Introduction to programming using Python

Session 3

Matthieu Choplin

http://pythonguy.com/

Objectives of Session 3

- Remainder of Session 2: Quiz + exercises
- Controlling the flow of our programs
    - While loop
    - For loop

Remainder of Session 2: Quiz (1)

- What are the two values of the **Boolean data type**? How do you write them?

Remainder of Session 2: Quiz (2)

- What are the three **Logical Boolean operators**?

Remainder of Session 2: Quiz (3)

- Write out the truth tables of each Boolean operator (that is, every possible combination of Boolean values for the operator and what they evaluate to).

Remainder of Session 2: Quiz (4)

- What do the following expressions evaluate to?
  - (5 > 4) and (3 == 5)
  - not (5 > 4)
  - (5 > 4) or (3 == 5)
  - not ((5 > 4) or (3 == 5))
  - (True and True) and (True == False)
  - (not False) or (not True)

Remainder of Session 2: Quiz (5)

- What are the six **comparison operators**?

Remainder of Session 2: Quiz (6)

- What is the difference between the **"equal to"** operator and the **"assignment"** operator?

Remainder of Session 2: Quiz (7)

- Explain what a **condition** is and where you would use one.

Remainder of Session 2: Quiz (8)

Identify the three **blocks** in this code:

```
spam = 0
if spam == 10:
    print('eggs')
    if spam > 5:
        print('bacon')
    else:
        print('ham')
    print('spam')
print('spam')
```

Remainder of Session 2: Quiz (9)

- Write code that prints "Hello" if 1 is stored in spam, prints "Howdy" if 2 is stored in spam, and prints "Greetings!" if anything else is stored in spam.

Remainder of Session 2: Quiz (10)

What is wrong in the following code? What happened if you run it?

```
radius = -20
if radius > 0:
    area = radius * radius * 3.14
print("The area is", area)
```

Objectives

- Looping with **while**
- Looping with **for**

Motivation

On one of our previous programs, we asked the user to enter a password.

- If the password was correct, we printed "Access Granted"
- Else, we printed, "Forbidden"

```
PASSWORD = 'super_password123'
password_entered = input("Enter the password: ")
if password_entered == PASSWORD:
    print("Access Granted")
else:
    print("Forbidden")
```

Motivation

However, the user only had *one chance* to enter a correct password. If the password was incorrect or correct, the program would stop.

What if we want to make the user able to try more than once to enter a correct password?

