Parables and Pythons

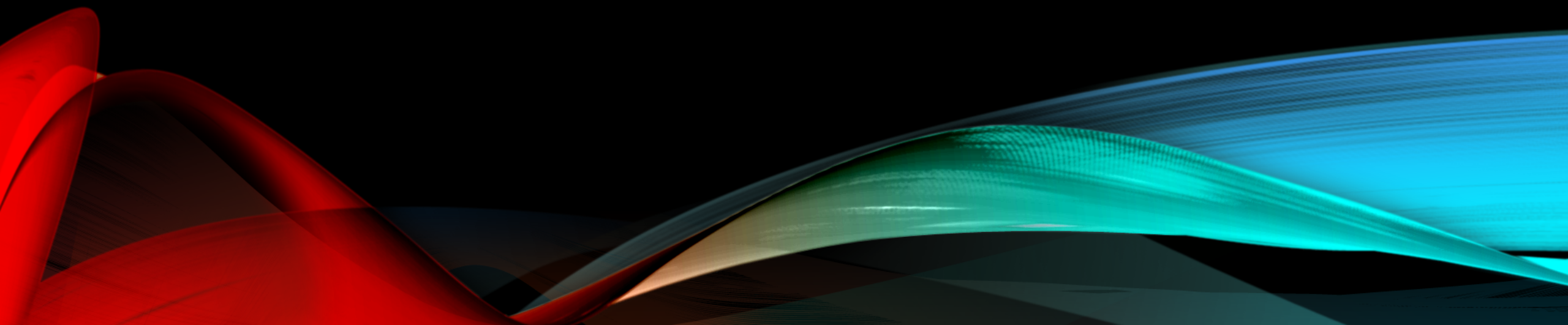
Dr. Charles “Chuck” Bell

Lesson 9: 14 November 2018

CLASS AGENDA

- Bible Study
  - The shortest parables
- The Golden Age of British Comedy
  - Stoning
- Computer Programming with Python
  - Review of Concepts
  - Hands-On Practice
    - Sample challenges
- Homework

BIBLE STUDY: THE PARABLES OF JESUS



THE SHORTEST PARABLES

- There are two parables that are very short, but go together - the mustard seed and the yeast (leaven).
- They are in response to Jesus being asked, "What is the Kingdom of Heaven like?"
- They are found in several books in the Gospel with nearly identical wording with very little variation. So, they must be important.
  - Mathew 13:31-33 (both)
  - Mark 4:30-32 (mustard seed)
  - Luke 13:18-21 (both)
  - Thomas 20 (non-canonical writings - mustard seed)
- Theses parables are often considered controversial because, while they appear together and seem related, some believe they have different meanings.

THE SHORTEST PARABLES – MATHEW 13:31-33

- (Matt 13:31): *He told them another parable: "The kingdom of heaven is like a mustard seed, which a man took and planted in his field.*
- (Matt 13:32): *Though it is the smallest of all seeds, yet when it grows, it is the largest of garden plants and becomes a tree, so that the birds come and perch in its branches."*
- (Matt 13:33): *He told them still another parable: "The kingdom of heaven is like yeast that a woman took and mixed into about sixty pounds of flour until it worked all through the dough."*
- These parables belong together. Each should help us to understand the other, but of all the parables Christ told, none has produced such diametrically opposed interpretations as these two.

THE MUSTARD SEED - CONTROVERSY

- This parable has come under a lot of scrutiny and some conclude it is a parody or myth.
  - The mustard plant was not something a gardener or farmer would plant (cultivate).
  - It grows wild in most places
  - Often used as a 'cover' crop for fields
  - In Palestine, mustard plants mostly grew in the desert areas
  - The mustard plant could never grow large enough to support many birds and no bird nests.
    - There is one variant of the black mustard plant that can grow up to 10 feet tall, but it is mostly found near the Jordan. Keep in mind, trees in the region rarely grew larger
    - The black mustard plant, while it has stems that resembles tree trunks, it not strong enough to support bird nests, but could support a few birds perching or resting on the limbs if the wind were not blowing
  - Mustard plants, in general, grow into shrub-like plants and not trees
- Given these, some argue the illustration is flawed and thus is a 'negative' message

THE MUSTARD SEED - INTERPRETATIONS

- On the one hand, some teachers see these as parables of the kingdom's expansion and growth, so that in time it actually comes to fill the whole world.
- Thus it is a positive message predicting the grow of the Church and therefore the Kingdom of God – starting from a small number of followers to a world-wide gathering of the faithful for which God has prepared Heaven.
- This interpretation is supported by interpreting the following parable about yeast.

