Parables and Pythons

Dr. Charles "Chuck" Bell

Lesson 8: 7 November 2018

CLASS AGENDA

- Bible Study
  - The rich fool
- The Golden Age of British Comedy
  - Deductive Reasoning
- Computer Programming with Python
  - Review:
    - Dictionaries
  - Hands-On Practice
    - Functions
- Homework

BIBLE STUDY: THE PARABLES OF JESUS

BACKGROUND

- Jesus was teaching important lessons to his disciples in the presence of a multitude and one in the crowd made a request of Jesus (Luke 12:1-13)
- While Jesus was teaching some spiritual lessons, this man had his mind on earthly things; Jesus used the opportunity to teach some valuable lessons in the parable of "the rich fool"

WHY DID THE MAN PETITION JESUS?

- According to Jewish custom, rabbis could settle legal disputes when it came to the division of property between heirs, and that explains why the man described in today's passage came to Jesus to get a share of his brother's inheritance (Luke 12:13).
- But what is immediately striking about this passage is that Jesus did not take the opportunity to exercise His right to judge between the two brothers. Instead, the encounter provided Him with an opportunity to speak a parable warning about covetousness.

LUKE 12:13-21 - THE RICH FOOL

13 Someone in the crowd said to him, “Teacher, tell my brother to divide the inheritance with me.” 14 Jesus replied, “Man, who appointed me a judge or an arbiter between you?” 15 Then he said to them, “Watch out! Be on your guard against all kinds of greed; life does not consist in an abundance of possessions.” 16 And he told them this parable: “The ground of a certain rich man yielded an abundant harvest. 17 He thought to himself, ‘What shall I do? I have no place to store my crops.’ 18 “Then he said, ‘This is what I’ll do. I will tear down my barns and build bigger ones, and there I will store my surplus grain. 19 And I’ll say to myself, “You have plenty of grain laid up for many years. Take life easy; eat, drink and be merry.”’ 20 “But God said to him, ‘You fool! This very night your life will be demanded from you. Then who will get what you have prepared for yourself?’ 21 “This is how it will be with whoever stores up things for themselves but is not rich toward God.”

POSITIVE CHARACTERISTICS

- This man was rich – “a certain rich man” (Abraham, Job, Joseph of Arimathea)
- This man made his wealth honestly – “the ground … brought forth plentifully”
- This man had foresight – “he reasoned within himself” (Mt. 25:14ff)
- This man had resolve – “This will I do”

A TEST OF CHARACTER

- A change of fortune, whether good (rich farmer) or bad (Job), reveals character
- There are those who gain a level of success and they allow it to “go to their head”; they are no longer “down to earth”, but they are filled with pride, superiority, immorality, etc.
- Saul changed from humility (1 Sam. 10:20-24) to pride (1 Sam. 13; 15)

NEGATIVE CHARACTERISTICS

- He thought about his gifts, but forgot about his Giver (Psa. 24:1; Jas. 1:17)
- He thought about keeping, but forgot about giving (Mt. 6:19-21; Gal. 6:10; 1 Tim. 6:17-19; 1 Jn. 3:17)
- He thought about his physical body, but forgot about his spiritual soul (Mt. 4:4; 2 Cor. 4:16 – 5:10; 1 Tim. 4:8; 6:6-11).
- He thought about his life ("many years"), but forgot about his death – "this night" (1 Cor. 15:32-33; Heb. 9:27; Jas. 4:13-15)
- He thought about life as "eating and drinking", but forgot about life as serving God (Eccl. 2:1-11; 12:13-14)
- He thought about himself ("I … my" = 11 times) , but not about his God (Lk. 12:31,34).

KEEP YOURSELVES FROM ALL COVETOUSNESS

- Consider how many times Jesus says or does something in his ministry against a person's money or possessions
- Covetousness (Gr. pleonexias) means "a desire for more, always in a bad sense" (Mk. 7:22; Lk. 12:15; Rom. 1:29; 1 Cor. 5:11; 2 Cor. 9:5; Eph. 4:19; 5:3; Col. 3:5; 1 Thess. 2:5; 2 Pet. 2:3,14)
- The covetous man is the "rich poor" man (Rev. 3:15-19)
- The solution is to get our minds on things that are above worldly wealth (Col. 3:1-2, 5).

