Why loops?

- Computers offer several advantages over calculators.
- If it is necessary, they can perform the same steps over and over again, simply by rerunning the program.
- But is this the only way to get a computer to perform the same action repeatedly? And is this the only reason for getting a computer to repeat itself?
Example : Average of three numbers

• Let’s take another look at our program that finds the average of three numbers:
  ```python
  value1 = int(input("What is the first value?"))
  value2 = int(input("What is the second value?"))
  value3 = int(input("What is the third value?"))
  sum = value1 + value2 + value3
  average = sum / 3
  print("Average = ", average)
  ```
• What would we do if we wanted the program to average 5 values instead of 3? or 10? or 100?
• This is clearly not the best way to write this!

Loops

• We need the ability to perform the same set of instructions repeatedly so we don’t have to write them over and over again.
• This is why Python includes several ways of using repetition in a program.
• Each case where we repeat a set of statement is called a loop.
Counting Loops

• The first type of loop is a counting loop.
• Counting loops are repeated a specific number of times.
• If you read the loop, you can easily figure out how many times its statements will be performed.

Example: Hello Again

• Example - Write a program that greets the user with "Hi there!" five times.
• We could write the program like this:
  
  # Hello again - this program writes
  # "Hello, again" five times
  print("Hello, again")
  print("Hello, again")
  print("Hello, again")
  print("Hello, again")
  print("Hello, again")
Counting Loops

• We use a for loop to write basic counting loops

• In Python, it looks like this:
  ```python
  for count in range(size) :
    statements
  ```
  or
  ```python
  for count in range(start, size) :
    statements
  ```
  or
  ```python
  for count in range(start, size, increment) :
    statements
  ```
  or
  ```python
  for count in range(size, increment) :
    statements
  ```

Counting Loops (continued)

```python
for count in range(start, size, increment) :
    statements
```

variable used to count times through the loop

number of loops

initial value of the counter

increment of the counter
**for** Loops - Examples

```python
for i in range(3):
    print(i, " ", end="")
print()

for i in range(1, 3):
    print(i, " ", end="")
print()

for i in range(1, 6, 2):
    print(i, " ", end="")
print()
```

Output

```
0  1  2
1  2
1 3  5
```

---

**Example: Rewriting HelloAgain**

- Let's write the steps that our new version of that program must perform:

  1. Write "Hi, there!" on the screen 5 times.

    1. FOR i goes from 1 TO 5
    1.1 Write “Hi, there!”
Refining *HelloAgain*

1. **FOR** i goes from 1 **TO** 5
1.1 Write “Hi, there!”

```python
for i in range(5):

print("Hello, again")
```
The New *HelloAgain*

```python
# HelloAgain2 - This is a better way to write
#              "Hello, again" five times

for i in range(5):
    print("Hello, again")
```

Generalizing *HelloAgain*

- This program is also flawed; it gives us no choices as to how many times we can print “Hi, there!”
- We can to let the user select how many times to print the message and making this version of the program more general is fairly easy:
- Our algorithm will start as:
  1. Find out how many time to print the message.
  2. Print "Hi, there!" that many times.
Generalizing *HelloAgain* (continued)

1. Find out how many time to print the message.
2. Print "Hi, there!" that many times.

```python
totalTimes = int(input
("How many times do you want to say \"hello\"?
))
```

Generalizing *HelloAgain* (continued)

```python
2. Print "Hi, there!" that many times.

print("Hello, again")
```
The Revised *HelloAgain*

```python
# HelloAgain3 - Write "Hello, again" as many times
# as the user wants

totalTimes = int(input
    ("How many times do you want to say \"hello\"?
))

for count in range(totalTimes):
    print("Hello, again")
```

Example: Averaging $n$ Numbers

- Let's get back to our original problem. We want to able to average any number of values.
- Let's start by outlining our algorithm:
  1. Find out how many values there are.
  2. Add up all the values.
  3. Divide by the number of values
  4. Print the result
**Refining Avg**

1. Find out how many values there are.
2. Add up all the values.
3. Divide by the number of values
4. Print the result

```python
numValues = int(input
("How many values are you going to enter?"))
```

2. Add up all the values.
3. Divide by the number of values
4. Print the result

2.1 For CurrentValue goes from 1 to NumValues :
2.1.1 Get the next value
2.1.2 Add it to the total
Refining Avg

```python
numValues = int(input("How many values are you going to enter?"))
```