THE MUSTARD SEED - INTERPRETATIONS

- On the other hand, the mustard seed and the resulting plant is a very poor illustration for several botanical reasons including the size of the plant, its deplorable cultivation, and unlikely metaphor.
- Thus, the message conveyed is negative in that it doesn't make sense to compare Heaven to the mustard plant.
- This interpretation is also supported by interpreting the following parable about yeast in a positive manner.

THE YEAST - INTERPRETATIONS

- We see similar dichotomy of interpretations in this parable - even though it is only one sentence long.
- The positive position likens the growth of the church with bread that has been leavened (yeast added) so that it expands. Thus, the Kingdom of Heaven will grow larger with the addition of the redeemed who grow in Christ.
- However, the negative position has a lot of merit.

THE YEAST - INTERPRETATIONS

- The negative position suggests that the yeast or "leaven" in that age was a small portion of spoiled bread dough, which is added to good dough to make it rise.
- In nearly all cases in the Old Testament (and in Jewish life today) yeast is a symbol of evil. In the sacrificial laws of Israel it was/is excluded from every offering to the Lord made by fire.
- At the time of the feast of unleavened bread, every faithful Jew was to search his home for any trace of yeast and then get rid of it. That is done today by orthodox Jews and symbolizes for them, as it did earlier, the putting away of sin.
- Jesus spoke of the leaven (or yeast) of the Pharisees and Sadducees, and Herod, in each case meaning their evil influence (Matt. 16:12; Mark 8:15).
- Thus, the Kingdom of Heaven will contain "bad" people - people who are spoiled or not holy.

CONCLUSIONS

- The mustard seed very likely was well known in Jesus' time and his audience would have picked up the abnormal growth of a shrub into a tree. Thus, a negative view of this parable can be argued.
- However, Jesus' story telling was often meant to evoke thought and, at times, even controversy. If we focus too much on the subjects out of context (words like mustard seed), we will miss the point altogether. In fact, we will allow our own thinking to take our eyes off of the Kingdom of Heaven.
- Likewise, yeast has been used in the Old Testament as a symbol of evil, but it is also a very useful ingredient for baking bread - sometimes it is simply yeast.
- It is difficult to see how an important and thoroughly understood symbol of evil could be used by Jesus to represent the exact opposite, namely, the blessed impact of His gospel on the world.

CONCLUSIONS

- Also, it is significant that these two parables are bracketed by that of the devil's work in sowing tares among the wheat (vv. 24-30), and Christ's explanation of that parable (vv. 36-43). This suggests they should be taken not as teaching something entirely different from the parable of the tares, but as expanding it.
  - Read the other parables!
- Therefore, we should look at these parables as positive messages that predict not only the triumph of God's Kingdom, but more importantly we should take comfort that Jesus has promised that Heaven will be "leavened" by turning the sinner into a positive impact for Christ. That, I believe, is what Jesus was trying to say in these parables.

