STMC HKOI Training

Lesson 5: Looping structure and arrays (I)

Chan Yan Mong

November 20, 2021

Goal today

- Concept of loop
- `while` loop
- `for` loop using `range`
- The for loop `for`
- Basics `list`

Loop: Repeat and repeat ....

- Many times in programming we want the code to run repeatedly until certain conditions are met
- For example:
  - Recieving user input: User might input a wrong value. You would want to keep asking for an input until it’s right
  - Reading files: You want to keep reading lines until the end of file
  - Games: You want to keep the main code running until the game ends
  - Searching: Sometimes you use computer to search for answers. You would want the computer to keep searching until the solution / close enough solution is reached

Loop: Repeat and repeat ….

- From the examples above, we see the a looping structure always consist of two parts:
  1. The code inside the code that is looped over
  2. A condition that is checked everytime the loop ran to decide whether the loop should continue
- Example:
  - Recieving user input (code inside loop); Is the answer right (terminate condition)
  - Reading files (code inside loop); Is the end of file reached (terminate condition)
  - Main game code (code inside the loop); Is the game over (terminate condition)
  - Searching for answers (code inside the loop); Is the solution found (terminate condition)

Example: Print first N positive integer

- Let's write a program that takes in an integer N and print out all positive integers i in range $1 \leq i \leq N$
- For example:
  - If we enter 1, `{1}` will be printed
  - If we enter 4, `{1,2,3,4}` will be printed
  - and etc.

Example: Print first *N* positive integer

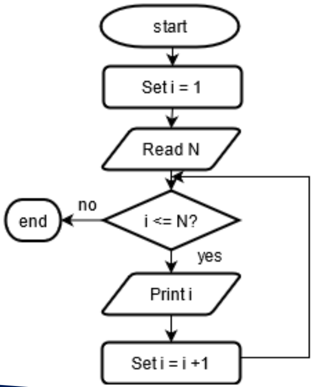
- Some example input and output:

```
$./main          $./main          $./main
5                4                100
1                1                1
2                2                2
3                3                .... /* too long won't list here*/
4                4                99
5                                 100
```

- **Problem: How can we implement this in code?**

Flow chart

- Let’s look at the flow chart
- Basically, we repeat certain blocks of code until a given condition (in this case $i \leq N$) is *false*
- This condition is called the **loop condition**
- Notice ”Set i = i + 1” is crucial otherwise i will always be smaller than N. This will cause an **infinite loop**



`while` loop

- In Python, we can implement that using `while` loop
- Here is the syntax of `while` loop

```python
while """loop condition""":
  # Remember to indent
  # This will keep looping as long as loop condition is True

# When loop condition is False, the loop will break
# The code will continue to run from here
```

Example: Print first *N* positive integer

- This is how we write print first *N* positive integer in python

```python
N = int(input('Enter N: '))
i = 1
while i <= N:
  print(i)   # Print i, remember to indent
  i = i + 1  # This is critical, otherwise infinite loop
print('End of story') # Just some useless print
```

Example: Sum of first n cubes

Write a program that takes n as an input and compute the sum of first n cubes S_n:

$$
\begin{aligned}
S_1 &= 1^3 \\
S_2 &= 1 + 2^3 \\
\cdots \\
S_n &= 1^3 + 2^3 + \cdots + (n-1)^3 + n^3
\end{aligned}
$$

Example: Sum of first *n* cubes

- This is similar to our previous example:

```python
N = int(input('Enter N: '))
i = 1 # Index to loop over
S = 0 # Storing the Sum

while i <= N:
  S = S + i**3 # New sum = Prev Sum + i^3

print('Result: ',S)
```

Example: Input validation

- You are writing a registration website for a company.
- In your website, the user is required to enter their age.
- However, some employee of the company might be careless and enter their age incorrectly.
- Write a program that reads in an age, and make sure it's between $18 - 65$ (inclusive)
- If the age is out of this range, prompt the user to renter the information until the input is correct

Example: Input validation

- Example input and ouput:
- Correct input:

```
Enter your age: 18
Ok! Have a nice day!
```

- Incorrect input:

```
Enter your age: 12
Age should be from 18-65
Enter your age: 69
Age should be from 18-65
Enter your age: 27
Ok! Have a nice day!
```

Example: Input validation

- One possible solution:

```python
""" Sample solution for Input validation """

age = int(input('Enter your age: '))
while age < 18 or age > 65:
  print('Age should be from 18-65')
  age = int(input('Enter your age: '))
print('Ok! Have a nice day!')
```

