COMP1021
Introduction to Computer Science

Using For Loops

Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Use the range command to make a range of numbers
  2. Write loops using the for command

Writing Loops Using For

- Previously, we have discussed the use of while loops to do things repeatedly in Python
- In this presentation, we will look at another way of doing loops, using *for loops*
- Using a for loop:
  - you can perform some actions a particular number of times, or:
  - you can loop through a set of data, performing the same actions on every item in the set

The Range Command

- Before explaining how to use for loops let's look at a command called *range*
- The range command creates a range of numbers
- For example, you can make a range of numbers between 1 and 5 using the following code:

```
range(1, 6)
```

- The above line of code returns a range of numbers which includes 1, 2, 3, 4 and 5

Showing a Range

- If you print the result of a range, you will only see this, which is not very helpful:

```
>>> print(range(1, 6))
range(1, 6)
>>>
```

- To see the numbers returned by the range command, you will need to use `list()`, like this:

```
>>> print(list(range(1, 6)))
[1, 2, 3, 4, 5]
>>>
```


The Ending Number

- If you look carefully at the line of code shown in the previous slides, i.e.

```
range(1, 6)
```

  you may think that it should return the number 6 as well because the range is from 1 to 6
- However, the range command **does not** give you the ending number
- In this example, the number 6 is not included

The Default Starting Number is 0

- If you do not provide the starting number, the default value used by the range command will be 0 (not 1)
- For example, you can create a range of numbers from 0 to 4 by using this line of code:

```
range(5)
```

- This code generates 0, 1, 2, 3 and 4 (again, the ending number 5 is not included by `range`)

The Step Value

- If you want, you can provide an optional third input value in the range command
- This value is called the *step* value
- You can use it to skip (jump over) numbers
- For example, a step value of 2 will skip every other number within the range, like this:

`range(1, 10, 2)` returns 1, 3, 5, 7 and 9
`range(2, 10, 2)` returns 2, 4, 6 and 8

Note that 10, the ending number, is not included in the resulting range

More Examples of the Range Command

- Here are more examples of using the step value:

`range(0, 10, 3)` returns 0, 3, 6 and 9
`range(-1,-10,-2)` returns -1, -3, -5, -7 and -9

- Here are some unusual examples of using `range()`:

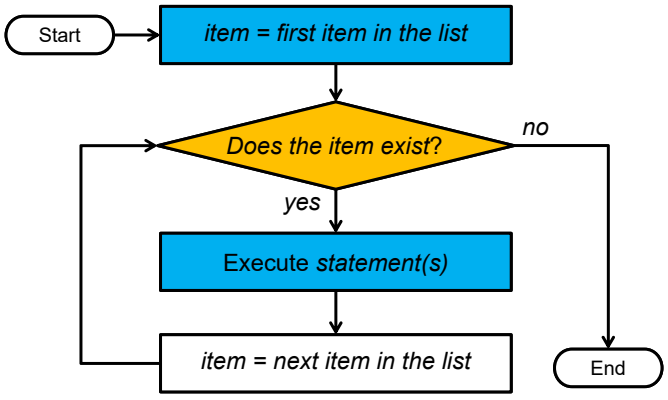
`range(10, 1)`
returns nothing because the default step value is 1
`range(-10, 1,-1)`
returns nothing because the step value is -1
`range(0, 10, 0)`
results in an error because the step value must not be zero

For Loops

`for` *item* `in` *a list of items* `:`
. . .statement(s). . .

- The *statement(s)* are executed for each item in a given list of items
- For example, if there are 10 items in the given list, the *statement(s)* will be executed 10 times

The Flow of a For Loop



Using Range in a For Loop

- You need to give a list of items to a for loop
- This is where the range command is commonly used
- For example, the following code prints out a list of four numbers:

```
for i in range(4) :
    print(i, end=" " )
```

The range command returns four items: 0, 1, 2 and 3

The end value means a space is put at the end of the number when it is printed, instead of moving to the next line

```
>>> for i in range(4):
        print(i, end=" ")


0 1 2 3
>>>
```

Controlling a For Loop

- As you can see from the previous slide the range command can be used to control:
  - how many times the content of a for loop is repeatedly executed
  - the number each time the loop content is executed
- Here are some more examples:

```
for i in range(0, 6):
    print(i, end=" ")
```

```
0 1 2 3 4 5
```

```
for i in range(1, 6, 2):
    print(i, end=" ")
```

```
1 3 5
```

Printing Things Using 'end'

- When you print things using the print command it moves to the next line after it finishes printing
- In the previous examples, we use the 'end' value to ask the print command to print a space at the end instead
- The end value is useful when you have multiple print commands that print things on the same line, e.g:

```
print("These are ", end="")
print("shown on ", end="")
print("the same line!")
```

In this example, nothing, i.e. "", is printed at the end of the text (there's no space in "")

```
>>>
These are shown on the same line!
>>>
```

Using a 'Fixed' List in a For Loop

- You can use a 'fixed' list instead of a list of numbers given by the range command
- For example, you can use any numbers you like:

```
for i in [33, 19, 5, -7]:
    print(i, end=" ")
```

```
33 19 5 -7
```

You use a pair of brackets, i.e. [], to enclose the list of items

- Or, you can choose not to use any numbers at all:

```
for word in ["How", "are", "you"]:
    print(word, end=" ")
```

```
How are you
```