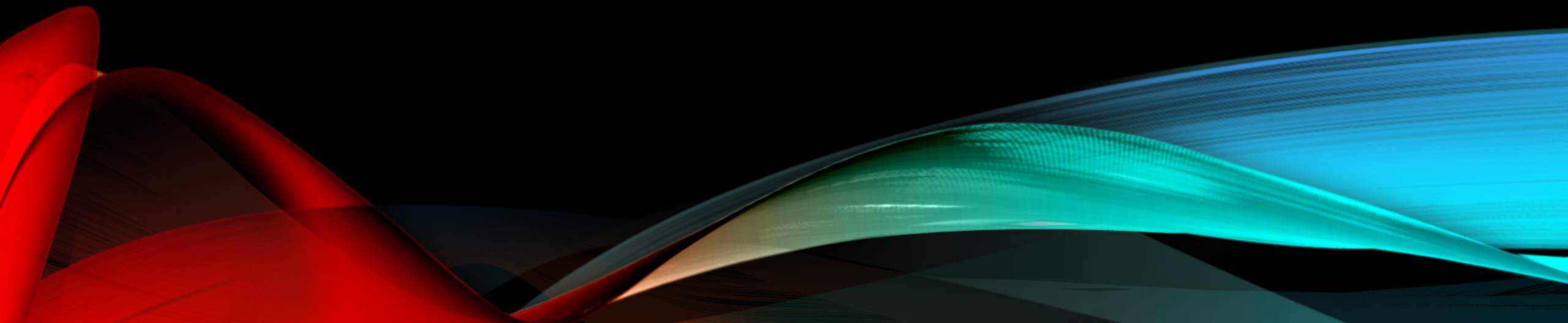
THE GOLDEN AGE OF BRITISH COMEDY

Subtle nuance: women were not allowed to participate.

https://www.youtube.com/watch?v=FQ5YU_spBw0

COMPUTER PROGRAMMING

Hands On Learning



EXPRESSIONS

- **expression**: A data value or set of operations to compute a value.
  Examples: `1 + 4 * 3`
  `42`

- Arithmetic operators we will use:
  `+ - * /` addition, subtraction/negation, multiplication, division
  `%` modulus, a.k.a. remainder
  `**` exponentiation

- **precedence**: Order in which operations are computed.
  - `* / % **` have a higher precedence than `+ -`
    `1 + 3 * 4` is `13`
  - Parentheses can be used to force a certain order of evaluation.
    `(1 + 3) * 4` is `16`

MATH FUNCTIONS

- Python has useful commands for performing calculations.
- To use many of these commands, you must write the following at the top of your Python program:

```
from math import *
```

Constant	Description
`e`	2.7182818...
`pi`	3.1415926...

Command name	Description
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	rounds down
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`max(`***value1***`,` ***value2***`)`	larger of two values
`min(`***value1***`,` ***value2***`)`	smaller of two values
`round(`***value***`)`	nearest whole number
`sin(`***value***`)`	sine, in radians
`sqrt(`***value***`)`	square root

VARIABLES

- **variable**: A named piece of memory that can store a value.
  - Usage: Compute an expression's result, store that result into a variable, and use that variable later in the program.
- **assignment statement**: Stores a value into a variable.

- Syntax:

  name `=` ***value***

- Examples:

```
x = 5
gpa = 3.14
```

- A variable that has been given a value can be used in expressions.

  `x + 4` is `9`

INPUT

- `input` : Reads a number from user input.
  - You can assign (store) the result of `input` into a variable.
  - Example:

```
age = input("How old are you? ")
print "Your age is", age
print "You have", 65 - age, "years until retirement"
```

Output:

```
How old are you? 53
Your age is 53
You have 12 years until retirement
```

THE FOR LOOP

- **`for` loop**: Repeats a set of statements over a group of values.
  - Syntax:

    `for` ***variableName*** `in` ***groupOfValues*** **:**
        statements
    - We indent the statements to be repeated with tabs or spaces.
    - ***variableName*** gives a name to each value, so you can refer to it in the ***statements***.
    - ***groupOfValues*** can be a range of integers, specified with the `range` function.
  - Example:

```
for x in range(1, 6):
    print x, "squared is", x * x
```

    Output:

```
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
```

RANGE FUNCTION

- The `range` function specifies a range of integers:
  - `range(`***start***`, `***stop***`)` - the integers between ***start*** (inclusive) and ***stop*** (exclusive)
- It can also accept a third value specifying the change between values.
  - `range(`***start***`, `***stop***`, `***step***`)` - the integers between ***start*** (inclusive) and ***stop*** (exclusive) by ***step***
  - Example:

```
for x in range(5, 0, -1):
    print x
print "Blastoff!"
```

Output:

```
5
4
3
2
1
Blastoff!
```

IF

- **`if` statement**: Executes a group of statements only if a certain condition is true. Otherwise, the statements are skipped.
  - Syntax:

```
if condition :
    statements
```

- Example:

```
gpa = 3.4
if gpa > 2.0:
    print "Your application is accepted."
```

IF/ELSE

- **`if/else` statement**: Executes one block of statements if a certain condition is True, and a second block of statements if it is False.
  - Syntax:

```
if condition:
    statements
else:
    statements
```

- Example:

```
gpa = 1.4
if gpa > 2.0:
    print "Welcome to Mars University!"
else:
    print "Your application is denied."
```

