CIS 125 Principles of Programming Logic
Repetition (Looping) Structures

Basic Information

Repetition Structure:

A programming structure that causes code or a block of code to execute repeatedly.
Aka. Loop, iteration.

Loops are used extensively in programs, reports and web pages.

- When Amazon or Google return search results, a loop runs that creates an HTML table with the 10 rows of search results initially returned.
- When a program has a menu that uses a loop to continue to display the menu until the exit option is chosen.

Types of loops:

1. Count controlled loops / definite loops

   Repeats a specific number of times
   Usually uses a for loop
   Can be in used to loop through arrays/lists in many languages

2. Condition controlled loops / Indefinite loops

   Repeats until a specific condition is met, e.g. true/false, etc.
   Usually uses a while, do-while or a do-until loop

3. Infinite loops

   Loops that never end. Often an accident and undesired.

4. Nested loop

   A loop within loop.

5. Sentinal loop

   A sentinel value is a special value that signals when there are no more items from a list of items to be processed. This value cannot be mistaken as an item from the list.

   For example, if you want a loop that prompts the user to enter a patient's weight, or 0 if there are no more weights. Since a patient's weight would never be 0, this value can work as a sentinel value.

- First two types of loops above are the most common.

Two other ways to classify loops:

1. A **pretest loop** tests first and then if the test or condition is true, it executes the statements. Examples: for loop, while loop.

2. A **posttest loop** will execute the statements and then at the end, test the condition. Examples: Do … While loop.

Basic elements to most loops:

1. Initializing statement
2. Condition/tested expression
3. Altering statement

Loop Flowcharts

https://www.youtube.com/watch?v=1jA9mPPoXu0
https://www.youtube.com/watch?v=kBaAwEkE8A8

Python Programming Examples - Loops

Python supports three types of loops:

1. While loops
2. For loops
3. Nested loops

Python also has three loop control statements:

1. **break** statement - terminates loop and transfers execution to statement immediately following loop.

2. **continue** statement - causes loop to skip remainder of its body and retest condition prior to reiterating.

3. **pass** statement - used when statement is required syntactically but you do not want any command or code to execute.

For Loops

For loops are designed to run a specific number of times, for example 10.

Example: for loop (<u>one value – end</u>) with accumulator

In the example below, x is the loop counter and total is the accumulator. When you only have one value in the range() command the loop begins at zero. Loops in Python end one value before the number specified in range().

```
total = 0
for x in range(4):
        total = total + x
        print("x=", x, "Total=", total)
```

Output:

```
x= 0 Total= 0
x= 1 Total= 1
x= 2 Total= 3
x= 3 Total= 6
```

Example: for loop – with <u>two values</u>: beginning and end

When you place two numbers inside the range() command, you can specify the beginning and end of the loop. The loops ends one value before the end value.

```
for x in range(1, 5):
        print ("x=", x)
```

Output:

```
x= 1
x= 2
x= 3
x= 4
```

Example: for loop with three values: beginning, end, incrementor

Lastly, you can specify three values in the range(). The third value is an incrementor

```
for i in range(5,26,5):
        print (i)
```

Output:

```
5
10
15
20
25
```

Example: for loop with decrementer (counting backwards)

```
for i in range(3,0,-1):
        print (i)
```

Output:

```
3
2
1
```

Example: for loop with user input as max value

You can make a loop run as many times as the user specified. Be sure to get the user input as an integer.

```
max = int(input("Enter maximum number:"))
for i in range(1,max):
        print("i =", i)
```

Output:

```
i= 1
i= 2
i= 3
```

Example: continue command - Print odd numbers between 1 and 20

Note: **break** is used to exit a for loop or a while loop, whereas **continue** is used to skip the current block, and return to the "for" or "while" statement. A few examples:

```
# Prints odd numbers between 0 and 20
for x in range(1, 20):
    if x % 2 == 0:
        continue
    print (x)
```

Example: break statement

```
for x in range(1, 20):
    if x * 2 >= 10:
        break
    print (x)
```

Output:

1
2
3
4

Example: Using a loop to call a function

You can use a loop to call (run) a function many times.

```
def displayOutput():
    print("Welcome to this function")

for x in range(3):
    displayOutput()
```

Output:

Welcome to this function
Welcome to this function
Welcome to this function

Example: Loops in functions

Or, you can place the loop in the function. The program below produces the same output as the previous program.

```
def displayOutput():
    for x in range(3):
        print("Welcome to this function")

displayOutput()
```