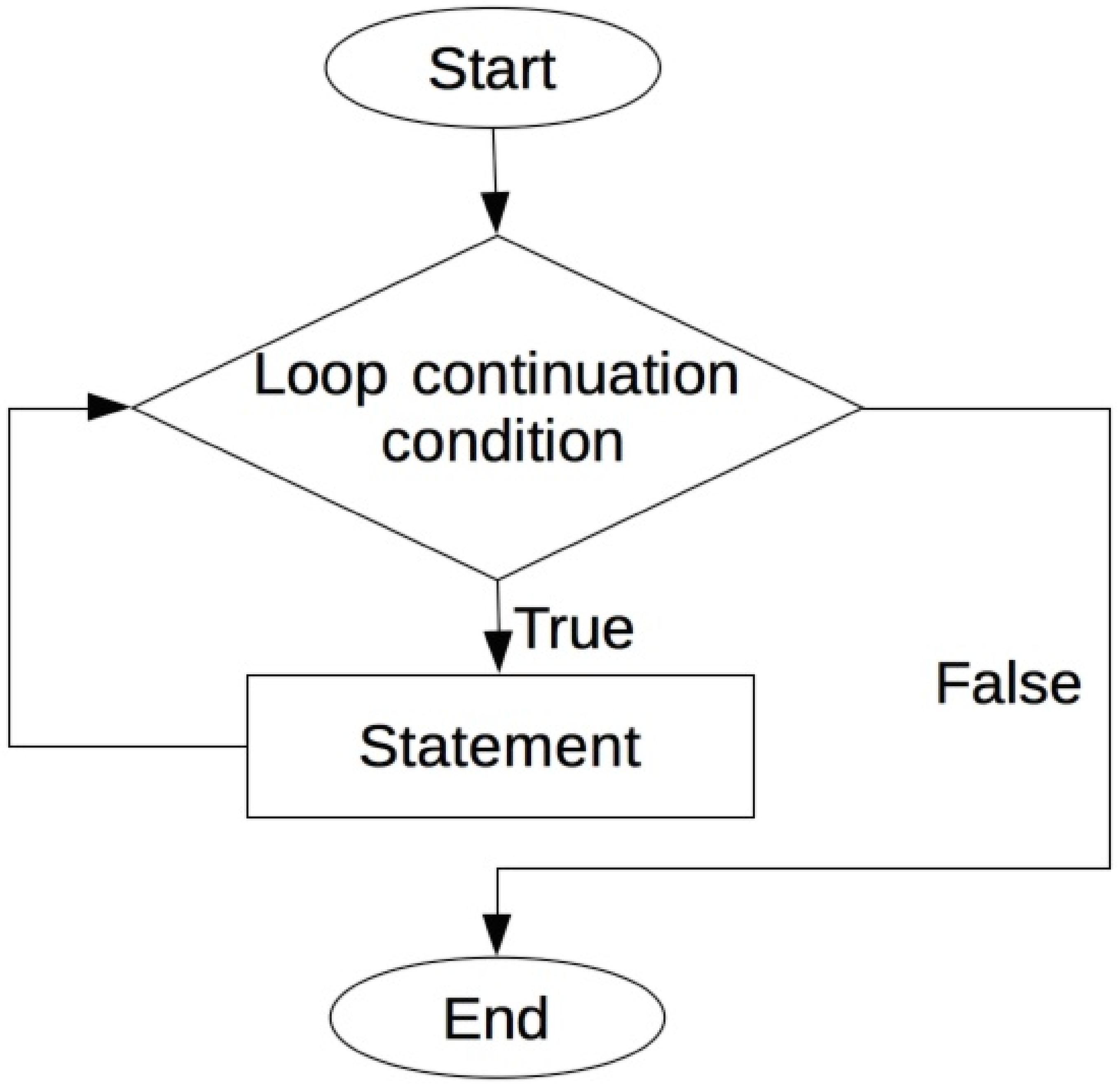
Structure of the while loop

```
while condition:
    # statement
```

Where:

- The **condition** is an expression that takes the value True or False (boolean)
- The **statement** does something, mind the **indentation**
- **While** the condition is True, the **statement** or **body** of the **loop** is executed
- Each time that the body of the loop is executed is an **iteration**

