Repeating actions

(1) Ask a person what they want until they say stop.

(2) Do repetitive arithmetic problems: Find the sum of numbers from $n$ to $m$ where $n$ and $m$ are chosen by the user.

(3) Print a table of data.

(4) Repeatedly check if an input is within proper bounds.
While statement

- `cont = True`
- print “Enter a series of positive numbers to add”
- print “Enter a 0 or a negative number to stop”
- `sum = 0`
- while `cont`:
  - `value = input(“Enter a positive number:”)`
  - if `value > 0`:
    - `sum = sum + value`
    - print “Current sum = ”, `sum`
  - else:
    - `cont = False`
Adding number in a sequence

- Write a program to ask the user for the starting number and final number of a sequence of consecutive integers, and print out the sum of those integers.

- Write a program to ask the user for the starting number, the number of numbers in the sequence of integers, and the spacing between the numbers, and print out the sum of those integers.
Printing a table

- def print_multiples(n):
  i = 1 #i and n are local variables
  while i <= 6:
    print n * i, '\t',
    i += 1
  print

i = 1
while i <= 8: #what is stack diagram?,
  print_multiples(i)
  i += 1
Tracing a program (ie playing computer)

- A initial stack diagram of the variables used in the many program
- As each statement is executed compute the value of each variable that is changed. If a print statement is executed write out what it prints. If a function is called, add its stack diagram and trace the action of the function. If a print statement is executed write out what it prints.
Tracing a program (ie playing computer)

- def print_multiples(n, high):
  i = 1
  while i <= high:
    print n*i, '	',
    i += 1
  print

def print_mult_table(high):
  i = 1
  while i <= high:
    print_multiples(i, high)
    i += 1
  print_mult_table(3) #trace
Analyzing Problems

• Write a program to ask the user to “Enter the number of rows in the triangle:” and then print out a triangle with that many rows. Example:

  Enter the number of rows in the triangle 5
  *
  **
  ***
  ****
  *****
  ******

• print_triangle1.py
Analyzing Problems

• Write a program to ask the user to “Enter the number of rows in the triangle:” and then print out a triangle with that many rows. Example:

Enter the number of rows in the triangle  5
  *
  * *
  * * *
  * * * *
  * * * * *
  * * * * *

• print_triangle2.py
The return statement

- `def check_skill(n):
  not_done=True
  prompt='enter a number divisible by '+str(n)+': '
  while not_done:
    number=input(prompt)
    if not(number%n):
      print 'correct, good job'
      return
    else:
      print 'incorrect: try again'

- In a boolean expression, 0, "," will act like False
- In a boolean expression any nonzero number, or nonempty string will act like True
The return statement

- def calc_sum_to(n):
  sum=0
  i=0
  while (i<n):
    while not_done:
      i=i+1
      sum=sum+i
    return sum
  return sum

print sum(10)
The break statement

- def check_skill(n):
  not_done=True
  prompt='enter a number divisible by '+str(n)+': '
  while not_done:
    number=input(prompt)
    if not(number%n):
      print 'correct, good job'
      break
    else print 'incorrect: try again'
The continue statement

• def calc_odds(n):
  sum=0
  i=0
  while (i<n):
    while not_done:
      i=i+1
      if (i % 2):
        continue
      sum=sum+i

  return sum

print sum(10)
Analyzing Problems

• Write a function `sum_of_squares_of_digits` that computes the sum of the squares of the digits of an integer passed to it. For example, `sum_of_squares_of_digits(987)` should return 194, since $9^2 + 8^2 + 7^2 = 81 + 64 + 49 = 194$.

• How do you find the digits in a number? 192: $192 \% 10 = 2$, $192/10 = 19$, $19 \% 10 = 9$, $19/10 = 1$, $1 \% 10 = 1$, $1/10 = 0$