Example: Loop in function: Raises over Five Years

Below is a more complex example of a loop within a function.

```
def calcRaise(salary, perc):
    total = 0
    print("Current salary is: ", salary)
    for x in range(1, 6):
        salary = salary * (1 + perc)
        print("Salary in the next year is $%.2f" % salary)
        total = total +  salary
    return total


salary = float(input("Enter current salary: "))
perc = float(input("Enter annual raise amount: "))
total = calcRaise(salary, perc)
print("You will make a total of $%.2f over the next five years" % total)
```

Output:

```
Enter current salary: 50000
Enter annual raise anount: .02
Current salary is:  50000.0
Salary in the next year is $51000.00
Salary in the next year is $52020.00
Salary in the next year is $53060.40
Salary in the next year is $54121.61
Salary in the next year is $55204.04
You will make a total of $265406.05 over the next five years
```

Example: Loop that calls functions and has decision (IF) statement in it

This is an example with a loop in the function with an IF statement inside the loop.

```
def displayOutput():
    num = int(input("Please enter a magic number: "))
    for x in range(1000):
        print(x)
        if x == num:
            print(num, "is the magic number.  End loop")
            break

displayOutput()
```

Output:

```
Please enter a magic number: 4
0
1
2
3
4
4 is the magic number.  End loop
```

Example: Loop that calls functions and has decision (IF) statement in it

This is another example with a loop in the function with an IF statement inside the loop.

```
def displayGrades():
    num_exams = int(input("Enter number of exams: "))
    for x in range(num_exams):
        score = float(input("Enter your exam score: "))
        if score >= 90:
            print("A")
        elif score >= 80:
            print("B")
        elif score >= 70:
            print("C")
        elif score >= 60:
            print("D")
        else:
            print("E")

displayGrades()
```

Output:

```
Enter number of exams: 2
Enter your exam score: 99
A
Enter your exam score: 22
E
```

While Loops – Sixteen Examples

Example: While loop that counts (count-controlled)

While loops can be count-controlled just like for loops (below). But this is not their best use. You can use a for loop for a count-controlled loop. While loops have their power in being condition-controlled. Nevertheless, here is an example of a count-controlled while loop.

```
i = 1
x = int(input("How high do you want to count to? "))
while (i <= x):
        print (i)
        i += 1
```

Output:

```
How high do you want to count:  5
1
2
3
4
5
```

Example: Decrementing loop

Here is another example of a count-controlled while loop – this one decrementing.

```
x = 3
while x >= 0:
   print(x)
   x-=1
```

Output:

```
3
2
1
0
```

Example: Loop that calls function based on user input

Here is one last example of a count-controlled while loop.

```
def counterLoop(x):
    total = x * 10
    print("Total: ", total)

i = 1
x = int(input("Please enter a number: "))
while i <= x:
    counterLoop(i)
    i+=1
```

Output:

```
Total: 10
Total: 20
Total: 30
Total: 40
```

Condition-Controlled While Loops

Condition-controlled while loops are very powerful but can take a bit more time to fully understand.

The while **True loop** is a powerful condition-controlled loop. This will run forever until a **break** command is used to end the loop. This example shows a loop that runs until the user enters the number -1 (i.e. a sentinel value).

```
while True:
    num = int(input("Enter a number (-1 to quit): "))
    if num == -1:
        break
    print("Total =", num * 10)
```

Output:

```
Enter a number (-1 to quit): 2
Total = 20
Enter a number (-1 to quit): 3
Total = 30
Enter a number (-1 to quit): -1
```

Example: While True loop to calculate Body Mass Index (BMI) with sentinel value to end loop

This is another example where a sentinel value such as -1 is used to end the loop.

```
while True:
    weight = float(input("How much do you weigh (-1 to quit)? "))
    if weight == -1:
        break
    height = float(input("How tall are you in inches? "))
    bmi = 703 * (weight / (height * height))
    print("Your BMI is: %.2f" % bmi)
```

Example: While loop with lists to end loop

This next examples, which are very useful, show two things:

a) Input validation – ensure the user types in correct value, for example Yes or No
b) A nested loop – a loop within a loop to assist with the input validation (the inner loops runs until valid input is typed in by the user)
c) Lists –yes_list and Python list of acceptable YES responses entered by the user

```
choice = "Y"
yes_list = ("Y","y","yes","Yes","YES")
all_list = ("Y","y","yes","Yes","YES", "N", "n", "no", "No", "NO")
while choice in yes_list:
    weight = float(input("\nHow much do you weight? "))
    height = float(input("How tall are you in inches? "))
    bmi = 703 * (weight / (height * height))
    print ("Your BMI is: %.2f" % bmi)
    choice = input("Would you like to make another BMI calculation (Y/N)? ")
    while choice not in all_list:
         choice = input("Invalid choice.  Enter a Y or N? ")
```

