CS 161: Introduction to Programming and Problem-solving

Warren Harrison

Control Flow and Looping

Control Flow

- Execution sequence of instructions
- Types of control flow
  - Sequence
  - Looping
  - Choice
- Decision-making
Doing the Same Set of Instructions Over and Over

```python
salesTot = 0
salesAmt = int(input(“Enter Store Sale “))
salesTot = salesTot + salesAmt
salesAmt = int(input(“Enter Store Sale “))
salesTot = salesTot + salesAmt
salesAmt = int(input(“Enter Store Sale “))
salesTot = salesTot + salesAmt
salesAmt = int(input(“Enter Store Sale “))
salesTot = salesTot + salesAmt
print(“Total Sales:“, salesTot)
```

Accumulation

- common activity when writing programs
- often used in summations of values or counting
- **counter variable** – initialized to a starting value
- **increment (or decrement) statement** adds (or subtracts) a particular value to the counter variable: `counter = counter + 1`
Doing the Same Set of Instructions Over and Over

```python
salesTot = 0
salesAmt = int(input("Enter Store Sale "))
salesTot = salesTot + salesAmt
salesAmt = int(input("Enter Store Sale "))
salesTot = salesTot + salesAmt
salesAmt = int(input("Enter Store Sale "))
salesTot = salesTot + salesAmt
salesAmt = int(input("Enter Store Sale "))
salesTot = salesTot + salesAmt
print("Total Sales:",salesTot)
```

Limitations

- Can only collect sales from exactly four stores – not three and not five – ‘Hard coded'  
- Write extra code  
  - tedious  
  - easier to make a mistake  
  - if you made a mistake, you might have to change all occurrences of the mistake – perhaps in every statement
Looping

- Causes a statement or set of statements to be repeatedly executed:

```python
salesAmt = int(input("Enter Store Sale "))
salesTot = salesTot + salesAmt
```

The While Loop

```python
storeCount=0
salesTot = 0
while(storeCount < 4):
    salesAmt=int(input("Enter Store Sale "))
salesTot = salesTot + salesAmt
    storeCount = storeCount + 1
print("Total Sales: ",salesTot)
```
The While Loop

continuation condition

storeCount=0
salesTot = 0
while(storeCount < 4):
    salesAmt=int(input("Enter Store Sale "))
    salesTot = salesTot + salesAmt
    storeCount = storeCount + 1
print("Total Sales:",salesTot)

The While Loop

sentry variable

storeCount=0
salesTot = 0
while(storeCount < 4):
    salesAmt=int(input("Enter Store Sale "))
    salesTot = salesTot + salesAmt
    storeCount = storeCount + 1
print("Total Sales:",salesTot)
Continuation Condition

- The loop continues looping while the condition is true – what conditions?
  - $a < b$ less than
  - $a > b$ greater than
  - $a == b$ equal in value
  - $a != b$ not equal in value
- *It is very important that the condition will eventually become false …*

Relational Operators

- A Relational Operator asserts that some relationship holds between two items
- $<, \leq, >, \geq, ==, !=$
- The assertion may be TRUE or FALSE

- $a = 4, b = 3, c = 12, d = 4$
  - $a < b$
  - $a == d$
  - $c!=d$
  - $a >= b$
The While Loop
... but counting down

storeCount = 4
salesTot = 0
while(storeCount != 0):
    salesAmt = int(input("Enter Store Sale "))
    salesTot = salesAmt
    storeCount = storeCount - 1
print("Total Sales:", salesTot)

Looping a Defined Number of Times

counter=0
while(counter < limit):
    <loop body>
    counter = counter + increment
Using a Code Tracing Chart

- Make each variable a column head
- Make a column for each condition
- Make a column for output

<table>
<thead>
<tr>
<th>storeCount</th>
<th>salesTot</th>
<th>salesAmt</th>
<th>storeCount != 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tracing Your Code

- Walk through code one line at a time
- Each time a variable changes, change it on the chart
- Each time you get to a condition statement, evaluate the condition
- For conditions, write TRUE or FALSE
Looping Until Something Happens

totalSales = 0
storeName = input("Enter Store Name ")
while(storeName != "QUIT"):
    storeSales = int(input("enter sales"))
    totalSales = totalSales + storeSales
    storeName = input("Enter Store Name ")

print("Total Sales: ",totalSales)

Loop Until Match

• Doesn’t Use a Counter
• Waits until there is a “match”

while(match not found):
    <loop body>

• Need to make sure that a match can occur
The Match Will Never Happen
Why?

```python
totalSales = 0

storeName = input("Enter Store Name ")
while(storeName != "QUIT"):
    storeSales = int(input("enter sales"))
    totalSales = totalSales + storeSales

print("Total Sales: ",totalSales)
```

Relational Operators and Strings

- Relationships between strings are based on the **ASCII collating sequence**.
- Usually makes sense
  - “a” < “b”
  - “adam” < “andrew”
  - “2” < “3”
  - “1000” < “2” **What?!?!?!**
ASCII Codes

Sales Report

totalSales = storeCount = 0
storeName = input("Enter Store Name ")
while(storeName != "QUIT"):
    storeSales = int(input("Enter Sales "))
    totalSales = totalSales + storeSales
    storeCount = storeCount + 1
storeName = input("Enter Store Name ")
avgSales = totalSales/storeCount
print("Total Sales: ",totalSales)
print("Average Sales per Store: ",avgSales)