`while` loop

Exercise: Fibonnaci number

HKOI Online Judge (D201): `https://judge.hkoi.org/task/D201`

Exercise: Statistical analysis

HKOI Online Judge (J024): `https://judge.hkoi.org/task/J024`

Remarks: If you want to learn more about statistics and using python to analyze data, read the supplimentary materials: Simple Statistics with Python

`while` loop (Might need list)

Exercise: Stamps

HKOI Online Judge (01014): `https://judge.hkoi.org/task/01014`

Exercise: Bin packing

HKOI Online Judge (01050): `https://judge.hkoi.org/task/01050`

`for` loop

- In principle all loops can be written using `while` loop
- But sometimes we want to be more *concise*
- For example, the following loop is clumsy:

```
i = 0
while i < 5:
  print(i)
  i = i+1
```

`for` loop

- In fact, if we want to do loop similar to that above, we can use the `for` loop
- The equivalent for loop for the loop just now is:

```python
for i in range(0,5):
  print(i) # Print numbers 0, 1, 2, 3, 4
```

which looks much nicer

Example: Print first *N* integer

- Using for loop, our previous example of printing first *N* integers can be greatly simplified:

```python
""" Print first N integer using for loop """
N = int(input('Enter N: '))
for i in range(0,N):
  print(i+1)
```

General syntax of `for` loop

- In general, the syntax for a `for` loop using `range` is:

```python
for i in range(begin,end,steps):
  # Do things here
```

- This will loop `i` from `begin <= i < end` with `i` increasing by `step` each time it loops
- For example: `range(1,7,1)` will gives you $1, 2, 3, 4, 5, 6$ (notice the last number is excluded)
- Another example: `range(2,9,3)` will give you $2, 5, 8$ (notice each number differ by 3, the step size)

Example: Sum of first *n* odd numbers

- Write a program using `for` loop that calculate the sum of first *n* odd numbers

$$S = 1 + 3 + 5 + \cdots + 2n - 1$$

```python
""" Solution: Sum of first n odd numbers """
N = int(input('Enter N: '))
S = 0
for i in range(1,2*N,2): # Upper limit 2N to include 2N-1
  S += i
print('Sum: ',S)
```

Example: Magic triangles

Write a program that recieve an integer *n*. Print a triangle of height *n* and base *n* with using $(*)$. Here are some example outputs

```
>>3            >>5          >>2
*              *            *
**             **           **
***            ***
               ****
               *****
```

(Hint: To print a $*$ without newline, you can use `print('*',end='')`)

Example: Magic triangles+

Modify the program previously to give the following output:

```
>>3                >>5              >>2
*                  *                *
**                 **               **
***                ***              *
**                 ****
*                  *****
                   ****
                   ***
                   **
                   *
```

Example: Number of ways to queue up

Problem Statement:

Let there be n people. Write a program that computes the number of ways the people can form a queue

Solution

Let's look at the 1st, 2nd, 3rd, etc. positions of the queue one by one. For the first position, there are n ways to assign someone to a queue; for the second position, there are $n-1$ ways, because one people have been placed on the queue. As we progress, we saw that the total number of ways $W = n \times (n-1) \times \cdots 2 \times 1$ ways.

Example: Number of ways to queue up

Solution

Therefore, the required code is:

```python
n = int(input('Enter number of people: '))
W = 1
for i in range(1,n):
  W = W*i
print('Number of ways is:', W)
```

List: List of objects

- Loops are useful, but they are most powerful when used with data structures like `list`
- List is also called *array* in language like C/C++
- A list is an **ordered list of objects**
- It stores multiple values in a single variable, which we can refer to using an **index**

List: Example of Lists

- To create a list, we surround some *comma-separated* values with `[]`
- Let’s look at a list to see what exactly it means:

```
intList = [10,328,321,392] # List of integers

floatList = [40.1,339.2,77.3] # List of floats

strList = ['Billy', 'May', 'Dorian'] # List of strings

boolList = [True,False,True,Flase] # List of booleans

mixedList = [183.3, 282, False, 'Hi'] # List of mixed data types
```

List: Indexing

- Each item in a list is labelled by an **index**, which we can use to refer to an item
- The **indices starts from 0**

```python
myList = ['Hello',831.9, False, 88]

print('myList[0]: ', myList[0]) # myList[0] = 'Hello'

print('myList[1]: ', myList[1]) # myList[1] = 831.9

print('myList[2]: ', myList[2]) # myList[2] = False

print('myList[3]: ', myList[3]) # myList[3] = 88
```

List: Indexing

- For a list of length `n`, the indices ranges from `0,1,2,...,n-2,n-1`
- Accessing outside this length will results in:
  `IndexError: list index out of range`

```
>> myList = [28,219,3298]

>> myList[3] # Error! Indices from 0 to 2

>> myList[2] # Corret. Get 3298
```