2.0 Set the total to zero (initially there are no values)

2.1 For CurrentValue goes from 1 to NumValues:

2.1.1 Get the next value

2.1.2 Add it to the total

3. Divide by the number of values

4. Print the result

```python
sum = 0.0;
for currentValue in range(numValues):
    value = float(input("What is the next value?"))
    sum = sum + value
```

```
```
Refining Avg

```python
numValues = int(input("How many values are you going to enter?"))
sum = 0.0;
for currentValue in range(numValues) :
    value = float(input("What is the next value?"))
    sum = sum + value
3. Divide by the number of values
4. Print the result
average = sum / numValues
print("The average is ", average)
```

The AverageN Program

```python
# AverageN - Find the average of N values

# Find out how many values there are
numValues = int(input("How many values are you going to enter?"))

# Read in each value and add it to the sum
sum = 0.0;
for currentValue in range(numValues) :
    value = float(input("What is the next value?"))
    sum = sum + value

# Calculate and print out the average
average = sum / numValues
print("The average is ", average)
```
Example: Interest Program

- **Example** - Write a program that calculates the interest that the Canarsie Indians would have accumulated if they had put the $24 that they had received for Manhattan Island in the bank at 5% interest.

  **Input** - none; all the values are fixed

  **Output** - Year and Principle

  **Other Information** -
  - Principle is initially 24
  - Interest = Interest Rate * Principle
  - New Principle = Old Principle + Interest

---

Example: Interest Program

- Our initial algorithm is:
  1. Set the principle to 24
  2. For every year since 1625, add 5% interest to the principle and print out the principle.
Refining The Interest Algorithm

1. Set the principle to 24
2. For every year since 1625, add 5% interest to the principle and print out the principle.

   2.1 FOR Year goes from 1625 TO Present:
      2.1.1 Add 5% interest to the principle
      2.1.2 Print the current principle

   2.1.1.1 Calculate 5% Interest
   2.1.1.2 Add the interest to the principle
Refining The Interest Algorithm

1. Set the principle to 24

2.1 FOR Year goes from 1625 TO Present:
   2.1.1.1 Calculate 5% Interest
   2.1.1.2 Add the interest to the principle
   2.1.2 Print the current principle

```python
principle = 24
for year in range(1625, present):
    # Calculate 5% Interest
    interest = principle * 0.05
    principle += interest
    # Add the interest to the principle
    # Print the current principle
```

---

Refining The Interest Algorithm

```python
principle = 24;
for year in range(1625, present):
    # Calculate 5% Interest
    interest = principle * 0.05
    principle += interest
    # Add the interest to the principle
    # Print the current principle
```
Refining The Interest Algorithm

```python
principle = 24;
for year in range(1625, present):
    interest = rate * principle
    principle = principle + interest
    2.1.2 Print the current principle
    print("year = ", year, "\tprinciple = ", principle)
```
The Interest Program

# Calculate the interest that the Canarsie Indians could have accrued if they had deposited the $24 in an bank account at 5% interest.
present = 2015
rate = 0.05;

# Set the initial principle at $24
principle = 24

# for every year since 1625, add 5% interest to the principle and print out the principle
for year in range(1625, present):
    interest = rate * principle
    principle = principle + interest
    print("year = ", year, "\tprinciple = ", principle)
Output from the Compound Interest Program

• What will our output look like?
  year = 1625 principle = 25.2
  year = 1626 principle = 26.46
  year = 1627 principle = 27.783
  year = 1628 principle = 29.172150000000002
     ... ... ... ...
  year = 2010 principle = 3624771902.2233915
  year = 2011 principle = 3806010497.3345613
  year = 2012 principle = 3996311022.201289
  year = 2013 principle = 4196126573.3113537
  year = 2014 principle = 4405932901.976921

• This does not look the way we expect monetary amounts to be written!

Formatted Output With print()

• The method print() gives us a way to write output that is formatted, i.e., we can control its appearance.

• We write:
  \texttt{print(ControlString, \%(Arg1, Arg2, \ldots \))}

• The control string is a template for our output, complete with the text that will appear along with whatever values we are printing.
System.out.printf(): Some Simple Examples

- `print()` will print whatever is in the control string with a few exceptions:

  ```java
  print("I paid $%4.2d for that shirt" % (dollars))
  ```
  
  will produce:

  ```java
  I paid $%4.2d for that shirt
  ```

Special Characters

- There are a number of special characters that all begin with a backslash:
  - `\n` new line
  - `\b` backspace
  - `\t` tab

- These can appear anywhere with a string of characters:

  ```java
  System.out.printf("This is a test\nIt is!!\n");
  ```
%d and %f

- The specifiers %d and %f allow a programmer to specify how many spaces a number will occupy and (in the case of float values) how many decimal places will be used.
- %n%d will use at least n spaces to display the integer value in decimal (base 10) format.
- %w.%df will use at least w spaces to display the value and will have exactly d decimal places.