Structure of the while loop, flow chart



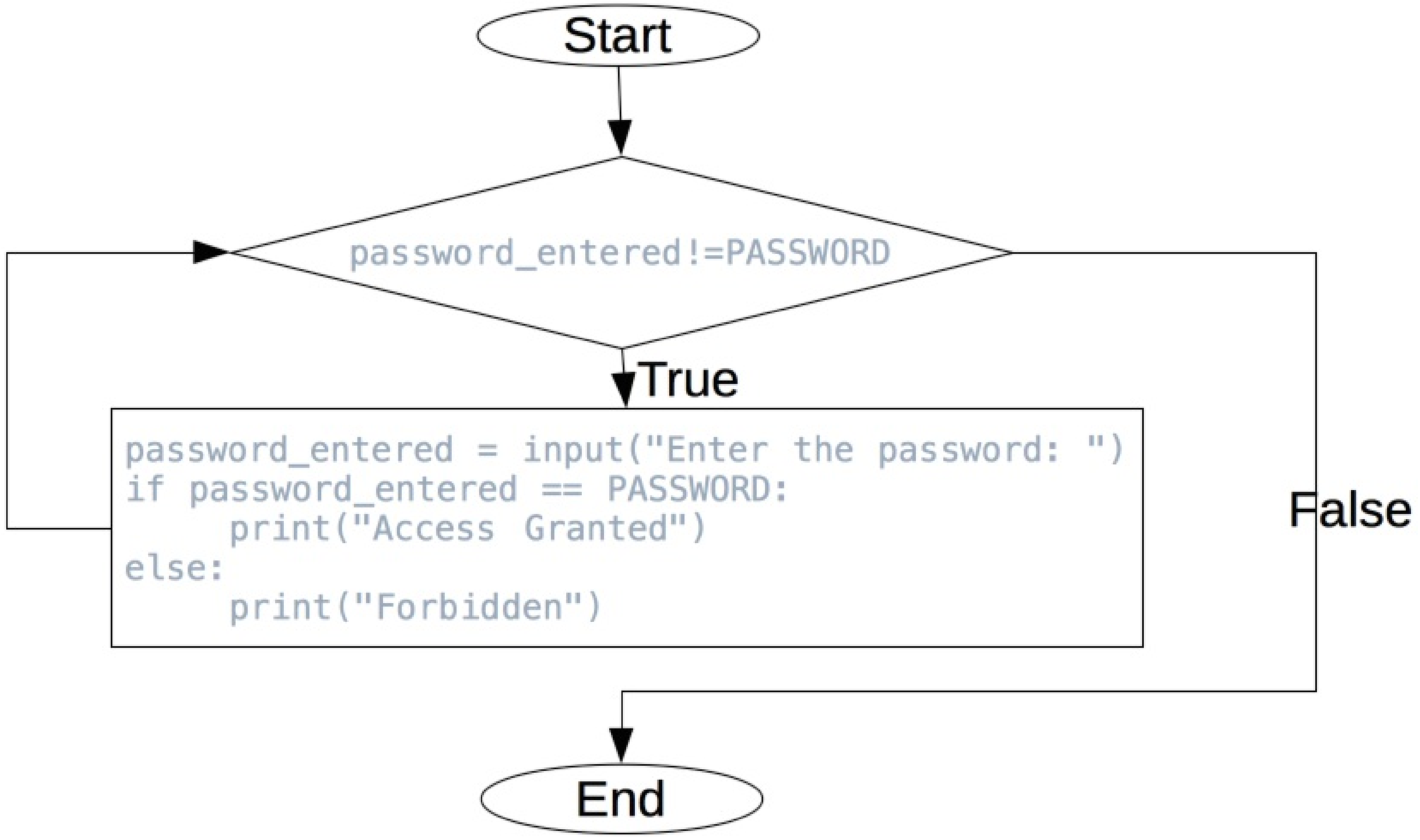
The while loop applied to our problem

```
PASSWORD = 'super_password123'
password_entered = ''
while password_entered!=PASSWORD:
    password_entered = input("Enter the password: ")
    if password_entered == PASSWORD:
        print("Access Granted")
    else:
        print("Forbidden")
```

Where:

- The **condition** is the boolean value given by the comparison of the password_entered compared to PASSWORD

Structure of the while loop, flow chart



```
password_entered = input("Enter the password: ")
if password_entered == PASSWORD:
     print("Access Granted")
else:
     print("Forbidden")
```

Reminder: using variable

You cannot use a variable that has not been declared

```
PASSWORD = 'super_password123'
while password_entered!=PASSWORD:
    password_entered = input("Enter the password: ")
    if password_entered == PASSWORD:
        print("Access Granted")
    else:
        print("Forbidden")
```

Can you see why this is wrong? Try to run this program. See the error and explain what you need to correct.