Output:

```
How much do you weight? 185
How tall are you in inches? 60
Your BMI is: 36.13
Would you like to make another BMI calculation (Y/N)? cat
Invalid choice.  Enter a Y or N? y

How much do you weight? 145
How tall are you in inches? 52
Your BMI is: 37.70
Would you like to make another BMI calculation (Y/N)? n
```

Example: While True menu loop – will not crash program if non-integer/invalid input entered

This next example is a slight variation of the **while True** loop where a **variable** is used to substitute for the True. In this case, **menu_chosen** is a variable created and set to True. The only way out of the loop is when menu option 3 is chosen and the variable is changed to the value **False**.

```
def menu():
    menu_chosen = True
    while menu_chosen:
        print("MAIN MENU")
        print("-----------------")
        print("1. Print pay check")
        print("2. Change benefits")
        print("3. Exit")
        choice = input("Chose menu option: ")
        if choice not in ("1", "2", "3"):
            print("Invalid menu choice.  Please re-enter")
        elif choice == "1":
            input("Change benefits function call will go here.\n")
        elif choice == "2":
            input("Change benefits function call will go here.\n")
        elif choice == "3":
            print("\n Goodbye")
            menu_chosen = False

# Main
choice = menu()
```

Output:

MAIN MENU
---
1. Print pay check
2. Change benefits
3. Exit
Chose menu option: 1
Change benefits function call will go here.

MAIN MENU
---
1. Print pay check
2. Change benefits
3. Exit
Chose menu option: cat
Invalid menu choice. Please re-enter

MAIN MENU
---

1. Print pay check
2. Change benefits
3. Exit
Chose menu option: 3

Goodbye

Example: While loop with list

Below is another example with lists, input validation and a nested while loop.

```
yes_list = ("Y","y","yes","Yes","YES")
all_list = ("Y","y","yes","Yes","YES", "N", "n", "No"", no", "NO")
while True:
    f = float(input("Please enter F temperature: "))
    c = (f - 32) * (5/9)
    print(f, "F temperature is equal to %.0f" % c, "temperature.")
    choice = input("Would you perform another conversion (y/n)? ")
    while choice not in all_list:
        choice = input("Invalid response. Would you perform another
conversion (y/n)?")
    if choice not in yes_list:
        break
```

Sample run:

Please enter F temperature: 32
32.0 F temperature is equal to 0 temperature.
Would you perform another conversion (y/n)? dddddd
Invalid response. Would you perform another conversion (y/n)?y
Please enter F temperature: 66
66.0 F temperature is equal to 19 temperature.
Would you perform another conversion (y/n)? n

Example: While True loop with nested while loop

This loop will run until the user enters a number that is divisible by 7.

```
while True:
   number = int(input("Please enter even number that is divisible by 7: "))
   while (number % 7) != 0:
       number = int(input("\nNumber must be divisible by 7.  Please re-enter:
"))
   print("\nCongratulations,", number, "divisible by 7")
   break
```

Output:

Please enter even number that is divisible by 7: 3

Number must be divisible by 7. Please re-enter: 4

Number must be divisible by 7. Please re-enter: 14

Congratulations, 14 divisible by 7

Example: While True loop with function

```
def calcAreaCircle(rad):
        area = 3.14159 * (rad ** 2)
        print("Area of circle is %.2f" % area)

while True:
        rad = int(input("Enter a radius (or 0 to exit): "))
        calcAreaCircle(rad)

        againYN = input("Perform another calculation (y/n? ")
        while againYN not in ["y","n","Y","N","Yes","No"]:
                againYN = input("Invalid choice.  Please enter y or n: ")
        if againYN in ["n","N","No"]:
                break
```

Output:

```
Enter a radius (or 0 to exit): 2
Area of circle is 12.57
Perform another calculation (y/n? pp
Invalid choice. Please enter y or n: y
Enter a radius (or 0 to exit): 3
Area of circle is 28.27
Perform another calculation (y/n? n
```

Example Number guessing game using while loop

```
import random
n = 20
guess = 0
num = random.randrange(1,20)
while guess != num:
        guess = int(input("Enter your guess (1-20): "))
        if guess > num:
                print("That's too large")
        elif guess < num:
                print("That's too small")
else:
        print("Congratulations. You got it correct.")
```

Output:

```
Enter your guess (1-20): 4
That's too small
Enter your guess (1-20): 9
That's too small
Enter your guess (1-20): 15
That's too large
Enter your guess (1-20): 12
Congratulations. You got it!
```