# DATA STRUCTURES: FILES IN PYTHON

As your program develops, so does your needs in handling data

# DATA STRUCTURES: FILES IN PYTHON

when you have lots of data to work with, using an individual variable for each piece of data gets really old, really quickly. So programmers employ some rather awesome containers (known as data structures) to help them work with lots of data. More times than not, all that data comes from a file stored on a hard disk. So, how can you work with data in your files? Turns out it’s a breeze.

# WHAT DOES THE FOLLOWING DO?

```
1 file_name ="foo.txt"
2 file_object = open(file_name)
3 print(file_object)
4 file_object.close()
```

# WHAT DOES THE FOLLOWING DO?

```
file_name ="foo.txt"
with open(file_name) as file_object:
  print(file_object)
```

# WHAT DOES THE FOLLOWING DO?

```
file_name ="foo.txt"
with open(file_name) as file_object:
  for line in file_object:
    print(line)
```

# WHAT DOES THE FOLLOWING DO?

```
file_name ="foo.txt"
with open(file_name) as file_object:
  for line in file_object:
    print(line.rstrip())
```

# WHAT DOES THE FOLLOWING DO?

```
file_name ="foo.txt"
with open(file_name) as file_object:
  for line in file_object:
    print(line.strip())
```

# WHAT DOES THE FOLLOWING DO?

```
file_name ="foo.txt"
lines_fom_file = []
with open(file_name) as file_object:
  for line in file_object:
    lines_from_file.append(line)
```

# WHAT DOES THE FOLLOWING DO?

```
file_name ="foo.txt"
names = ['ada', 'bob', 'charlie', 'erin']
with open(file_name, 'w') as file_object:
  for line in file_object:
    file_object.write(line)
```

## SURF'S UP IN CODEVILLE

The annual Codeville Surf-A-Thon is more popular than ever this year.
Because there are so many contestants, the organizers asked you to write a Python program to process the scores. Eager to please, you agreed.
The trouble is, even though the contest is over and the beach is now clear, you can't hit the waves until the program is written. Your program has to work out the highest surfing scores. Despite your urge to surf, a promise is a promise, so writing the program has to come first.

## FIND THE HIGHEST SCORE IN THE RESULTS FILE

After the judges rate the competitors, the scores are stored in a file called **report.txt**. There is one line in the file for each competitor's score. You need to write a program that reads through each of these lines, picks out the score, and then works out the highest score in the Surf-A-Thon.

- 8.65
- 9.12
- 8.45
- 7.81
- 8.05
- 7.21
- 8.31

## HOW DO WE READ AND ITERATE THAT FILE?

If you need to read from a file using Python, one way is to use the built-in open() command. Open a file called results.txt like this:

```
with open("results.txt") as results_file:
  for each_line in results_file:
    print(each_line)
```

## PYTHON GAME

```
with open("results.txt") as results_file:
  for each_line in results_file:
    print(each_line)
```

The call to open() creates what is known as a file handler. A file handler is a shortcut that you use to refer to the file

## PYTHON GAME

```
with open("results.txt") as results_file:
  for each_line in results_file:
    print(each_line)
```

Because you'll need to **the file one line at a time**, Python gives you the for loop for just this purpose. Like while loops, the for loop runs repeatedly, running the loop code once **for each** of the items in something.

## PYTHON GAME

```
with open("results.txt") as results_file:
  for each_line in results_file:
    print(each_line)
```

Each time the body of the for loop runs, a variable is set to a string containing the current line of text in the file. This is referred to as **iterating** through the data in the file:

# CODE MAGNETS

```
  highest_score = 0
  with open("results.txt") as results_file:
    for each_line in results_file:
      temp = ____(____)
        if temp _ _______"
        _______ = ____
print(f"The highest score was
{highest_score}")
```

- highest_score
- float
- >
- each_line
- highest_score
- temp

# CODE MAGNETS SOLUTION

```
highest_score = 0
with open("results.txt") as results_file:
  for each_line in results_file:
    temp = float(each_line)
      if temp > highest_score"
        highest_score = temp
print(f"The highest score was
{highest_score}")
```

# WHAT HAPPENS IF THE FILE CONTAINS MORE THAN NUMBERS?

To see what happened, let's take another look at the judge's score sheet to see if you missed anything:

# MORE THAN NUMBERS

- ada 8.65
- bob 9.12
- charles 8.45
- erin 7.81
- fred 8.05
- georgia 7.21
- harry 8.31

## SPLITTING STRINGS

You need to somehow extract the score from the string. In each line, there is a name, followed by a space, followed by the score

Programmers often have to deal with data in strings that contain several pieces of data separated by spaces. It’s so common, in fact, that Python provides a special string method to perform the cutting you need: split().

# HIGHEST SCORE PART II

```
highest_score = 0
with open("results.txt") as results_file:
  for each_line in results_file:
    name, score = each_line.split(" ")
    print(f"Hey, {name} scored {score})
print(f"The highest score was
{highest_score}")
```

can you write some code that finds both the highest score and writes their name?

# HIGHEST SCORE PART II SOLUTION

```
highest_score = 0
highest_name = ""
with open("results.txt") as results_file:
  for each_line in results_file:
    name, score = each_line.split(" ")
    score = float(score)
    if score > highest_score:
      highest_score = score
      highest_name = name
print(f"The highest score was
{highest_score} with {highest_score}")
```

can you write some code that finds both the highest score and writes their name?

Speaker notes