CONCLUSIONS

- Covetousness manifests itself in a lack of gratitude and generosity.
- There is nothing inherently wrong with being wealthy or seeking to increase one’s prosperity.
- The danger arises when we make riches our chief end, when we are never satisfied with what we have but think that acquiring more stuff will make us happy.
- That is what we see in the parable of the rich fool. The rich man did not stop to thank the Lord for his prosperity. He was dissatisfied with what he had and wanted bigger and better barns so that he could hold even more.

CONCLUSIONS

- He strove to acquire more and more because he prized self-sufficiency instead of a life of dependence upon God.
- He did not seek to help the poor, and thus failed to show trust that the Lord would continue to provide for him (Luke 12:13–19).
- What was the end of this man? God judged him for his idolatrous treatment of his wealth (vv. 20–21). People who are impenitently covetous will be likewise condemned for their lack of thankfulness and generosity.

THE GOLDEN AGE OF BRITISH COMEDY

BUT FIRST, A BIT OF HISTORICAL CONTEXT

- The following video contains a parody of flagellation.
- Flagellants are practitioners of an extreme form of mortification of their own flesh by whipping it with various instruments.
- Most notably, Flagellantism was a 14th-century movement, consisting of radicals in the Catholic Church. It began as a militant pilgrimage and was later condemned by the Catholic Church as heretical.
- The followers were noted for including public flagellation in their rituals.

https://www.youtube.com/watch?v=H9PY_3E3h2c

COMPUTER PROGRAMMING

Hands On Learning

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

WHAT ARE FUNCTIONS

- A function is a piece of code in a program. The function performs a specific task. The advantages of using functions are:
  - Reducing duplication of code
  - Decomposing complex problems into simpler pieces
  - Improving clarity of the code
  - Reuse of code
  - Information hiding
- Functions in Python are first-class citizens. It means that functions have equal status with other objects in Python. Functions can be assigned to variables, stored in collections or passed as arguments. This brings additional flexibility to the language.
- There are two basic types of functions. Built-in functions and user defined ones. The built-in functions are part of the Python language. Examples are: dir(), len() or abs().

