

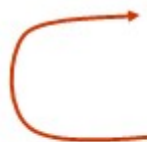
REPETITION CONTROL STRUCTURES

(May 15)

- so far learned 5 out of 6 key concepts:
input, processing, output, sequence, & selection
- repetition will enable expressing any computational algorithm
- many algorithms involve performing an action many times until a condition is met
- python provides: while and for
- by end of today, will learn how to use these statements in your program

The idea: go back up and repeate some code

- until now our programs were strictly sequential
- “if” enables skipping of some code but program still flows from top to bottom
- many situations require being able to go back up and repeat certain sections of code



```
print "Instructions to user..."
grade = input("Enter a numerical grade between 0 and 100: ")
if grade>100 or grade<0:
    print "Error: The grade must be between 0 and 100."
elif grade>=95:
    print "A+"
elif grade>=86:
    print "A"
...
```

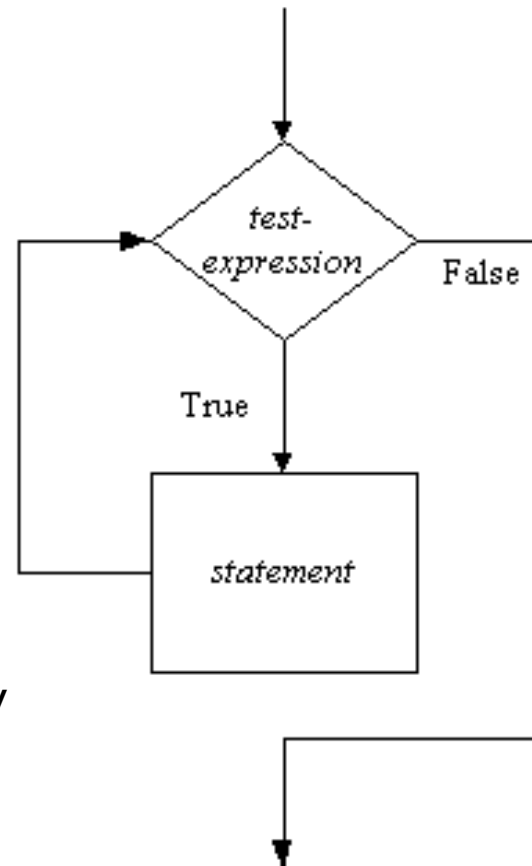
while loop:

Syntax

```
while test-expression:  
    statement
```

usage scenario:

repeat code as many times as necessary



Example: Five Hi!

```
# example: five hi  
counter = 5  
while counter > 0:  
    print("Hi!")  
    counter = counter - 1
```

```
.....  
Hi!  
Hi!  
Hi!  
Hi!  
Hi!
```

```
# example: five hi  
counter = 5  
while counter > 0:  
    print("Hi!")  
    counter -= 1
```

 augmented assignment

Beware of infinite loops!

```
# infinite loop  
while True:  
    print("Help! I'm stuck in a loop!")
```

```
Help! I'm stuck in a loop!  
Help! I'm stuck in a loop!  
Help! I'm stuck in a loop!  
Help! I'm stuck in a loop!  
Help! I'm stuck in a loop!  
Help! I'm stuck in a loop!  
Help! I'm stuck in a loop!
```

Input validation

```
validgrade = False #initialize
while not validgrade:
    print("please enter a valid grade between 0 and 100")
    grade = eval(input("Enter grade"))
    validgrade = grade >= 0 and grade <=100
```

```
....
please enter a valid grade between 0 and 100
Enter grade
-1
please enter a valid grade between 0 and 100
Enter grade
101
please enter a valid grade between 0 and 100
Enter grade
27
```

Input validation

```
validgrade = False #initialize
while not validgrade:
    print("please enter a valid grade between 0 and 100")
    grade = eval(input("Enter grade"))
    validgrade = grade >= 0 and grade <=100
```

```
# input validation
while True:
    print("please enter a valid grade between 0 and 100")
    grade = eval(input("Enter grade\n"))
    validgrade = grade >= 0 and grade <=100
    if validgrade:
        break
```

Repeating a program

```
# repeat a program
# here is a structure to enable user run a code again
# without having to restart the program
again='y'
while again=='y' or again=='Y':

    #
    # put the code you want to repeat here
    #

    again = input("Do it again (y/n)? ")

print("bye now")
```

Note: we're not doing "eval(input....)"