- Multiple conditions can be chained with `elif` ("else if"):

```
if condition:
    statements
elif condition:
    statements
else:
    statements
```

WHILE

- **`while` loop**: Executes a group of statements as long as a condition is True.
  - good for *indefinite loops* (repeat an unknown number of times)
- Syntax:

```
while condition:
    statements
```

- Example:

```
number = 1
while number < 200:
    print number,
    number = number * 2
```

  - Output:

```
1 2 4 8 16 32 64 128
```

LOGIC

- Many logical expressions use *relational operators*:

Operator	Meaning	Example	Result
==	equals	1 + 1 == 2	True
!=	does not equal	3.2 != 2.5	True
<	less than	10 < 5	False
>	greater than	10 > 5	True
<=	less than or equal to	126 <= 100	False
>=	greater than or equal to	5.0 >= 5.0	True

- Logical expressions can be combined with *logical operators*:

Operator	Example	Result
and	9 != 6 and 2 < 3	True
or	2 == 3 or -1 < 5	True
not	not 7 > 0	False

STRINGS

- **string**: A sequence of text characters in a program.
  - Strings start and end with quotation mark `"` or apostrophe `'` characters.
  - Examples:

```
"hello"
"This is a string"
"This, too, is a string.   It can be very long!"
```

- A string may not span across multiple lines or contain a " character.

```
"This is not
a legal String."
"This is not a "legal" String either."
```

- A string can represent characters by preceding them with a backslash.
  - `\t` tab character
  - `\n` new line character
  - `\"` quotation mark character
  - `\\` backslash character
  - Example: `"Hello\tthere\nHow are you?"`

INDEXES

- Characters in a string are numbered with *indexes* starting at 0:
  - Example:

```
name = "P. Diddy"
```

index	0	1	2	3	4	5	6	7
character	P	.		D	i	d	d	y

- Accessing an individual character of a string:
  variableName `[` ***index*** `]`
  - Example:

```
print name, "starts with", name[0]
```

  Output:

```
P. Diddy starts with P
```

STRING PROPERTIES

- `len(`***string***`)` - number of characters in a string (including spaces)
- `str.lower(`***string***`)` - lowercase version of a string
- `str.upper(`***string***`)` - uppercase version of a string

- Example:

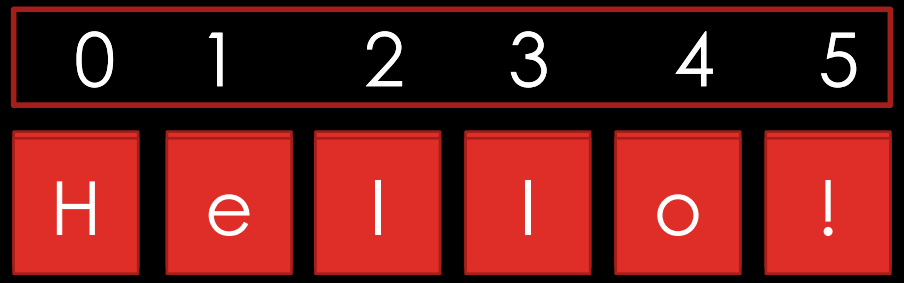
```
name = "Martin Douglas Stepp"
length = len(name)
big_name = str.upper(name)
print big_name, "has", length, "characters"
```

Output:

```
MARTIN DOUGLAS STEPP has 20 characters
```

LISTS - REVIEW

- The simplest list is one we've already used: a character string!
- A list is a contiguous segment of memory that stores a set of values.
- Lists are "counted" or "indexed" starting at 0.
- List indexes use the [n] syntax where n is the index (number).
- A list can hold any number of values.
- Lists have additional methods that we can use to manipulate the data.