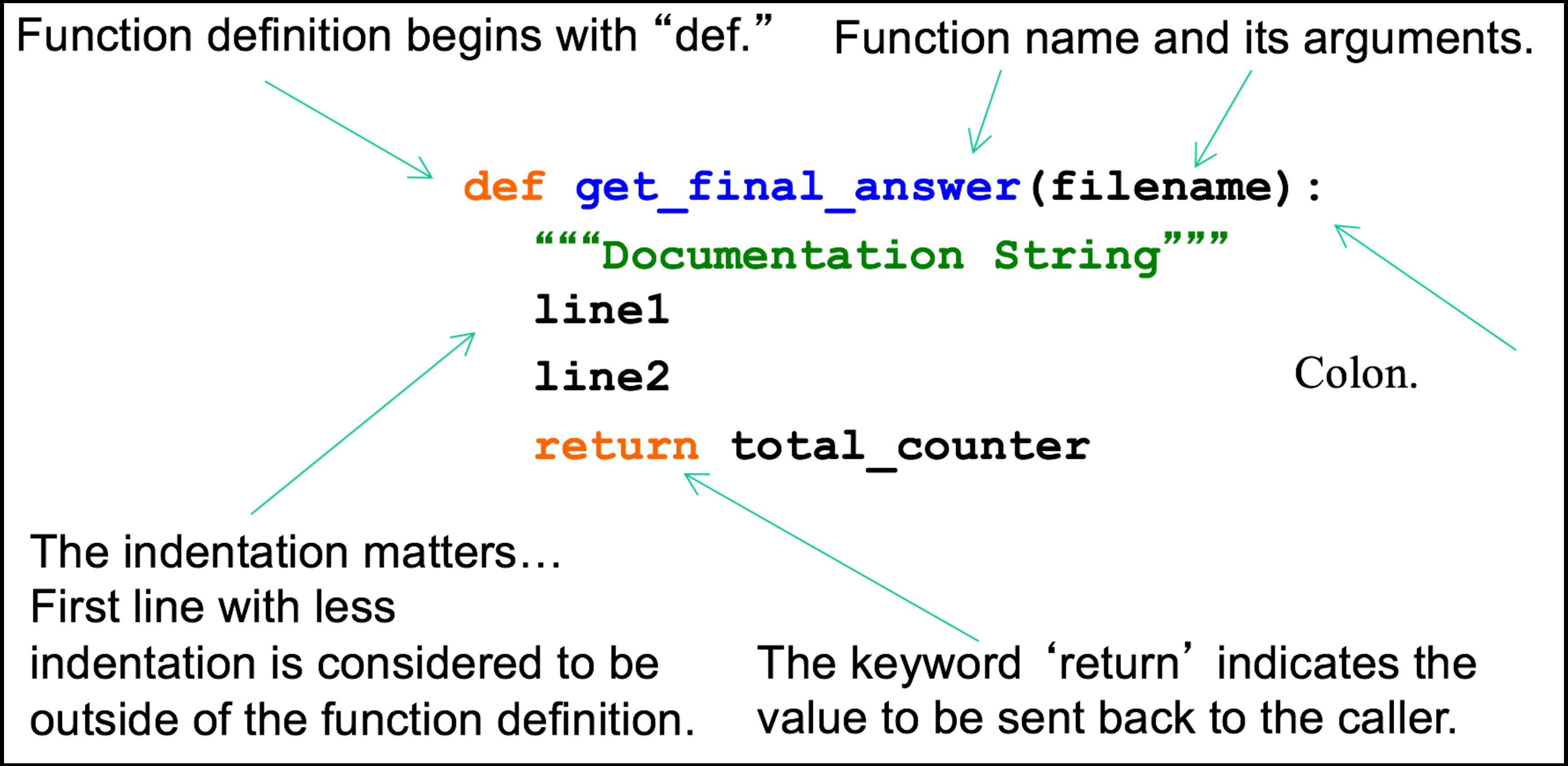
DEFINING FUNCTIONS

- Here are simple rules to define a function in Python:
    - Function blocks begin with the keyword **def** followed by the function **name** and parentheses **( )**.
    - Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
    - The code block within every function starts with a colon : and is indented.
    - The statement return [expression] exits a function, optionally passing back an expression to the caller. A **return** statement with no arguments is the same as return None.

```
>>> def my_func(x, y, z):
...     a = x + y
...     b = a * z
...     return b
...
>>>
```

```
>>> my_func(1.0, 3.0, 2.0)
8.0
>>> my_func(1.0, 3.0, 1.0)
4.0
>>> my_func(5.0, 0.0, 1.0)
5.0
>>> my_func(2.0, 0,0 3.0)
6.0
```

DEFINING A FUNCTION



Function definition begins with "def."

Function name and its arguments.

```
def get_final_answer(filename):
    """Documentation String"""
    line1
    line2
    return total_counter
```

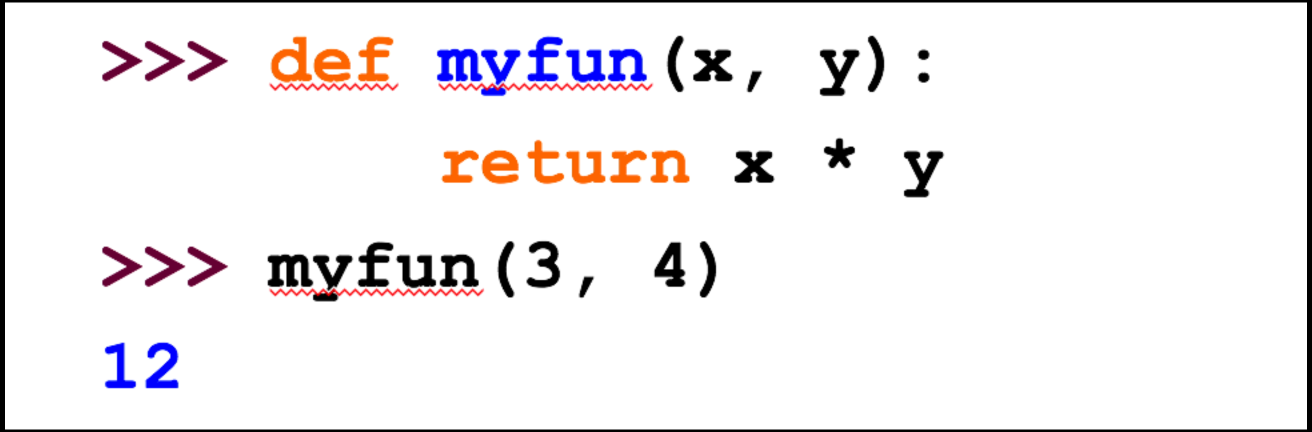
Colon.

