, cheese and chilli sauce
2. Chicken and chili
3. Chicken
4. Exit Program
Enter the sandwich number:
```

Menu Example

```
choice = eval (input ("Enter choice: "))
# get selection
while choice!=4: # continue until exited
   print("Add chicken") # print ingredients
   if choice<3:
      print("Add cheese")
   if choice<2:
      print ("Add chili")
   print() # leave a line
   choice = eval(input("Enter choice: "))
   # get selection again
```

Post-check loop

- When the "loop body" is going to be executed at least once, we should check the condition after the loop (instead of before).
- No statement in Python for this. Can we do still do it?



Problem 9

- Find the reverse of an integer.
- For example, the reverse of the integer 12345 is 54321 and the reverse of 98 is 89. Use only integer manipulations - do not convert the number to a string.

Other Loopy Techniques

- Infinite loops
- `break`
- `continue`
- `else`
- `pass`

Infinite Loops

- A loop where the condition is always true
  - This is often an error
- Example:

```
while True:
    print("Wheeee!")
```

break

- Exits immediately from a loop
  - Best to avoid if possible
- Example:

```
i = 0
while True:
   i+=1
   print(i)
   if i == 10:
       break
```

continue

- Immediately starts next iteration
- Example:

```
for i in range(10):
   if (i % 3 == 0):
      continue
   print(i)
```

else

- Execute at end of loop that ends normally (not **break**)
- Example:

```
for i in range(10):
    print(i)
else:
    print("done")
```

pass

- Do nothing
- Example:

```
for i in range (10):
   if (i % 3 == 0):
      pass
   else:
      print(i)
```

Selecting Loops

- When you know the number of iterations
  - use a **counter-controlled** or **definite** loop such as "*for*"
- When the iterations depend on a condition
  - use a **conditional** of **indefinite** loop such as "*while*"

Converting Loops

- How do we write the equivalent of
  - "while" using "for"?
  - "for" using "while"?

Problem 10

- Write a program to calculate the value of `Sin(x)` for any real value of x. Use the infinite Taylor series approximation:

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}.$$

- Extend your program to draw a `Sin(x)` graph using ASCII art.