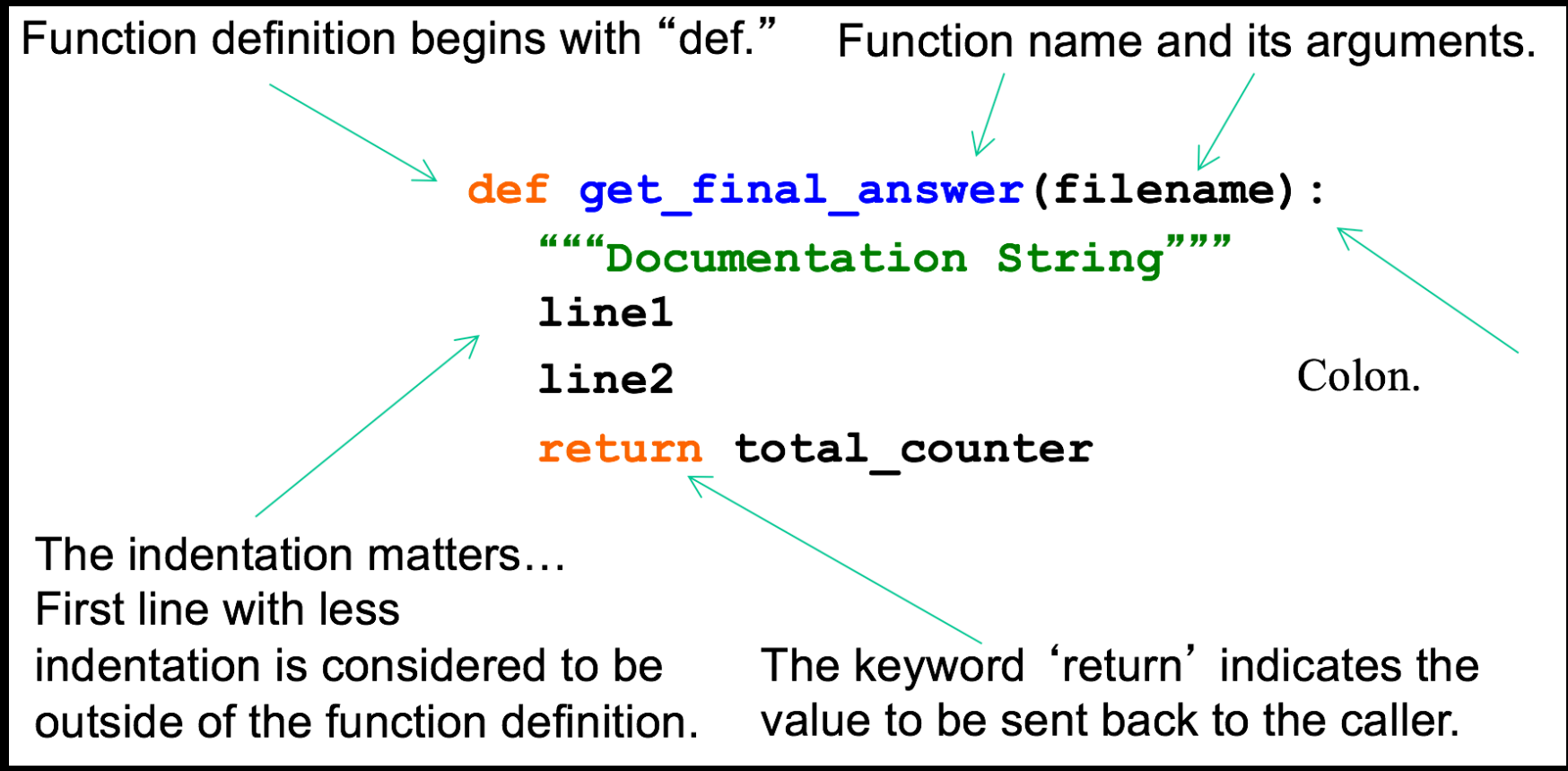
REVIEW: DICTIONARIES

- Store ***pairs*** of entries called ***items***
  { 'CS' : '743-713-3350', 'UHPD' : '713-743-3333'}
- Each pair of entries contains
  - A ***key***
  - ***A value***
- Key and values are separated by a colon
- Paris of entries are separated by commas
- Dictionary is enclosed within curly braces

REVIEW: DICTIONARIES

- Strings, lists, tuples, sets and dictionaries all deal with aggregates
- Two big differences
  - ***Lists*** and ***dictionaries*** are ***mutable***
    - Unlike strings, tuples and sets
  - ***Strings***, ***lists*** and ***tuples*** are ***ordered***
    - Unlike sets and dictionaries

DEFINING A FUNCTION



SAMPLE CHALLENGES

- The following are three sample challenges like the ones we will have during our programming contest, which is **NEXT WEEK!**
- Rules for the programming contest:
  - You can attempt as many challenges as you like.
  - Each challenge has a maximum point total – those with higher values are more complicated.
  - You can earn partial credit for each challenge if it is partially correct, but there must be **no syntax errors**.
  - You may **not** use the Internet to search for solutions.
  - You may ask for explanation of technical details, but you may not ask any Python questions.
  - The student with the highest point total wins.
  - Winner will be notified by email.