Reminder: using variable

You need to declare the variable *password_entered* before using it, else, you get:

```
NameError: name 'password_entered' is not defined
```

```
PASSWORD = 'super_password123'
password_entered = ''
while password_entered!=PASSWORD:
    password_entered = input("Enter the password: ")
    if password_entered == PASSWORD:
        print("Access Granted")
    else:
        print("Forbidden")
```

How to avoid infinite loop

Make sure that the condition gets False at some point during the execution of the program

You can implement a counter, to limit the number of **iterations**:

```
counter=0
while counter < 5:
    counter = counter + 1 # that you can also write counter+=1
    print('test infinite loop')
```

NB: counter = counter + 1 is equivalent to counter += 1

We say that we **increment** the counter at each **iteration**

The keyword break

Instead of a condition, you can also use the keyword **break** to end the iteration of a loop.

```
while True:
    print('Please type your name.')
    name = input()
    if name == 'your name':
        break
print('Thank you!')
```

The keyword continue

You can use the keyword **continue** to ignore the remaining code in the iteration and jump to the next iteration

```
sum = 0
number = 0
while number < 3:
    number += 1
    if (number ==2):
        continue
    sum += number
print("The sum is ", sum)
```

Combining break and continue

```
while True:
  print('Who are you?')
  name = input()
  if name != 'Joe':
    continue
  print('Hello, Joe. What is the password? (It is a fish.)')
  password = input()
  if password == 'swordfish':
    break
print('Access granted.')
```

Exercise: quit the program with `Q`

Enable the use to enter some text and only quit the program if he clicks on "q" or "Q"

👁 Hint

👁 Show solution

Sentinel value

This is what you have just used in the previous exercise.

A sentinel value is a value entered by the user (with input) that will make the program stop. You can put a sentinel value in your loop to decide when you want to **break** it, to stop it.

Exercise: compute average

Count positive and negative numbers and compute the average of numbers

Write a program that reads an unspecified number of integers, determines how many positive and negative values have been read, and computes the total and average of the input values (not counting zeros). Your program ends with the input 0. Display the average as a floating point number. Here is a sample run:

```
Enter an integer, the input ends if it is 0: 1
Enter an integer, the input ends if it is 0: 2
Enter an integer, the input ends if it is 0: -1
Enter an integer, the input ends if it is 0: 3
Enter an integer, the input ends if it is 0: 0
The number of positives is 3
The number of negatives is 1
The total is 5
The average is 1.25
Enter an integer, the input ends if it is 0: 0
You didn't enter any number
```

Solution: compute average

👁 Show solution

Structure of the for loop

```
for element in sequence:
    # statement
```

Where:

- **element** is a variable that is going to take the value of each element of the sequence
- **element** is NOT a keyword, it is a variable name, so you can give it whatever name you want
- the keywords are **for** and **in**
- notice the indentation that indicates the **body** of the loop (same as for **while**)

Example: a string is a sequence

A string is a sequence of characters on which we can iterate.

The value of **element** is going to be the value of each character of the string (each letter of the word) successively

```
for element in ('matt'):
  if element not in 'aeiou'):
    print(element)
```

The function range

You can create a sequence of numbers with the function **range()**

```
for element in range(initialValue, endValue, step):
    # statement
```

Where:

- **initialValue** and step value are optional arguments
- The default initialValue is 0 and the **endValue** is excluded from the interval
- **step** represents the increment and can be positive or negative

Example: range(initialValue, endValue)

Notice how the endValue is excluded

```
for v in range(4, 8):
  print(v)
```

Example: range(initialValue, endValue, step)

Step specifies the increment

```
for v in range(3, 9, 2):
  print(v)
```

Exercise: conversion from miles to kilometers

Write a program that displays the following table (note that 1 mile is 1.609 kilometres):

```
Miles Kilometres
1 1.609
2 3.218
...
9 15.481
10 16.090
```

Solution: conversion from miles to kilometers

👁 Show solution

Sequences: check point

- Sequences are objects on which we can **iterate** (by using a while or a for loop)
- For each element you have one iteration
- At this point we have seen two types of sequences:
  - string: sequences of characters (letters)
  - range object: sequences of integer (numbers)
- NB: sequences are containers, they contain objects