Repeating a program

Example: f2c

```
# f2c_while.py: converts a given temperature in Fahrenheit to Celsius
# CPSC 128 Example program
# S. Bulut, Spring 2018-19
```

```
again='y'
while again=='y' or again=='Y':

    print("This program converts temperatures from Fahrenheit to Celsius.")
    print("Enter a temperature in Fahrenheit (e.g. 10) and press Enter.")
    temp_in_f = eval(input("Temperature in Fahrenheit: "))
    temp_in_c = (temp_in_f - 32)*5/9
    print(temp_in_f, "degrees Fahrenheit =", temp_in_c, "degrees Celsius.")

    again = input("Do it again (y/n)? ")

print("bye now")
```

```
In [39]: runfile('/home/sbulut/github/cpsc128/code/python3/f2c_wl
code/python3')
```

```
This program converts temperatures from Fahrenheit to Celsius.
Enter a temperature in Fahrenheit (e.g. 10) and press Enter.
```

```
Temperature in Fahrenheit: 32
32 degrees Fahrenheit = 0.0 degrees Celsius.
```

```
Do it again (y/n)? y
Enter a temperature in Fahrenheit (e.g. 10) and press Enter.
```

```
Temperature in Fahrenheit: -40
-40 degrees Fahrenheit = -40.0 degrees Celsius.
```

```
Do it again (y/n)? n
bye now
```

```
In [40]:
```

Example: Find average

Write a program that can be used to find the average of a set of numbers

Example: Sentinel value controlled loop

Re-write find_average.py with a sentinel value controlled loop

Programing practice: Smallest and Largest values

Write a program that reads a sequence of integer values typed by the user and then displays the smallest and largest values entered. Use a sentinel value to stop the reading of the values, and allow the user to specify the value of the sentinel.

Programing practice: Smallest and Largest values

```
# sentinel.py
# Find the smallest and largest values in a sequence entered by the user.
#
# Tim Topper
# Fall 2008

print "This program will display the largest and smallest values you enter."
print
print "You have to choose a special number to indicate the end of your input."
sentinel = input("What number will indicate that you have entered all your values? ")
print

print "Begin entering your values now."
print "Press Enter after each one, and type your special number when you are done."
num = input("> ")
smallest = num
largest = num
while num != sentinel:
    if num < smallest:
        smallest = num
    elif num > largest:
        largest = num
    num = input("> ")

if smallest == largest == sentinel:
    print "You only entered the special number you chose."
else:
    print "The smallest value in your list is: %d" % (smallest)
    print "The largest value in your list is: %d" % (largest)
```

Problem: The guessing game

There is a guessing game children are fond of in which one player chooses a number between 1 and 100 and the other player must guess it. The only feedback given the guesser is whether their most recent guess was too high or too low. The game ends when the number has been guessed.

Write a program that 'plays' this game by picking a number and then accepting guesses about the number's value. After each guess it should inform the user if the guess was too high or too low. The game should end when the number is guessed, at which point the computer should display the number of guesses taken.

Hint: in order to get a random number between 0 and 100, do =>

```
In [59]: import random  
  
In [60]: print(random.randint(1,100))  
53  
  
In [61]: print(random.randint(1,100))  
79
```

Playing computer:

A puzzle: What output does this produce?

Step through code and keep track of values (on white board or paper).

```
# playing computer v1
x = 0
while x < 5:
    y = 0
    while y < 3:
        print("*")
        y=y+1
    print()
    x = x+1
```

Playing computer: (version 2)

A puzzle: What output does this produce?

```
# playing computer v2
x = 0
while x < 5:
    y = 0
    while y < x:
        print("*")
        y=y+1
    print()
    x = x+1
```


Nested loops

- we just saw example of nested loops

The syntax is:

```
# nested while
while test1:
    [statement]
    while test2:
        statement
```

The key point:

inner loop does **all** of its iterations for **each** of the outer loop iterations!!!