CHALLENGE #1 – 1 POINT

- We have a function that controls two monkeys, a and b, and the parameters `a_smile` and `b_smile` indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return True if we are in trouble.
- Solve the challenge by completing this code:

```
def monkey_trouble(a_smile, b_smile):
  # Complete this code

print("Are we in trouble if:")
print("Both A and B are smiling?", monkey_trouble(True, True))
print("Neither A nor B are smiling?", monkey_trouble(False, False))
print("Only A is smiling?", monkey_trouble(True, False))
print("Only B is smiling?", monkey_trouble(False, True))
```

CHALLENGE #2 – 10 POINTS

- Make a two-player Rock-Paper-Scissors game. (*Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game*)
- Remember the rules:
  - Rock beats scissors
  - Scissors beats paper
  - Paper beats rock
- Use a function named **find_winner()** that accepts two parameters; the answer from player 1 and the answer from player 2.

CHALLENGE #3 – 20 POINTS

- Write a program that asks the user how many Fibonacci numbers to generate and then generates them. Take this opportunity to think about how you can use functions. Make sure to ask the user to enter the number of numbers in the sequence to generate.(*Hint: The Fibonacci sequence is a sequence of numbers where the next number in the sequence is the sum of the previous two numbers in the sequence. The sequence looks like this: 1, 1, 2, 3, 5, 8, 13, …*)

CHALLENGE #4 – 50 POINTS

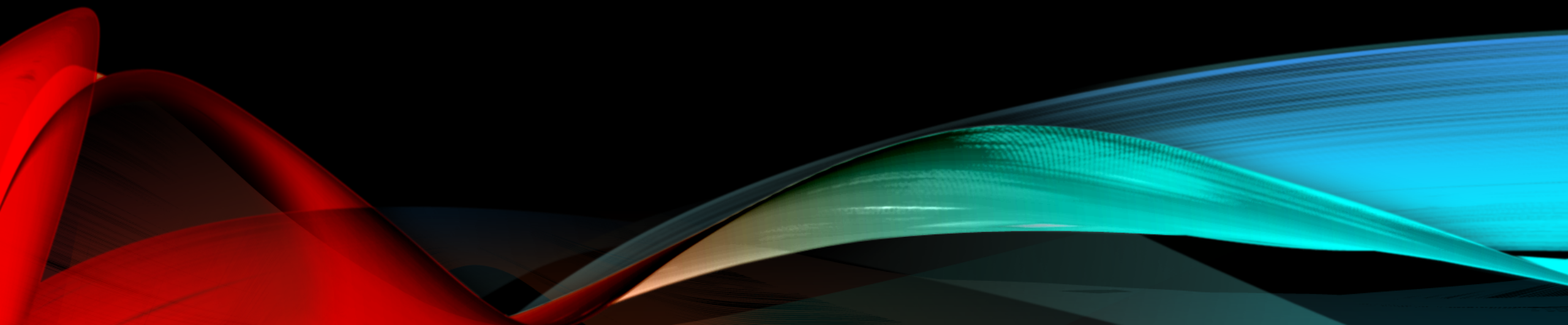
- Hashmat is a brave warrior who with his group of young soldiers moves from one place to another to fight against his opponents. Before Fighting he must calculate one thing: the difference between his and the opponent's soldiers (in number). From this difference he decides whether to fight or not. Hashmat's will not fight if his enemy forces exceed 120% of his own force.
- The input contains two numbers in every line separated by a comma. These two numbers in each line denote the number soldiers in Hashmat's army and his opponent's. The input numbers are not greater than 2^32. Input is terminated by 0, 0.
- For each line of input, if Hashmat withdraws, randomly determine the number of soldiers lost during the withdraw limited to no more than 10%. If there is a battle, use a random number of casualties for each side and determine a winner based on superior number of survivors.
- Sample Input/output

10, 12

> Hashmat attacks! Friendly losses: 3, enemy losses: 6. Hashmat wins!

HOMEWORK

All homework assignments can be handed in on hardcopy (with your name at the top) or emailed to me at drcharlesbell@gmail.com.