The indentation matters…
First line with less
indentation is considered to be
outside of the function definition.

The keyword 'return' indicates the
value to be sent back to the caller.

CALLING A FUNCTION

- Parameters in Python are Call by Assignment
- Old values for the variables that are parameter names are hidden, and these variables are simply made to refer to the new values
- All assignment in Python, including binding function parameters, uses reference semantics.



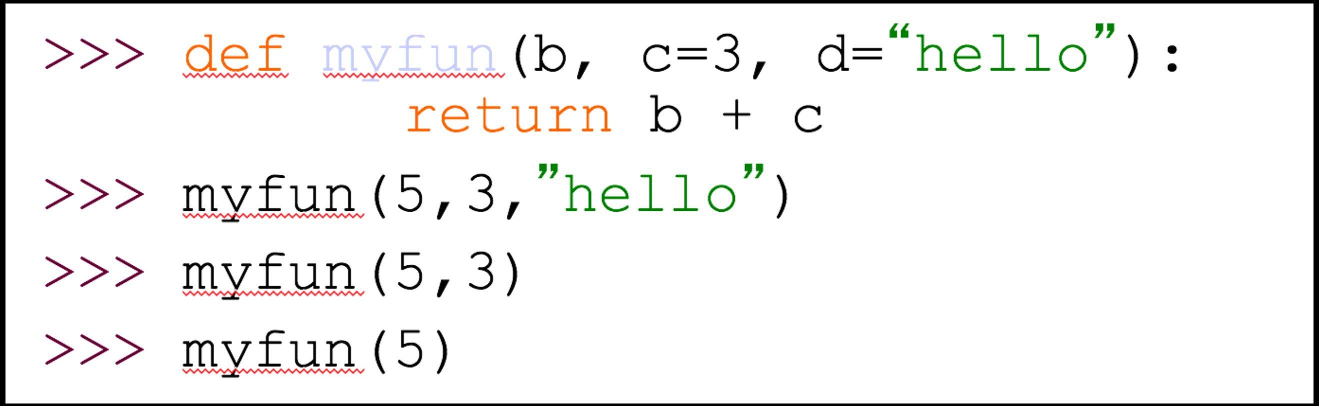
```
>>> def myfun(x, y):
        return x * y
>>> myfun(3, 4)
12
```

FUNCTIONS WITHOUT RETURNS

- All functions in Python have a return value, even if no return line inside the code
- Functions without a return return the special value None
- None is a special constant in the language
- None is used like NULL, void, or nil in other languages
- None is also logically equivalent to False
- The interpreter’s REPL doesn’t print None

DEFAULT VALUES FOR ARGUMENTS

- You can provide default values for a function’s arguments
- These arguments are optional when the function is called



```
>>> def myfun(b, c=3, d=“hello”):
            return b + c
>>> myfun(5,3,”hello”)
>>> myfun(5,3)
>>> myfun(5)
```

- All of the above function calls return 8

HANDS ON: FUNCTIONS

- Let's see a modified form of our Yahtzee example. In this case, we will make a function for rolling the dice.
- See yahtzee3.py. Let's look at the function first.

```
import random
def roll_dice(num_dice=5):
    """Roll n 6-sided dice and return as a list"""
    dice = [0,0,0,0,0]
    print("Rolling the dice...")
    for i in range(0, num_dice):
        dice[i] = random.randint(1, 6)
    print("You rolled the following dice:")
    dice.sort()
    return dice
```

HANDS ON: FUNCTIONS

```
print("Welcome to the Yahtzee dice roller!")
dice_rolled = []
roll_number = 1
repeat = 'y'
while (repeat == 'y') or (repeat == 'Y'):
    dice = roll_dice()   ### <<<< Here is the function!
    print(dice)
    print()
    dice_rolled.append((roll_number, dice))
    repeat = input("Would you like to roll again? [Y/n]: ")
    print()
    roll_number = roll_number + 1
print("Dice rolled:\n{0}".format(dice_rolled))
print("bye!")
```