List: Length of list

- The length of list can be obtained by using the `len()` function
- The returned value is an *integer*
- For example, to get the length of `myList` we write `len(myList)`

```python
myList = ['Hello',831.9, False, 88]

print('Length of list: ', len(myList)) # Length of list: 4
```

List: Add values to end

- We can add values to the *end* of the list by `append` method
- Syntax: `myList.append(<values>)`

```python
myList = [] # Empty list
print(myList) # Print []

myList.append(3) # Append 3 to list
print(myList) # Print [3]

myList.append('Hi') # Add 'Hi' to the end
print(myList) # Print [3, 'Hi']
```

List: Reading list of inputs

- Let's say we want to write a program that read in scores of students in a course and see how well they perform
- We can use list to do it

```python
studentScore = []
score = 0

while score >= 0: # Keep looping until input -1
  score = float(input('Enter score, enter -1 to terminate:'))
  if score >= 0:
    studentScore.append(score)
```

List: Loop over list

- After reading in data, we can loop the list over with for loop

```python
studentScore = [82,42,72,64,22]

# Print the items in the list
for i in range(0,len(studentScore)):
  print('Student ',i,'score ',studentScore[i])
```

List: Loop over list

- For example, find the largest in the list:

```python
studentScore = [82,42,72,64,22]
largest = studentScore[0]

for i in range(0,len(studentScore)):
  if studentScore[i] > largest:
    largest = studentScore[i] # If we find a score larger than
    largest, update largest score

print('Highest score: ',largest) # Print highest score
```

List: Loop over list

Exercise: Find minimum

Modify the code above to find the smallest in the list

Exercise: Average score

Write a program that takes scores until -1 is entered, then calculate and output the average score in the group

Exercise: Best student

Write a program that takes in the name and score in two list and output the name of the student with the highest score

List: Loop over list

Challenge: Sorting

Write a program that takes in a list of *N* numbers and return a sorted list of the numbers. We will come back to sorting in next slide. You may google for keywords like *bubble sort*, *insert sort* or *quicksort* for early exposure.

Application: Weather by month

- The Hong Kong Observatory (HKO) has historic weather data available on their website
- For instance, you can download daily average temperature from 1992 to now
- In this exercise, we will try to use this data to compile the monthly average temperature of Hong Kong and plot some nice graphs on Excel

Step 1: Data preparation

1. Browse HKO’s open data site (Click Me)
2. Click **Data on daily maximum, mean and minimum temperatures**
3. This will lead you to a page listing all the different weather stations that you can download data from
4. Find the data for **Shatin**
5. Click the URL to download the data set
6. Open the file in Excel and take a look

Step 2: Read data

1. To save you from the trouble of file io, a piece of helper code has been written
2. Download the helper code `csv_helper.py` from the course webpage
3. Copy the code and insert them at the **beginning** of your code

```python
def cast(cast_type,val,fallback_val = -1):
  ...
def format_line(line,sep=','):
  ...
def read_hko_csv(path):
  ...

# Your code starts here
```

Step 2: Read the data

1. Read the data by

```python
data = read_hko_csv('<path to your csv>')
```

2. Inspect the data by:

```python
print(data)
```

3. You can access the data as follows:

```python
data[0] # Get the first entry on csv (i.e. [1984,10,1,***,#])
data[0][0] # 1984
data[0][1] # 10
```

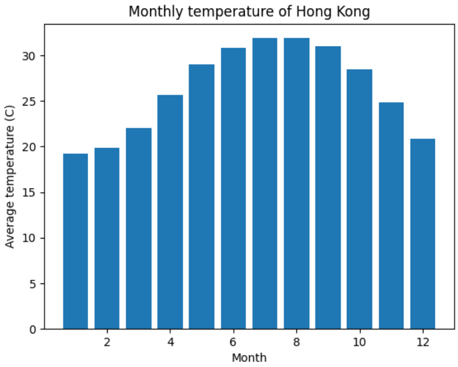
Step 3: Analysis the data

Now *you* are in charge. Try to do the following to obtain the monthly average temperature of Hong Kong:

1. Create two empty list with 12 0 called `temp` and `count`
2. Using a for loop, loop over the data set
3. For each entry, add the daily temperature to `temp` according to month and increase `count` of that month by 1. (Warning: On some date the data might be missing. Missing data are represented by -1)
4. After looping, divide each entry in the list by the number of valid entries. This will give you the average. Print the result and copy them to Excel

Step 4: Visualize

- You can now visualize the data on Excel. This should give a graph similar to this


Monthly temperature of Hong Kong
Average temperature (C)
Month