

# 1 ICS 104 - Introduction to Programming in Python and C

## 1.1 Loops - Lab 1

## 2 Lab Objectives

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

## 3 Worked Example

- **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> (<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.



© Stevegeer/iStockphoto.

- **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.**

**. . .**

- 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.**

- 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.**

- 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.**

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

- `current month <= 12`

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

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

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

# current month highest value highest month

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

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

- 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.**

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

| current month | current value   | highest month | highest value   |
|---------------|-----------------|---------------|-----------------|
|               |                 | <del>1</del>  | <del>22.6</del> |
| <del>2</del>  | <del>36.6</del> | <del>2</del>  | <del>36.6</del> |
| <del>3</del>  | <del>44.5</del> | 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.

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

```
In [21]: 1  ## This program reads 12 temperature values corresponding to the 12 months of the year (starting at 1),
2  # and then prints the month with the highest temperature.
3  #
4  highestValue = float(input("Enter a value: ")) # Read the temperature for January
5  highestMonth = 1 # This indicates January
6  for currentMonth in range(2, 13) : # Iterate for months 2, 3, 4, ..., 12 (December)
7      nextValue = float(input("Enter a value: ")) # Read the temperature of the next month.
8      if nextValue > highestValue : # Check whether it is higher than the highest
9          highestValue = nextValue # It is higher, so update the highest value
10         highestMonth = currentMonth # Correspondingly, update the highest month
11
12 # Print the highest month
13 print(str(highestMonth) + " was the highest month, with record temperature of " + str(highestValue) + " degrees Celsius")
14
```

Enter a value: 18.2

Enter a value: 22.6

Enter a value: 26.4

Enter a value: 31.3

Enter a value: 36.6

Enter a value: 42.2

Enter a value: 45.7

Enter a value: 44.5

Enter a value: 40.2

Enter a value: 33.1

Enter a value: 24.2

Enter a value: 17.6

7 was the highest month, with record temperature of 45.7 degrees Celsius

## 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 [22]: 1 course = "ICS 104"
          2 University = "KFUPM"
          3 print(course,end="@")
          4 print(University)
          5 print("First", "second", sep="- ")
```

```
ICS 104@KFUPM
First-second
```

## 5 Exercises

- [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.

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

```
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 [1]: 1 # YOUR CODE HERE
2
3 from math import *
4
5 num = int(input("Enter a positive integer value: "))
6 squares = 0
7
8 while squares < num:
9     if sqrt(squares) % 1 == 0:
10         print(squares//1, end=" ")
11         squares += 1
```

Enter a positive integer value: 34

0 1 4 9 16 25

## 5.1 Exercise # 2:

Write a loop that reads positive numbers from the user and sum them. The loop continues till a negative number is entered. Your program then prints the result. If the user enter a negative value from the beginning, your code should display 'No positive number was entered'.

- The following are sample runs of the program.

### Sample run 1:

Enter a positive number: (or a negative value to finish):5

Enter a positive number: (or a negative value to finish):2

Enter a positive number: (or a negative value to finish):

Enter a positive number: (or a negative value to finish):4

Enter a positive number: (or a negative value to finish):-5

Sum = 11.0

### Sample run 2:

Enter a positive number: (or a negative value to finish):-8

No positive number was entered.

Note that the 3rd input in sample run 1 is 'Enter' key which does not stop the loop. You should include the empty string in your loop condition.

In [2]:

```
1  # YOUR CODE HERE
2
3  total = 0
4
5  num = input("Enter a positive number: (or a negative value to finish):")
6  if num == "":
7      num = 0
8  else:
9      num = float(num)
10
11  if num < 0:
12      print("No positive number was entered.")
13
14  else:
15      while num >= 0:
16          total = total + num
17          num = input("Enter a positive number: (or a negative value to finish):")
18          if num == "":
19              num = 0
20          else:
21              num = float(num)
22      print("Sum =", total)
```

```
Enter a positive number: (or a negative value to finish):5
Enter a positive number: (or a negative value to finish):2
Enter a positive number: (or a negative value to finish):
Enter a positive number: (or a negative value to finish):4
Enter a positive number: (or a negative value to finish):-5
Sum = 11.0
```

## 5.2 Exercise # 3:

Using for loop, write a python program that reads an integer from the user and prints back the number of the digits in that integer and the sum of them. For example, if the input was 2308 , the output would be: Your integer has 4 digits and their sum is 13.

- Hint: take your input as a string and use for loop to get the digits.
- The following are sample runs of the program:

- *Sample run 1:*  
Enter a positive integer value: **2308**  
Your integer has 4 digits and their sum is **13**
- *Sample run 2:*  
Enter a positive integer value: **714702**  
Your integer has 6 digits and their sum is **21**

In [3]:

```

1  # YOUR CODE HERE
2
3  total = 0
4
5  integer = input("Enter a positive integer value: ")
6  digits = len(integer)
7
8  for i in integer:
9      i = int(i)
10     total += i
11
12 print("Your integer has", digits, "digits and their sum is", total)

```

Enter a positive integer value: 2308  
Your integer has 4 digits and their sum is 13

- **Exercise # 4:** Write a python 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
  - $\text{volume} = \frac{4}{3}\pi r^3$ .
  - $\text{surface area} = 4\pi r^2$ .
- The following are sample runs of the program.

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

```
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 [4]:

```
1  # YOUR CODE HERE
2
3  from math import pi
4
5  num = float(input("Enter number of spheres to be processed: "))
6
7  if num > 0 and num.is_integer():
8      num = int(num)
9      totalSurface = 0
10     count = 0
11
12     for x in range(0,num):
13         x = str(x+1)
14
15         volume = float(input("Enter volume of sphere "+x+": "))
16         radius = ((3 * volume) / (4 * pi)) ** (1/3)
17         surfaceArea = 4 * pi * (radius ** 2)
18         print("Sphere #" + x + " surface area= %0.2f" % surfaceArea)
19         totalSurface = totalSurface + surfaceArea
20         count = count + 1
21     averageSurface = totalSurface / count
22     print("The average surface area = %0.2f" % averageSurface)
23 else:
24     print("numbers of sphere must be > 0 ")
```

```
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
```

## 6 End of the Loops - Lab 1

Good luck...

