COMP1021
Introduction to Computer Science

More on Loops

Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Use the continue command and the break command to stop a loop
  2. Explain the difference between using the continue command and the break command

Stopping a Loop

- There are two commands you can use to stop a loop
- The continue command:
  - stops the ***current*** execution of the loop
- The break command:
  - stops the ***whole*** execution of the loop
  - After running the break command, the program moves on to the rest of the code under the loop

Example of Using Continue

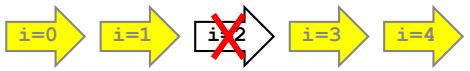
- Let's say we have a *for* loop that repeats the loop content 5 times, as illustrated below:

```
for i in range(5):
    print(i, end="")
```


```
01234
>>>
```

- If we run `continue` the third time the loop is executed i.e. i = 2, the execution will look like this:

```
for i in range(5):
    if i == 2:
        continue
    print(i, end="")
```




```
0134
>>>
```


Example of Using Break

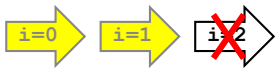
- Again, we have a *for* loop that repeats the loop content 5 times, as illustrated below:

```
for i in range(5):
    print(i, end="")
```


```
01234
>>>
```

- If we run `break` the third time that the loop is executed i.e. i = 2, the execution will look like this:

```
for i in range(5):
    if i == 2:
        break
    print(i, end="")
```




```
01
>>>
```


Continue vs Break

- Let's compare the continue command and the break command using another set of examples

Using continue

```
for i in range(10):
    if i % 2:
        continue
    print(i)
print("done!")
```

Output:

```
0
2
4
6
8
done!
>>>
```

Using break

```
for i in range(10):
    if i % 2:
        break
    print(i)
print("done!")
```

Output:

```
0
done!
>>>
```

A Quick Reminder

```
for i in range(10):
    if i % 2:
        continue
    print(i)
print("done!")
```

```
for i in range(10):
    if i % 2:
        break
    print(i)
print("done!")
```

`if i % 2:`

has the same meaning as:

`if i % 2` *has a result which is not zero* :

Illustration of What Happens in the Program Using 'continue' 1/3

- Remember `range(10)` generates 0, 1, 2, 3, … 9
- In the example using `continue` :

– when `i` = 0, `i%2` is false so `print(i)` is executed and the loop continues with the next number

```
if 0 % 2:  (false)
    continue ✗
print(0) ✓
```

```
0
```

– when `i` = 1, `i%2` is true so `continue` is executed and the loop immediately continues with the next number (`print` is not executed)

```
if 1 % 2:  (true)
    continue ✓
print(1) ✗
```

```
0
```


Illustration of What Happens in the Program Using 'continue' 2/3

– when `i` = 2, `i%2` is false so `print(i)` is executed and the loop continues with the next number

```
if 2 % 2:  (false)
    continue ✗
print(2) ✓
```

```
0
2
```

– when `i` = 3, `i%2` is true so `continue` is executed and the loop immediately continues with the next number (`print` is not executed)

```
if 3 % 2:  (true)
    continue ✓
print(3) ✗
```

```
0
2
```

– when `i` = 4, `i%2` is false so `print(i)` is executed and the loop continues with the next number

```
if 4 % 2:  (false)
    continue ✗
print(4) ✓
```

```
0
2
4
```

Illustration of What Happens in the Program Using 'continue' 3/3

- when `i` = 9, `i%2` is true so `continue` is executed and the loop stops immediately because there is no number left (`print` is not executed)
- Finally, the `print` statement after the for loop is executed

```
if 9 % 2: (true)
    continue ✓
print(9) ✗

print("done!")
```

```
0
2
4
6
8
done!
```

Illustration of What Happens in the Program Using 'break'

- In the example using `break` :
  - when `i` = 0, `i%2` is false so `print(i)` is executed and the loop continues with the next number
  - when `i` = 1, `i%2` is true so `break` is executed and the loop immediately stops (`print` is not executed)
  - Finally, the `print` statement after the for loop is executed

```
if 0 % 2: (false)
    break ✗
print(0) ✓

if 1 % 2: (true)
    break ✓
print(1) ✗

print("done!")
```

```
0

0

0
done!
```

You can use `break` for *for* and *while* loops

```
for i in range(5):
    if i == 2:
        break
    print(i)
```

```
0
1
```

```
i = 0
while i < 5:
    if i == 2:
        break
    print(i)
    i = i + 1
```

```
0
1
```