Changing the width

<table>
<thead>
<tr>
<th>Number</th>
<th>Formatting</th>
<th>Print as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>182</td>
<td>%2d</td>
<td>182</td>
</tr>
<tr>
<td>182</td>
<td>%3d</td>
<td>182</td>
</tr>
<tr>
<td>182</td>
<td>%5d</td>
<td>```182</td>
</tr>
<tr>
<td>182</td>
<td>%7d</td>
<td>```182</td>
</tr>
<tr>
<td>-182</td>
<td>%4d</td>
<td>-182</td>
</tr>
<tr>
<td>-182</td>
<td>%5d</td>
<td>`-182</td>
</tr>
<tr>
<td>-182</td>
<td>%7d</td>
<td>```-182</td>
</tr>
</tbody>
</table>
### Changing the width (continued)

<table>
<thead>
<tr>
<th>Number</th>
<th>Formatting</th>
<th>Print as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>%1d</td>
<td>23</td>
</tr>
<tr>
<td>23</td>
<td>%2d</td>
<td>23</td>
</tr>
<tr>
<td>23</td>
<td>%6d</td>
<td>....23</td>
</tr>
<tr>
<td>23</td>
<td>%8d</td>
<td>......23</td>
</tr>
<tr>
<td>11023</td>
<td>%4d</td>
<td>11023</td>
</tr>
<tr>
<td>11023</td>
<td>%6d</td>
<td>.11023</td>
</tr>
<tr>
<td>-11023</td>
<td>%6d</td>
<td>-11023</td>
</tr>
<tr>
<td>-11023</td>
<td>%10d</td>
<td>.....11023</td>
</tr>
</tbody>
</table>

### Changing The Precision

<table>
<thead>
<tr>
<th>Number</th>
<th>Formatting</th>
<th>Prints as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.718281828</td>
<td>%8.5f</td>
<td><code>2.71828</code></td>
</tr>
<tr>
<td>2.718281828</td>
<td>%8.3f</td>
<td><code>\</code>`2.718`</td>
</tr>
<tr>
<td>2.718281828</td>
<td>%8.2f</td>
<td><code>\</code>``2.72`</td>
</tr>
<tr>
<td>2.718281828</td>
<td>%8.0f</td>
<td><code>\</code>````3`</td>
</tr>
<tr>
<td>2.718281828</td>
<td>%13.11f</td>
<td>2.71828182800</td>
</tr>
<tr>
<td>2.718281828</td>
<td>%13.12f</td>
<td>2.718281828000</td>
</tr>
</tbody>
</table>
The revised *Compound* program

```
# Calculate the interest that the Canarsie
# Indians could have accrued if they had
# deposited the $24 in a bank account at
# 5% interest.
present = 2015
rate = 0.05;

# Set the initial principle at $24
principle = 24;

# For every year since 1625, add 5% interest
# to the principle and print out
# the principle

for year in range(1625, present) :
    interest = rate * principle;
    principle = principle + interest;

    print("year = %4d\tprinciple = $%13.2f"
          %(year, principle))
```
The output from the Revised Compound Program

Our output now looks like this:

\[
\begin{align*}
\text{year} & = 1625 \quad \text{principle} = \$ 25.20 \\
\text{year} & = 1626 \quad \text{principle} = \$ 26.46 \\
\text{year} & = 1627 \quad \text{principle} = \$ 27.78 \\
\text{year} & = 1628 \quad \text{principle} = \$ 29.17 \\
\ldots & \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\
\text{year} & = 2010 \quad \text{principle} = \$ 3624771902.22 \\
\text{year} & = 2011 \quad \text{principle} = \$ 3806010497.33 \\
\text{year} & = 2012 \quad \text{principle} = \$ 3996311022.20 \\
\text{year} & = 2013 \quad \text{principle} = \$ 4196126573.31 \\
\text{year} & = 2014 \quad \text{principle} = \$ 4405932901.98
\end{align*}
\]

Integer Division

- Our compound interest program prints the values for every year where every ten or twenty years would be good enough.
- What we really want to print the results only if the year is ends in a 5. (The remainder from division by 10 is 5).
• There are two types of division where the dividend and divisor are both integers.
• Floor by an integer produces an integer quotient, which is the largest integer smaller than the quotient:

<table>
<thead>
<tr>
<th>Dividend</th>
<th>Quotient</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

• In Python, the // operator produces an integer quotient for integer division.
• If you want the remainder from integer division, you want to use the % operator.
# A few examples of integer division using 
# // and %
print("8 / 3 = ", 8 / 3 )
print("8 // 3 = ", 8 // 3 )
print("8 % 3 = ", 8 % 3 )

print("2 / 3 = ", 2 / 3 )
print("2 // 3 = ", 2 // 3 )
print("2 % 3 = ", 2 % 3 )

print("49 // 3 = ", 49 // 3 )
print("49 % 3 = ", 49 % 3 )

print("49 // 7 = ", 49 // 7 )
print("49 % 7 = ", 49 % 7 )

print("-8 // 3 = ", -8 // 3 )
print("-8 % 3 = ", -8 % 3 )

print("-2 // 3 = ", -2 // 3 )
print("-2 % 3 = ", -2 % 3 )

print("-2 // -3 = ", -2 // -3 )
print("-2 % -3 = ", -2 % -3 )

print("2 // -3 = ", 2 // -3 )
print("2 % -3 = ", 2 % -3 )

print("-49 // 3 = ", -49 // 3 )
print("-49 % 3 = ", -49 % 3 )

print("-49 // -3 = ", -49 // -3 )
print("-49 % -3 = ", -49 % -3 )

print("49 // -3 = ", 49 // -3 )
print("49 % -3 = ", 49 % -3 )
print("-49 // 7 = ", -49 // 7 )
print("-49 % 7 = ", -49 % 7 )

print("-49 // -7 = ", -49 // -7 )
print("-49 % -7 = ", -49 % -7 )

print("49 // -7 = ", 49 // -7 )
print("49 % -7 = ", 49 % -7 )

Integer Division Results

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>49</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>-8</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>-2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>-2</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>-49</td>
<td>3</td>
<td>-16</td>
</tr>
</tbody>
</table>
Integer Division Results (continued)

<table>
<thead>
<tr>
<th>-49 // -3 = 16</th>
<th>-49 % -3 = -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>49 // -3 = -16</td>
<td>49 % -3 = 1</td>
</tr>
<tr>
<td>-49 // 7 = -7</td>
<td>-49 % 7 = 0</td>
</tr>
</tbody>
</table>

Final Compound Interest Program

```python
# Calculate the interest that the Canarsie Indians could have accrued if they had deposited the $24 in a bank account at 5% interest.
present = 2015;
rate = 0.05;

# Set the initial principle at $24
principle = 24;

# For every year since 1625, add 5% interest to the principle and print out the principle
for year in range(1625, present):
    interest = rate * principle
    principle = principle + interest
```
A program to calculate Grade Point Average

Example – Professor Smith gives n tests during the term and uses a grading system, where each test is 1/n of the course grade. Assuming that the average of the test grades translate into a letter grade as follows:

<table>
<thead>
<tr>
<th>Test Average</th>
<th>Letter Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>90.0+</td>
<td>A</td>
</tr>
<tr>
<td>80–89.9</td>
<td>B</td>
</tr>
<tr>
<td>70–79.9</td>
<td>C</td>
</tr>
<tr>
<td>60–69.9</td>
<td>D</td>
</tr>
<tr>
<td>below 60.0</td>
<td>F</td>
</tr>
</tbody>
</table>

write a program that will calculate a student’s grade.
A Program To Calculate Test Average

**Input** - Number of tests and the student's test grades

**Output** – Test average and course grade

**Other information**

A 90+ average is an “A”.
A 80-90 average is a “B”.
A 70-80 average is a “C”.
A 60-70 average is a “D”
An average below 60 is an “F”.

Test average = Sum of the test grade/ Number of tests

Our first step is to write out our initial algorithm:

1. Find the number of tests
2. Find the average of *n* tests
3. Find the corresponding letter grade and print it out.

Designing the Grade Program

1. Find the number of tests
2. Find the average of *n* tests
3. Find the corresponding letter grade and print it out.

2.1 Add up the test grades
2.2 Divide the total by *n*
Refining the Grade Program

1. Find the number of tests
   2.1 Add up the test grades
   2.2 Divide the total by $n$
3. Find the corresponding letter grade and print it out.