HANDS ON: FUNCTIONS

```
$ python3 ./yahtzee3.py
Welcome to the Yahtzee dice roller!
Rolling the dice...
You rolled the following dice:
[2, 3, 3, 3, 6]

Would you like to roll again? [Y/n]: y

Rolling the dice...
You rolled the following dice:
[2, 4, 6, 6, 6]

Would you like to roll again? [Y/n]: y

...

Dice rolled:
[(1, [2, 3, 3, 3, 6]), (2, [2, 4, 6, 6, 6]), (3, [1, 1, 4, 5, 6]), (4, [1, 3, 4, 5, 6])]
bye!
```

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 #6 - REVIEW

```
import json

print("Welcome to the rolodex simulator (v1)!")
myRolodex = []
repeat = 'y'
while (repeat == 'y') or (repeat == 'Y'):
    first, last = input("First and last name (use space): ").split()
    street = input("Street: ")
    city, state, zipcode = input("City, state, and zip (use commas): ").split(",")
    more_phones = 'y'
    phones = []
    while (more_phones == 'y') or (more_phones == 'Y'):
        phone = input("Enter a phone number (XXX) XXX-XXXX: ").strip()
        phones.append(phone)
        more_phones = input("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
    }
    myRolodex.append(record)
    print()
    repeat = input("Enter another record? [Y/N] ")
    print()
…
```

HOMEWORK #6 - REVIEW

```
…
# Print out the data
print("Contents of the rolodex (default):")
for record in myRolodex:
    print(record)
print()
print("Contents of the rolodex (JSON):")
for record in myRolodex:
    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("bye!")
```

HOMEWORK #6 - REVIEW

```
$ python3 ./rolodex1.py
Welcome to the rolodex simulator (v1)!
First and last name (use space): Will Smith
Street: 123 Main Street
City, state, and zip (use commas): Florence, FL, 44001
Enter a phone number (XXX) XXX-XXXX: (800) 555-1212
Enter another phone? [Y/N] y
Enter a phone number (XXX) XXX-XXXX: (866) 123-4567
Enter another phone? [Y/N] n

Enter another record? [Y/N] y

First and last name (use space): Anne Arbor
Street: 45 West Landing
City, state, and zip (use commas): Greeneville, NC, 33009
Enter a phone number (XXX) XXX-XXXX: (252) 534-2301
Enter another phone? [Y/N] n

Enter another record? [Y/N] n
...
```

HOMEWORK #6 - REVIEW

```
...
Contents of the rolodex (default):
{'first': 'Will', 'last': 'Smith', 'address': {'street': '123 Main Street', 'city': 'Florence',
'state': 'FL', 'zip': '44001'}, 'phones': ['(800) 555-1212', '(866) 123-4567']}
{'first': 'Anne', 'last': 'Arbor', 'address': {'street': '45 West Landing', 'city': 'Greeneville',
'state': 'NC', 'zip': '33009'}, 'phones': ['(252) 534-2301']}

Contents of the rolodex (JSON):
{
    "address": {
        "city": "Florence",
        "state": "FL",
        "street": "123 Main Street",
        "zip": "44001"
    },
    "first": "Will",
    "last": "Smith",
    "phones": [
        "(800) 555-1212",
        "(866) 123-4567"
    ]
}
...
```

HOMEWORK #6 - REVIEW

```
...
{
    "address": {
        "city": "Greeneville",
        "state": "NC",
        "street": "45 West Landing",
        "zip": "33009"
    },
    "first": "Anne",
    "last": "Arbor",
    "phones": [
        "(252) 534-2301"
    ]
}

Alphabetical list of names:
Arbor, Anne
Smith, Will
bye!
```

HOMEWORK ASSIGNMENT #7

- (2 points) Create a script named 'rolodex2.py' that revises (called refactoring) the rolodex assignment at least one function:
  - **`get_record()`** – gets a record (dictionary) from the user
- (2 bonus points) Isolate the input() call to a single function:
  - **`get_string(prompt)`** – pass in the prompt and return a string from the user. Hint: there should be exactly (1) call to input() in the resulting code.
- (1 bonus point) Use a function to print the rolodex:
  - **`print_rolodex(rolodex)`** – print the contents of the rolodex.

HOMEWORK ASSIGNMENT #7

- BIG HINT: The main source code should look like this (with bonus challenges):

```
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!")
```

QUESTIONS OR COMMENTS?