Slide Type Slide

1 ICS 104 - Introduction to Programming in Python and C

1.1 Loops - Lab 1

Slide Type Slide

2 Lab Objectives

- To implement while and for loop
- To become familiar with common loop algorithms

Slide Type Slide

3 Worked Example

Slide Type Fragment

- Problem Statement: Read twelve temperature values (one for each month), and display the number of the month with the highest temperature. For example, according to http://worldclimate.com, the average maximum temperatures for Death Valley are (in order by month, in degrees Celsius):

  > 18.2 22.6 26.4 31.1 36.6 42.2 45.7 44.5
  > 40.2 33.1 24.2 17.6

- In this case, the month with the highest temperature (45.7 degrees Celsius) is July, and the program should display 7.

Slide Type -

Slide Type Slide

- **Step 1:** Decide what work must be done inside the loop.
  - If you can't figure out what needs to go inside the loop, start by writing down the steps that you would take if you solved the problem by hand.

```
Read first value.
Read second value.
If second value is higher than the first, set highest temperature to that value, highest month to 2.
Read next value.
If value is higher than the first and second, set highest temperature to that value, highest month to 3.
Read next value.
If value is higher than the highest temperature seen so far, set highest temperature to that value,
    highest month to 4.
. . .
```

Slide Type Fragment

- Now look at these steps and reduce them to a set of uniform actions that can be placed into the loop body. The first action is easy:

```
Read next value.
```

Slide Type Slide

- The next action is trickier. In our description, we used tests "higher than the first", "higher than the first and second", and "higher than the highest temperature seen so far".
- We need to settle on one test that works for all iterations. The last formulation is the most general.
- Similarly, we must find a general way of setting the highest month. We need a variable that stores the current month, running from 1 to 12. Then we can formulate the second loop action:

```
If value is higher than the highest temperature, set highest temperature to that value,
    and set highest month to current month.
```

Slide Type Slide

- Now the loop becomes

```
Repeat
    Read next value.
    If value is higher than the highest temperature,
        set highest temperature to that value,
```

```
    set highest month to current month.
Increment current month.
```

Slide Type: Fragment

- **Step 2:** Specify the loop condition. What is the condition of the loop?

Slide Type: Fragment

- `current month <= 12`

Slide Type: Fragment

- **Step 3:** Determine the loop type. Which is more suitable, a `while` loop or a `for` loop?

Slide Type: Fragment

- A `for` loop is more suitable, since we know exactly how many iterations to perform.

Slide Type: Slide

- **Step 4:** Set up variables for entering the loop for the first time.

Slide Type: Fragment

```
current month
highest value
highest month
```

Slide Type: Fragment

- **Step 5:** Process the result after the loop has finished.

Slide Type: Fragment

- In our case, the result is the highest month.

Slide Type: Slide

- The complete pseudo code is as follows:

```
Read first value; store as highest value.
highest month = 1
For current month from 2 to 12
    Read next value.
    If value is higher than the highest value
        Set highest value to that value.
        Set highest month to current month.
```

Slide Type: Fragment

- **Step 6:** Trace the loop with typical examples.

current month	current value	highest month	highest value
		1	22.6
2	36.6	2	36.6
3	44.5	3	44.5
4	24.2		

- Note that there is no need to scratch the values out, as the textbook suggests, if you follow the convention that the last value of each column/variable in the table is the current value of that variable, you won't need to scratch anything out.

Slide Type: Slide

- **Step 7:** Translate the pseudo code into Python.

In [ ]:

Slide Type: Fragment

```
## This program reads 12 temperature values corresponding to the 12 months of the year (starting at 1),
#  and then prints the month with the highest temperature.
#
highestValue = float(input("Enter a value: "))  # Read the temperatrue for January
highestMonth = 1                                # This indicates January
for currentMonth in range(2, 13) :              # Iterate for months 2, 3, 4, ..., 12 (December)
    nextValue = float(input("Enter a value: ")) # Read the temperature of the next month.
    if nextValue > highestValue :               # Check whether it is higher than the highest
        highestValue = nextValue                # It is higher, so update the highest value
        highestMonth = currentMonth             # Correspondingly, update the highest month

# Print the highest month
print(str(highestMonth) + " was the highest month, with record temperature of "+ str(highestValue) + " degrees Celsius")

```

Slide Type: Slide

4 Side Note Regarding the `print` Function

- The `print` function displays an end of line by default.
- If we want to change this behavior, we can set the `end` parameter to another string.
  - The default value of the `end` parameter is \n.
- Consider the following example

In [1]:

Slide Type: Fragment

```
course = "ICS 104"
University = "KFUPM"
print(course, end = "@")
print(University)
print("KRK")
```

```
ICS 104@KFUPM
KRK
```

Slide Type: Slide

5 Exercises

Slide Type Fragment

- Exercise # 1: Write a while loop that prints all squares less than an input integer value **n**. For example, if the user enters 100, the program shall print

  > 0 1 4 9 16 25 36 49 64 81.

- The following are sample runs of the program.

