# Augmented Assignment Operator

| | | | |
|---|---|---|---|
| **Notebook:** | Computers and Programming I | | |
| **Created:** | 10/25/2018 2:59 PM | **Updated:** | 10/25/2018 3:50 PM |
| **Author:** | Anonymous | | |

- Augmented Assignment Operator
    - In many assignment statement, the variable on the left side of the assignment operator (=) also appears on the right side of the operator
    - Augmented Assignment Operator:
        - special set of operators designed for this type of job
            - shorthand operators
            - total = total + number
            - total += number

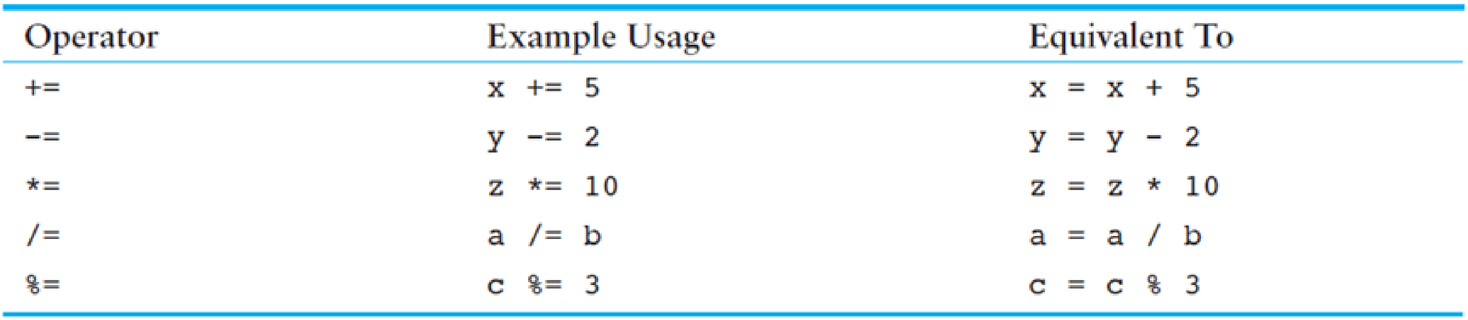


**Table 4-2** Augmented assignment operators

| Operator | Example Usage | Equivalent To |
|---|---|---|
| `+=` | `x += 5` | `x = x + 5` |
| `-=` | `y -= 2` | `y = y - 2` |
| `*=` | `z *= 10` | `z = z * 10` |
| `/=` | `a /= b` | `a = a / b` |
| `%=` | `c %= 3` | `c = c % 3` |

- Sentinel
    - Special value that marks the end of a sequence
        - When program reaches a sentinel, it knows that the end of the sequence of items was reached, and the loop terminates.
        - Must be distinctive enough so as not to be mistaken for a regular value in the sequence
        - Example: when reading an input file, empty line can be used as a sentinel
        - Ex: while (grade>= 0):
            - Grade must be greater than zero otherwise error message will occur
- Input Validation Loops
    - Computer cannot tell the difference between good data and bad data:
        - If the user provides bad input, program will produce bad output
        - GIGO - Garbage in, garbage out
        - It is important to design program such that bad input is never accepted
        - If input is invalid, prompt user to enter correct data
        - Commonly accomplished while using a while loop which repeats as long as the input is bad
            - If input is bad, display error message and receive another set of data
            - In input is good, contiue to process in information

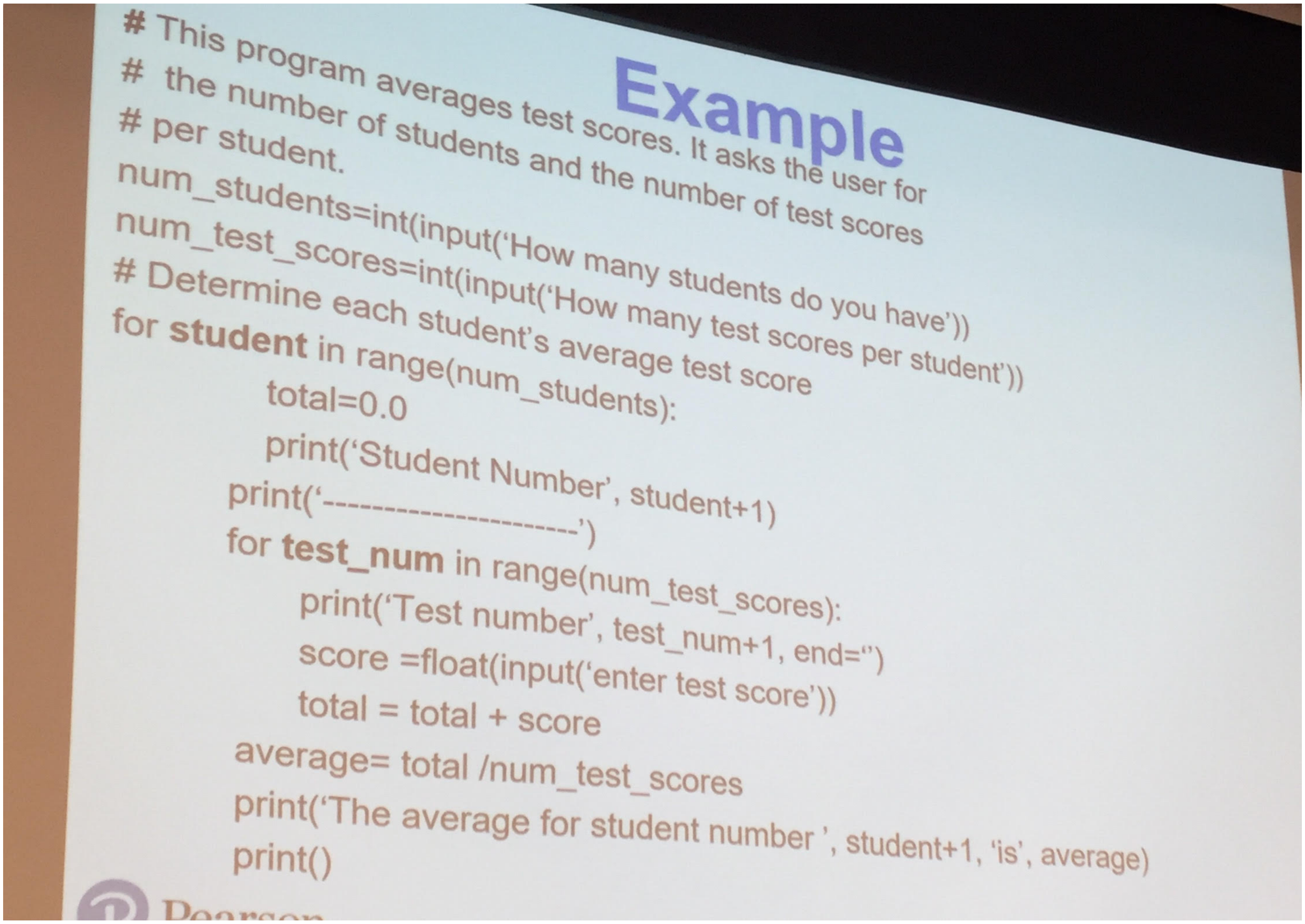
```
grade = int(input("Enter and grade:"))
while (grade < 0):
  print ("Error, enter a positive grade.")
   grade = int(input("Enter and grade:"))

print ("Yea!")
```

- Nested Loop
    - Loop that is contained inside another loop

```
for counter in range (1,11):
   grade = int(input("Enter and grade:"))
  while (grade < 0):
     print ("Error, enter a positive grade.")
      grade = int(input("Enter and grade:"))

print ("Yea!")
```



Example

```
# This program averages test scores. It asks the user for
#  the number of students and the number of test scores
# per student.
num_students=int(input(‘How many students do you have’))
num_test_scores=int(input(‘How many test scores per student’))
# Determine each student’s average test score
for student in range(num_students):
        total=0.0
        print(‘Student Number’, student+1)
      print(‘--------------------’)
      for test_num in range(num_test_scores):
          print(‘Test number’, test_num+1, end=‘’)
          score =float(input(‘enter test score’))
          total = total + score
      average= total /num_test_scores
      print(‘The average for student number ’, student+1, ‘is’, average)
      print()
```

- Key Points
    - Inner loop goes through all of it iterations for each iteration of outer loop
    - Inner loops complete their iterations faster that outer loops
    - Total Number of iterations in nested loop:
        - number_iterations_inner X number_iterations_outer
- Write a program to compute the sum of the following series:
    - 1/2 + 2/3 3/4 + ... + 50/51

```
total = 0.0

for number in range (1,51):
   total += (number)/(number + 1)

print (total)
```