HOMEWORK #7 REVIEW

```
import json

def get_string(prompt):
    return str(input("{0}".format(prompt)))

def get_record():
    """Prompt user for a record.
    Returns dictionary.
    """
    first, last = get_string("First and last name (use space): ").split()
    street = get_string("Street: ")
    city, state, zipcode = get_string("City, state, and zip (use commas): ").split(",")
    more_phones = 'y'
    phones = []
    while (more_phones == 'y') or (more_phones == 'Y'):
        phone = get_string("Enter a phone number (XXX) XXX-XXXX: ").strip()
        phones.append(phone)
        more_phones = get_string("Enter another phone? [Y/N] ")
    record = {
        'first': first.strip(),
        'last': last.strip(),
        'address': {
            'street': street.strip(),
            'city': city.strip(),
            'state': state.strip(),
            'zip': zipcode.strip(),
        },
        'phones': phones
    }
    return record
```

HOMEWORK #7 REVIEW

```
def print_rolodex(rolodex):
    """Display the rolodex"""
    print("Contents of the rolodex (default):")
    for record in rolodex:
        print(record)
    print()
    print("Contents of the rolodex (JSON):")
    for record in rolodex:
        print(json.dumps(record, sort_keys=True, indent=4, separators=(',', ': ')))
    print()
    print("Alphabetical list of names:")
    last_names = [(r['last'], r['first']) for r in myRolodex]
    last_names.sort()
    for last_name in last_names:
        print("{0}, {1}".format(last_name[0], last_name[1]))

print("Welcome to the rolodex simulator (v2)!")
myRolodex = []
repeat = 'y'
while (repeat == 'y') or (repeat == 'Y'):
    myRolodex.append(get_record())
    print()
    repeat = get_string("Enter another record? [Y/N] ")
    print()
# Print out the data
print_rolodex(myRolodex)
print("bye!")
```

HOMEWORK #7 - REVIEW

```
...
Contents of the rolodex (default):
{'first': 'Joe', 'last': 'Smith', 'address': {'street': '123 East Main', 'city':
'Tappahannock', 'state': 'VA', 'zip': '22521'}, 'phones': ['804 555-1212']}
{'first': 'Amos', 'last': 'Anthony', 'address': {'street': '500 Oracle Pkwy', 'city': 'Glen
Forest', 'state': 'CA', 'zip': '90125'}, 'phones': ['560 344-1212']}

Contents of the rolodex (JSON):
{
    "address": {
        "city": "Tappahannock",
        "state": "VA",
        "street": "123 East Main",
        "zip": "22521"
    },
    "first": "Joe",
    "last": "Smith",
    "phones": [
        "804 555-1212"
    ]
}
...
```

HOMEWORK #7 - REVIEW

```
...
{
    "address": {
        "city": "Glen Forest",
        "state": "CA",
        "street": "500 Oracle Pkwy",
        "zip": "90125"
    },
    "first": "Amos",
    "last": "Anthony",
    "phones": [
        "560 344-1212"
    ]
}

Alphabetical list of names:
Anthony, Amos
Smith, Joe
bye!
```

HOMEWORK ASSIGNMENT #8

- (2 points) Make a tic-tac-toe game that takes input in the form of row, col for each player. Use a list of lists to store the board and contents. Prompt until a winner is determined.
  - `ttt_board = [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]]` -1 = not chosen, X or 0
- (2 bonus points) Use a function to print the "board".
  - `print_board(board, row, col)` – takes a list of
- (1 bonus point) Use a function to determine if there is a winner:
  - `check_for_winner(board)` – print the contents of the rolodex.

HOMEWORK ASSIGNMENT #8

- WAIT! How do you print a tic-tac-toe board on the screen?
- Notice on your keyboard there is a | and – as well as +.
- Consider what this does:

```
board = [[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]]
print("+-+-+-+")
for row in board:
    for col in range(0,3):
        if row[col] in ('X','O'):
            print("|{0}".format(row[col]), end='')
        else:
            print("| ", end='')
    print("|")
    print("+-+-+-+")
```

HOMEWORK ASSIGNMENT #8

- Example Output:

```
$ python3 ./ttt1.py
+-+-+-+
| | | |
+-+-+-+
| | | |
+-+-+-+
| | | |
+-+-+-+
```

QUESTIONS OR COMMENTS?