Slide Type -

```
Enter a positive integer value: 34
Squares of numbers < 34 are:
0    1    4    9    16   25
```

Slide Type -

```
Enter a positive integer value: 150
Squares of numbers < 150 are:
0    1    4    9    16   25   36   49   64   81   100  121  144
```

In [2]:

Slide Type -

```
# Exercises # 1 - Source Code

inputNum1 = int(input("Enter a positive interger value: "))
numSq = 0
count = 0
print("Squares of numbers <",inputNum1,"are: ")
while numSq < inputNum1:
    print(numSq, end="\t")
    count = count + 1
    numSq = count*count
```

```
Enter a positive interger value: 150
Squares of numbers < 150 are:
0       1       4       9       16      25      36      49      64      81      100     121     144
```

Slide Type Slide

- Exercise # 2: Write a while loop that prints all positive numbers that are divisible by 10 and less than the user input value **n**. For example, if the user enters 100, the program shall print

  > 10 20 30 40 50 60 70 80 90.

- The following are sample runs of the program.

Slide Type -

```
Enter a positive integer value: 64
Positive numbers < 64 and divisible by 10 are:
10   20   30   40   50   60
```

Slide Type -

```
Enter a positive integer value: 9
There are no positive numbers < 9 divisible by 10 :
```

Slide Type -

```
Enter a positive integer value: -20
There are no positive numbers < -20 divisible by 10 :
```

In [40]:

Slide Type -

```
# Exercises # 2 - Source Code

inputNum2 = int(input("Enter a positive interger value: "))
if inputNum2 >= 10:
    numMultup = 10
    count = 1
    print("Positive numbers <",inputNum2,"and divisible by 10 are: ")
    while numMultup < inputNum2:
        print(numMultup, end="   ")
        count = count + 1
        numMultup = count*10

else:
    print("There are no positive numbers <",inputNum2,"divisble by 10")
```

```
Enter a positive interger value: 64
Positive numbers < 64 and divisible by 10 are:
10   20   30   40   50   60
```

Slide Type Slide

- Exercise # 3: Write a program that first asks the user to type today's price for one dollar in Japanese yen, then reads US dollar values and converts each to yen. Use 0 as a sentinel.
- The following are sample runs of the program.

Slide Type -

```
Enter today's price in yens for 1 $: 105.69
Enter amount in dollars to convert: (0 to quit) 200
200.00 dollars = 21138.00 yens
Enter amount in dollars to convert: (0 to quit) 500
500.00 dollars = 52845.00 yens
Enter amount in dollars to convert: (0 to quit) 100
100.00 dollars = 10569.00 yens
Enter amount in dollars to convert: (0 to quit) 0
```

Slide Type -

```
Enter today's price in yens for 1 $: 105.6
Enter amount in dollars to convert: (0 to quit) 0
```

In [41]:

Slide Type -

```
# Exercises # 3 - Source Code
yenForDoll = float(input("Enter today's price in yens for 1$: "))
inputDollar = float(input("Enter amount in Dollars to convert : (0 to quit) "))
while inputDollar!= 0:
    yens = inputDollar * yenForDoll
    print("%0.2f dollars = %0.2f yens" % (inputDollar,yens))
    inputDollar = float(input("Enter amount in Dollars to convert : (0 to quit) "))
```

```
Enter today's price in yens for 1$: 105.69
Enter amount in Dollars to convert : (0 to quit) 200
200.00 dollars = 21138.00 yens
Enter amount in Dollars to convert : (0 to quit) 500
500.00 dollars = 52845.00 yens
Enter amount in Dollars to convert : (0 to quit) 100
100.00 dollars = 10569.00 yens
Enter amount in Dollars to convert : (0 to quit) 0
```

Slide Type Slide

- Exercise # 4: Write a program that prompts for and reads the number n of spheres to be processed. If $n \leq 0$ your program must display an error message and terminate; otherwise it does the following for n times:
  - Prompts for and reads the volume of a sphere, it then displays the surface area of the sphere with that volume. Assume that each volume is in cubic centimeters.
- The program finally displays the average of the surface areas of the n spheres.
- Please note that
  - $\pi = 3.14159$.
  - volume $= \frac{4}{3}\pi r^3$.
  - surface area $= 4\pi r^2$.
- The following are sample runs of the program.

```
Enter number of spheres to be processed: 3
Enter volume of sphere 1: 15
Sphere #1 surface area= 29.41
Enter volume of sphere 2: 27
Sphere #2 surface area= 43.52
Enter volume of sphere 3: 9
Sphere #3 surface area= 20.92
The average surface area = 31.29
```

```
Enter number of spheres to be processed: -1
 number of spheres must be > 0
```

In [1]:

```
# Exercises # 4 - Source Code
import math
num = float(input("Enter number of spheres to be processed: "))
if num > 0 and num.is_integer():
    num = int(num)
    totalSurface = 0
    count = 0
    for x in range(0,num):
        x = str(x+1)
        volume = float(input("Enter volume of sphere "+x+": "))
        radius = ((3*volume)/(4*math.pi))**(1/3)
        surfaceArea = 4 * math.pi * radius**2
        print("Sphere #"+x+" surface area= %0.2f" % surfaceArea)
        totalSurface = totalSurface + surfaceArea
        count = count + 1
    averageSurface = totalSurface / count
    print("The average surface area = %0.2f"%averageSurface)
else:
    print("numbers of sphere must be > 0 integers")
```

```
Enter number of spheres to be processed: 3
Enter volume of sphere 1: 15
Sphere #1 surface area= 29.41
Enter volume of sphere 2: 27
Sphere #2 surface area= 43.52
Enter volume of sphere 3: 9
Sphere #3 surface area= 20.92
The average surface area = 31.29
```

In [ ]: