COMP1021
Introduction to Computer Science

Loops

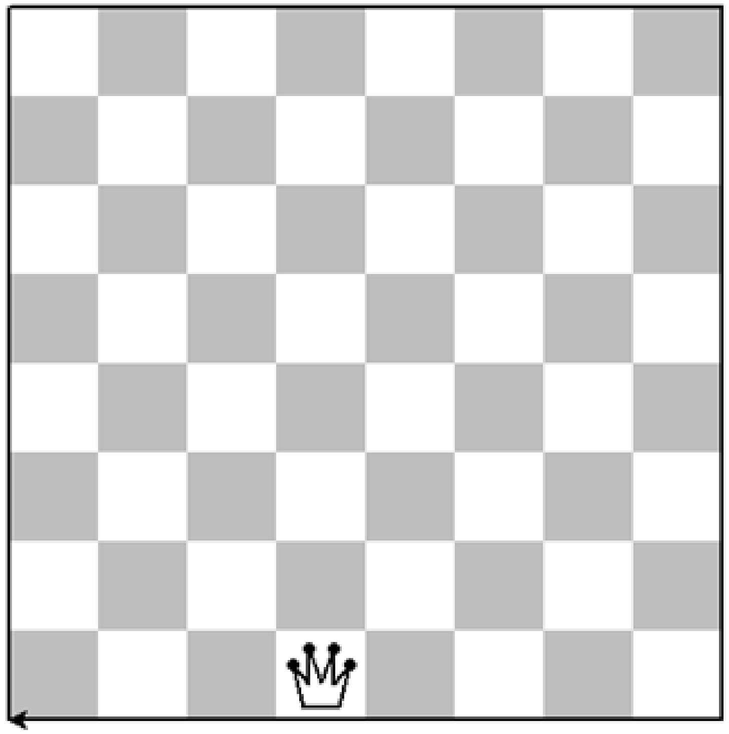
Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Write loops using the while command
  2. Work with conditions using logical operators
  3. Write code using nested loops

Loops

- You can write loops in Python to do things repeatedly
- Looping is a very useful feature because it makes repetitive work easier
- For example, you can use loops to generate a chess board
- In this presentation we look at *while loops* which are a common way to do looping in Python

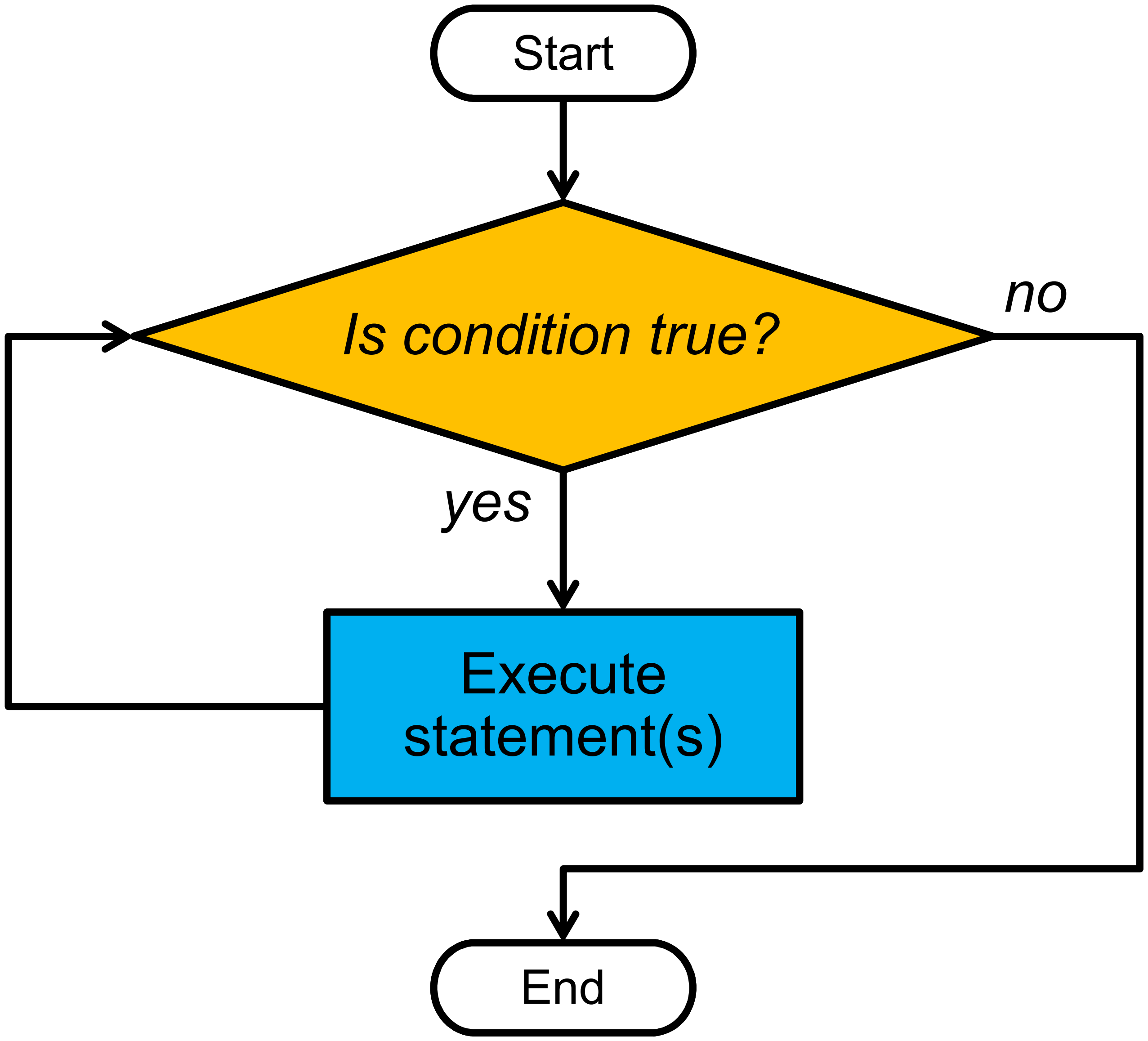


While Loops

```
while ...condition... :
    ...statement(s)...
```

- While *condition* is true, repeatedly execute *statement(s)*
  - A statement is a Python instruction
- When *condition* is false, the while loop finishes

The Flow of a While Loop



The First While Loop Example

- The following example keeps asking a yes/no question until the response is 'y'

```
print("This is a great course!")

response = ''

while response != 'y' :
    response = input('Do you agree? (y/n) ')
```

You need the : *(colon) here*

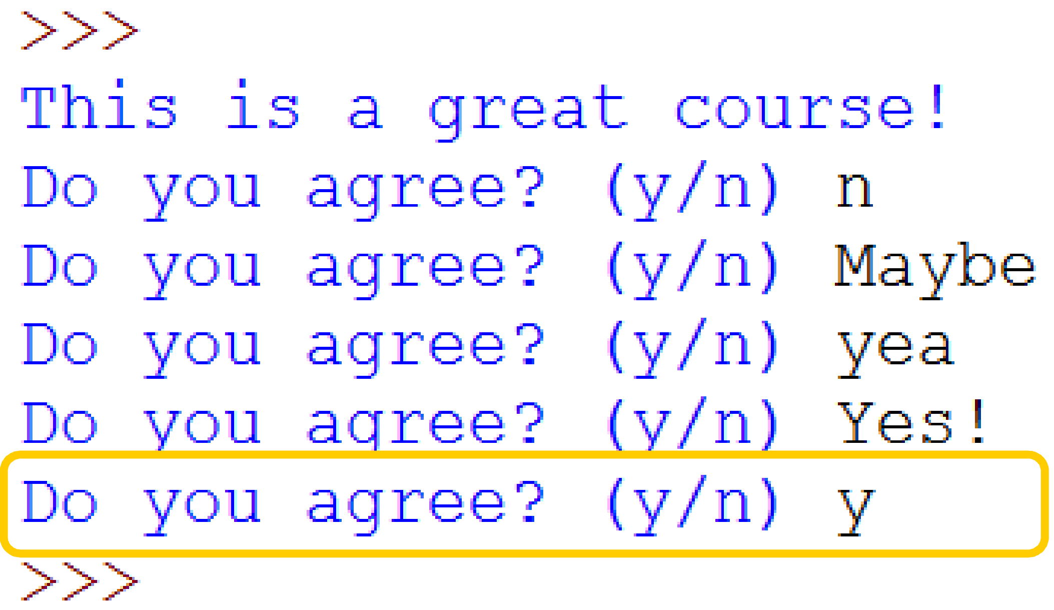
Code inside the loop must be indented

The user must respond with a single letter, 'y'

Running the First Example

- For example, here is what you see when you run the example:

The program stops only when the answer is 'y'



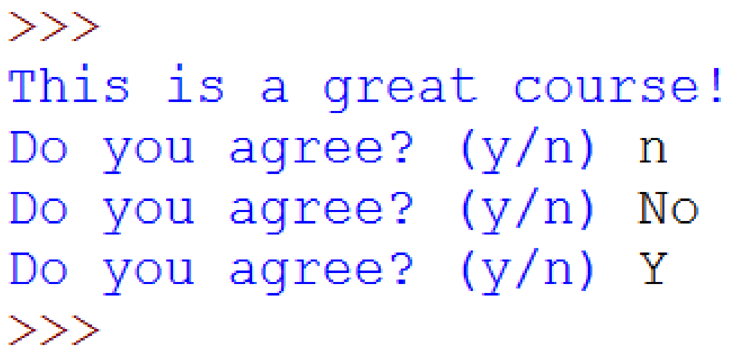
```
>>>
This is a great course!
Do you agree? (y/n) n
Do you agree? (y/n) Maybe
Do you agree? (y/n) yea
Do you agree? (y/n) Yes!
Do you agree? (y/n) y
>>>
```

Using Logical Operators

- You use the comparison operators (<, <=, >, >=, == and `!=`) to compare two values in one condition
- You can combine two conditions or reverse the value of a condition using *logical operators*:

a `and` *b*	if both condition *a* and condition *b* are true, the result is true; otherwise, it's false
a `or` *b*	if either condition *a* or condition *b* is true, the result is true; otherwise, it's false
`not` *a*	if *a* is true, then the result is false; and vice versa

Extending the First Example



```
>>>
This is a great course!
Do you agree? (y/n) n
Do you agree? (y/n) No
Do you agree? (y/n) Y
>>>
```

- Let's extend our first example using `and`

```
print("This is a great course!")

response = ''

while response != 'y' and \
      response != 'Y' :
    response = input('Do you agree? (y/n) ')
```

You can use '\' to break a single line of code into multiple lines

The user must respond with 'y' or 'Y'

An Eating Candy Example

- The program shown below uses a while loop to repeatedly buy candy bars until there is not enough money to buy more

Money in the pocket initially

The loop runs while there is enough money to buy a candy bar

```
money_in_pocket = 30
cost_of_candy_bar = 7

while money_in_pocket >= cost_of_candy_bar:
    print("I have $", money_in_pocket)
    print("I am buying and eating a delicious candy bar!")

    money_in_pocket = money_in_pocket - cost_of_candy_bar

print("Now, I only have $", money_in_pocket, "left.")
print("I don't have enough money for any more candy :(")
```

Running the Eating Candy Example

- Here is the result of running the program:

```
>>>
I have $ 30
I am buying and eating a delicious candy bar!
I have $ 23
I am buying and eating a delicious candy bar!
I have $ 16
I am buying and eating a delicious candy bar!
I have $ 9
I am buying and eating a delicious candy bar!
Now, I only have $ 2 left.
I don't have enough money for any more candy :(
>>>
```

In this example, $7 has been used to buy one candy bar each time inside the while loop

An Improved Candy Example

- Let’s improve the eating candy example to include the number of candy bars that are bought
- First, a variable to count the number of candy bars is added at the top of the program, like this:

```
candy_bars_eaten = 0
```

- Then inside the while loop, the variable is increased by one, like this:

```
candy_bars_eaten = candy_bars_eaten + 1
```

```
>>>
I have $ 30
I am buying and eating a delicious candy bar!
I have $ 23
I am buying and eating a delicious candy bar!
I have $ 16
I am buying and eating a delicious candy bar!
I have $ 9
I am buying and eating a delicious candy bar!
I have eaten 4 candy bars.
Now, I only have $ 2 left.
I don't have enough money for any more candy :(
>>>
```

Running the Improved Example

- Here is the program:

```
money_in_pocket = 30
cost_of_candy_bar = 7
candy_bars_eaten = 0

while money_in_pocket >= cost_of_candy_bar:
    print("I have $", money_in_pocket)
    print("I am buying and eating a delicious candy bar!")
    money_in_pocket = money_in_pocket - cost_of_candy_bar
    candy_bars_eaten = candy_bars_eaten + 1

print("I have eaten", candy_bars_eaten, "candy bars.")
print("Now, I only have $", money_in_pocket, "left.")
print("I don't have enough money for any more candy :(")
```

These are newly added code

A Math Question Example

- Here is an example which shows a math question to the user:

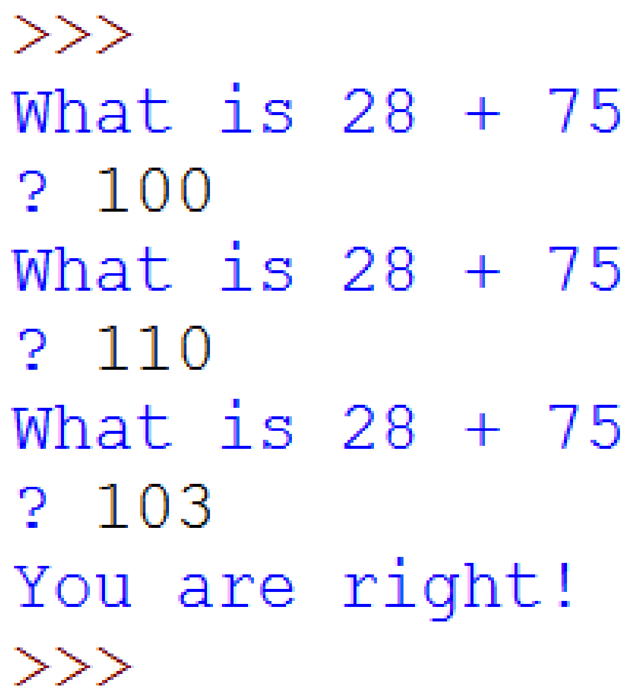
```
import random

number1 = random.randint(1, 99)
number2 = random.randint(1, 99)

answer = number1 + number2
guess = 0

while guess != answer:
    print("What is", number1, "+", number2)
    guess = input("? ")
    guess = int(guess)

print("You are right!")
```

Generate two random numbers between 1 and 99

The user guesses the answer inside the while loop

Running the Math Question Example

- To finish the program the user has to guess the correct answer
- This is because the while loop does not stop when `guess` is not equal to `answer`
- In other words, `guess` must be equal to `answer` to finish the game
- Here is an example of running the program:



```
>>>
What is 28 + 75
? 100
What is 28 + 75
? 110
What is 28 + 75
? 103
You are right!
>>>
```

Writing Comments

- Python will ignore anything on the right of #
- So you can use it to make notes, like this:

```
# This is an example of a loop
# It will count from 1 to 10
count=1 # Start with the number 1
while count<=10:
    print(count) # Show the number
    count=count+1 # Increase the variable
```

Another Way to Do Comments

- When you want to write a big comment, you can use `"""` *comments* `"""` , instead of starting every line of your comment with a #

```
"""
In this example the user has to enter the
result of adding two numbers.
We use a while loop to repeatedly ask for
the answer until the user gets it correct.
"""
```

  - (However, sometimes Python gets a bit confused when you use this method, the # method is safer)

Nested Loops

- A *nested loop* is a loop within a loop

```
start outer loop
    start inner loop
            ...statement(s)...
    end inner loop
end outer loop
```

- For example, you can put a while loop inside another while loop

Using an Infinite While Loop

- The previous math question program asks the question only once
- We will change the program so that it asks the math questions indefinitely
- We do this by using an *infinite loop* to enclose the previous loop
- An infinite loop is a loop that never stops, i.e. the condition is always true, like this:

```
while True :
```

...statement(s)...

A Nested Loops Example

This code asks random math questions indefinitely because the loop never stops

```
import random

while True:
    number1 = random.randint(1, 99)
    number2 = random.randint(1, 99)

    answer = number1 + number2
    guess = 0

    while guess != answer:
        print("What is",number1,"+",number2)
        guess = input("? ")
        guess = int(guess)

    print("You are right!")
```

Running the Nested Loops Example

- The program will not stop asking you math questions (because of the infinite loop!)
- One way to stop the program is by pressing *Control-C*, like this:

```
>>>
What is 78 + 50
? 128
You are right!
What is 55 + 42
? 97
You are right!
What is 8 + 97
? 105
You are right!
What is 19 + 77
?
Traceback (most recent call last):
  File "C:\06_while_loop_math_question_repeat_indefinite.py", line 21, in <modul
e>
    guess = input("? ") # Get the user input and store it
KeyboardInterrupt
>>>
```

Instead of answering the question, the user pressed *Control-C* here

Improving the Example

- It is not very nice when the user has to use *Control-C* to stop a program
- Let's put a proper condition in the outer loop of the example
- We will only ask three different math questions in the program
- To do that, we use a variable to keep track of the number of questions the user have answered correctly so far

The Improved Example

- Here is the improved program:

```
import random

number_of_questions_so_far = 0

while number_of_questions_so_far < 3:
    number1 = random.randint(1, 99)
    number2 = random.randint(1, 99)

    answer = number1 + number2
    guess = 0

    while guess != answer:
        print("What is", number1, "+", number2)
        guess = input("? ")
        guess = int(guess)

    print("You are right!")

    number_of_questions_so_far = number_of_questions_so_far + 1
```

```
>>>
What is 27 + 20
? 47
You are right!
What is 30 + 30
? 60
You are right!
What is 44 + 37
? 77
What is 44 + 37
? 71
What is 44 + 37
? 81
You are right!
>>>
```

Keep track of the number of questions answered so far

Increase the number of questions answered so far