2.1 For each of the $n$ tests:
   2.1.1 Read the test grade
   2.1.2 Add it to the total

```
numTests = int(input
    ("How many tests did you take ?"))
```
Refining the Grade Program

\text{numTests} = \text{int}(\text{input} ("\text{How many tests did you take ?"}))

2.1 For each of the \(n\) tests:
  2.1.1 Read the test grade
  2.1.2 Add it to the total
  2.2 Divide the total by \(n\)

3. Find the corresponding letter grade and print it out.

\text{for thisTest in range(numTests) :}

Refining the Grade Program

\text{numTests} = \text{int}(\text{input} ("\text{How many tests did you take ?"}))

\text{for thisTest in range(numTests) :}

  2.1.1 Read the test grade
  2.1.2 Add it to the total
  2.2 Divide the total by \(n\)

3. Find the corresponding letter grade and print it out.

\text{thisGrade} = \text{int}(\text{input} ("\text{What grade did you get on this test ?"}))
Refining the Grade Program

numTests = int(input
    ("How many tests did you take ?"))
for thisTest in range(numTests) :
    thisGrade = int(input
        ("What grade did you get on this test ?"))
        \[2.1.2\quad \text{Add it to the total}\]
    total = total + thisGrade
    \[2.2\quad \text{Divide the total by } n\]
3. Find the corresponding letter grade and print it out.

total = total + thisGrade
Refining the Grade Program

```python
numTests = int(input("How many tests did you take ?"))
total = 0
for thisTest in range(numTests):
    thisGrade = int(input("What grade did you get on this test ?"))
    total = total + thisGrade
2.2 Divide the total by n
3. Find the corresponding letter grade and print it out.

testAverage = total/numTests
```

Refining the Grade Program

```python
numTests = int(input("How many tests did you take ?"))
total = 0
for thisTest in range(numTests):
    thisGrade = int(input("What grade did you get on this test ?"))
    total = total + thisGrade

testAverage = total/numTests;
3. Find the corresponding letter grade and print it out.
```
Refining the Grade Program

```python
numTests = int(input("How many tests did you take ?"))
total = 0
for thisTest in range(numTests) :
    ... ...
testAverage = total/numTests;
```

3. Find the corresponding letter grade and print it out.

```python
3.1 IF Average >= 90 THEN Grade = 'A'
3.2 ELSE IF Average >= 80 THEN Grade = 'B'
3.3 ELSE IF Average >= 70 THEN Grade = 'C'
3.4 ELSE IF Average >= 60 THEN Grade = 'D'
3.2 ELSE Grade = 'F'
```

```python
if testAverage >= 90 :
    courseGrade = 'A'
elif testAverage >= 80 :
    courseGrade = 'B'
elif testAverage >= 70 :
    courseGrade = 'C'
elif testAverage >= 60 :
    courseGrade = 'D'
else :
    courseGrade = 'F'
```
# Find out the number of classes
numTests = int(input
    ("How many tests did you take ?"))
total = 0
for thisTest in range(numTests) :
    thisGrade = int(input
        ("What grade did you get on this test ?"))
    # Add it to the total
    total = total + thisGrade

# Find the average
testAverage = total/numTests;

# Find the letter grade corresponding to the
# average
if testAverage >= 90 :
    courseGrade = 'A'
elif testAverage >= 80 :
    courseGrade = 'B'
elif testAverage >= 70 :
    courseGrade = 'C'
elif testAverage >= 60 :
    courseGrade = 'D'
else :
    courseGrade = 'F';

# Print the results.
print("Your test average is ", testAverage)
print("Your grade will be ", courseGrade)
One Last Refinement

- A test score must be between 0 and 100; therefore any grade greater than 100 or else than 0 is invalid.
- Let’s test each grade to make sure it’s valid.
- Also, we have to initialize the total to zero before we do any addition.

The Final Grade Program

```python
# Find out the number of classes
numTests = int(input
        ("How many tests did you take ?"))
total = 0
for thisTest in range(numTests) :
    thisGrade = int(input
            ("What grade did you get on this test ?"))
    # Make sure that the grades are valid
    # percentages
    if thisGrade > 100 :
        print("This is not a valid test grade.")
    elif thisGrade >= 0 :
        total = total + thisGrade
    else :
        print("This is not a valid test grade.")
```
total = total + thisGrade

# Find the average
testAverage = total/numTests

# Find the letter grade corresponding to the average
if testAverage >= 90 :
    courseGrade = 'A'
elif testAverage >= 80 :
    courseGrade = 'B'
elif testAverage >= 70 :
    courseGrade = 'C'
elif testAverage >= 60 :
    courseGrade = 'D'
else :
    courseGrade = 'F'

# Print the results.
print("Your test average is ", testAverage)
print("Your grade will be ") + courseGrade