You can use `continue` for *for* and *while* loops

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

```
0
1
3
4
```

```
i = 0
while i < 5:
    if i == 2:
        i = i + 1
        continue
    print(i)
    i = i + 1
```

```
0
1
3
4
```

`break` and `continue` apply to the loop they are in

- For example, if you have a nested loop, those commands only apply to the loop in which they are used

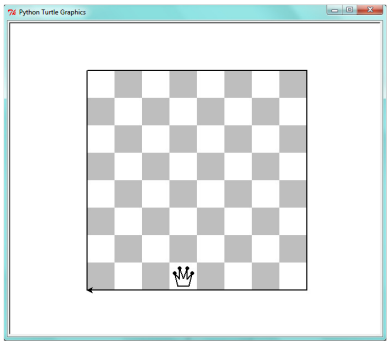
```
for year in range(1, 5):
    money=8000+(year*1000)
    for month in range(1, 13):
        print("Year", year, "month", month, end=": ")
        money=money-1000
        if money==0:
            print("No money left, must go home")
            break
        print("I have enough to stay away from home")
```

- In this example a student doesn't want to go home and is calculating how long he can stay away each year

```
Year 1 month 1: I have enough to stay away from home
. . .
Year 1 month 8: I have enough to stay away from home
Year 1 month 9: No money left, must go home
Year 2 month 1: I have enough to stay away from home
. . .
Year 2 month 9: I have enough to stay away from home
Year 2 month 10: No money left, must go home
Year 3 month 1: I have enough to stay away from home
. . .
Year 3 month 10: I have enough to stay away from home
Year 3 month 11: No money left, must go home
Year 4 month 1: I have enough to stay away from home
. . .
Year 4 month 11: I have enough to stay away from home
Year 4 month 12: No money left, must go home
```

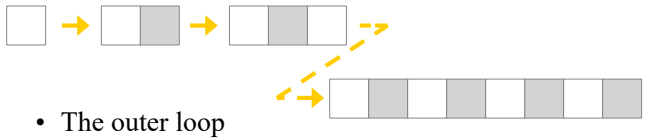
Drawing a Chessboard

- The next example uses a nested loop with the continue command to draw a chessboard:

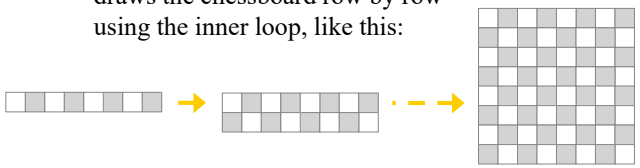


Inner Loop and Outer Loop

- The inner loop
  - draws a single row box by box, like this:

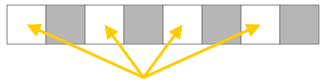


- The outer loop
  - draws the chessboard row by row using the inner loop, like this:



The Inner Loop – Drawing a Row

- A white box or a gray box is shown in the chessboard depending on the row number and the column number of the box
- No drawing is required for a white box because the background is already white, so we can use `continue` to skip the drawing for a white box



Designing the Code

- Let's draw the chessboard using two loops
  - Move to the top-left hand corner of the chess board
  - **Outer loop:** repeat eight times for drawing eight rows
    - **Inner loop:** repeat eight times for the eight boxes
      - If the current box is a white box, move to the next box position (no drawing occurs) and stop the current loop
      - Draw a gray box
      - Move to the next box position
  - Move to the position of the next row
- We will show the code in the following slides

Drawing the Chessboard – The Code 1/2

```
turtle.up()
turtle.goto(-200, 200)
turtle.down()
```

Move to the top-left hand corner of the chessboard

```
for row in range(8):
    for col in range(8):
        if col % 2 == row % 2:
            turtle.forward(50)
            continue
```

If both row and column are odd numbers, or both are even numbers, move to the next box (i.e. leave this part white) and stop the current loop

... the inner loop is continued on the next slide ...

Drawing the Chessboard – The Code 2/2

... this is the inner loop continued from the previous slide ...

```
turtle.begin_fill()
for _ in range(4):
    turtle.forward(50)
    turtle.right(90)
turtle.end_fill()
```

Draw a gray box

```
turtle.forward(50)
```

Move to the next box position

```
turtle.backward(400)
turtle.right(90)
turtle.forward(50)
turtle.left(90)
```

Move to the position of the next row

Finishing the Chessboard

- We could add a border around the chessboard and a queen chess piece to make a nice final image, like this:
- The code to draw the black border and the chess piece is not shown in this presentation