# Parables and Pythons

Dr. Charles "Chuck" Bell

Lesson 6: 17 October 2018

# CLASS AGENDA

- Bible Study
  - The Sower and the Seed (a story of four Christians)
- The Golden Age of British Comedy
  - Holy Hand Grenade
- Computer Programming with Python
  - Review:
    - While loop
    - Lists
  - Hands-On Practice
    - Lists and sets
- Homework

# BIBLE STUDY: THE PARABLES OF JESUS

## THE SOWER AND THE SEED LUKE 8:5–8

*"A farmer went out to sow his seed. As he was scattering the seed, some fell along the path; it was trampled on, and the birds ate it up. Some fell on rocky ground, and when it came up, the plants withered because they had no moisture. Other seed fell among thorns, which grew up with it and choked the plants. Still other seed fell on good soil. It came up and yielded a crop, a hundred times more than was sown." When he said this, he called out, "Whoever has ears to hear, let them hear."*

## WHO ARE THE FOUR CHRISTIANS?

1. Those that hear but don’t discern
   - Allow Satan to take their eyes off of God easily
   - May even join the church, but any small thing turns they away
2. Those that hear and understand but turn away after a time
   - Make a great start on their way to sanctification, but eventually wander away
3. Those that hear but allow distractions to hold them back from sanctification
   - Events and things in their lives become a priority over God
4. Those that hear, discern, and prosper in the Word of God
   - Those on the path and eventually reach sanctification

## SO, ARE (1)-(3) REALLY CHRISTIANS?

- What do you think?
  - Those that hear but don’t discern
  - Those that hear and understand but turn away after a time
  - Those that hear but allow distractions to hold them back from sanctification
  - Those that hear, discern, and prosper in the Word of God

## HOW DOES THE PARABLE EXPLAIN THE FOUR CHRISTIANS?

- Jesus used the sower, the seed, and the soil to teach about mankind’s reception of God’s Word. There are many who hear God’s Word, but very few live it. Some accept it for a short time, only to allow sin and self to choke out the Word.
- Interestingly, this is one of Jesus’s parables that He had to explain so that people would understand.
- Let’s look at his explanation then take it apart.

## THE SOWER AND THE SEED LUKE 8:9-15

*His disciples asked him what this parable meant. He said, “The knowledge of the secrets of the kingdom of God has been given to you, but to others I speak in parables, so that, “‘though seeing, they may not see; though hearing, they may not understand.’ “This is the meaning of the parable: The seed is the word of God. Those along the path are the ones who hear, and then the devil comes and takes away the word from their hearts, so that they may not believe and be saved.*

## THE SOWER AND THE SEED LUKE 8:9-15

*Those on the rocky ground are the ones who receive the word with joy when they hear it, but they have no root. They believe for a while, but in the time of testing they fall away. The seed that fell among thorns stands for those who hear, but as they go on their way they are choked by life's worries, riches and pleasures, and they do not mature. But the seed on good soil stands for those with a noble and good heart, who hear the word, retain it, and by persevering produce a crop.*

## CONCLUSIONS

- God wants all Christians to be like the seed falling upon the good ground. His Word will:
  - Establish us (I Peter 3:15)
  - Instruct us (II Tim. 2:15)
  - Cleanse us (Ps. 119:9)
  - Keep us from sin (Ps. 119:11)
- Allow His Word to grow within you, to be an active part of your daily life (James 1:22).
- **Most importantly**: allow the seed to grow on the fertile ground of a mind focused on Christ devoted completely to serving Him above all others. Do this and nothing with wither or be choked by thorns.

## THE GOLDEN AGE OF BRITISH COMEDY

## LYRICS

| Latin | English |
|---|---|
| Pie Iesu domine, dona eis requiem. | Gentle Lord Jesus, grant them rest. |
| Pie Iesu domine, dona eis requiem. | Gentle Lord Jesus, grant them rest. |
| Pie Iesu domine, dona eis requiem. | Gentle Lord Jesus, grant them rest. |
| Pie Iesu domine, dona eis requiem. | Gentle Lord Jesus, grant them rest. |

- https://www.youtube.com/watch?v=ashgP4YMdJw

## COMPUTER PROGRAMMING

Hands On Learning

## WHILE LOOP - REVIEW

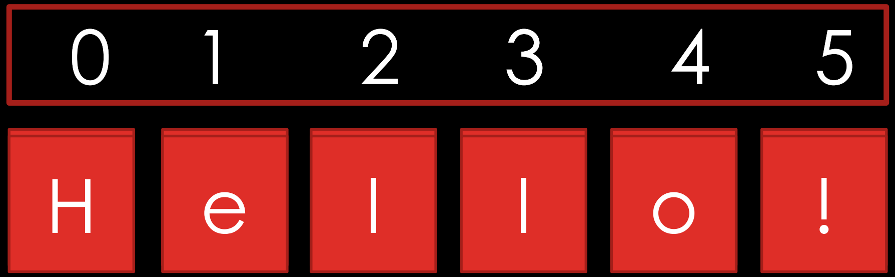
- Another type of loop is the while loop.
- The loop uses a condition such that as long as the expression evaluates to True, it will execute the body (code block) of the loop.
- The while loop goes like this:

```
while <expression>:
    <code to execute>
```

- The expression can be any combination of inequalities and conditionals.

## 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.



## LIST FUNCTIONS – COUNTING

- `len(myList)`: number of elements in list (top level).
- `min(myList)`: smallest element. Must all be the same type!
- `max(myList)`: largest element, again all must be the same type
- `sum(myList)`: sum of the elements, numeric only

## LIST FUNCTIONS – HANDS ON

- Open the file “lists2.py” and observe the code.
  - Or, open a new file and type this code.

```
# Working with lists - part 2
myList = [123, 800, 0.25]
print("The length of the list is: {0}".format(len(myList)))
print("The minimum value is: {0}".format(min(myList)))
print("The maximum value is: {0}".format(max(myList)))
print("The sum of the values in the list is: {0}".format(sum(myList)))
```

## LIST FUNCTIONS – RESULTS

- You should see the following output.

```
$ python3 ./lists2.py
The length of the list is: 3
The minimum value is: 0.25
The maximum value is: 800
The sum of the values in the list is: 923.25
```

## LISTS ARE MUTABLE

- Lists are mutable. You can change the object's contents!

```
my_list = [1, 2, 3]
my_list[0] = 127
print(my_list) ⇒ [127, 2, 3]
```

## LIST METHODS

- Remember, a function is a small program (such as len) that takes some arguments, the stuff in the parenthesis, and returns some value
- a method is a function called in a special way, the ***dot call***. It is called in the context of an object (or a variable associated with an object)

## AGAIN, LISTS HAVE METHODS

```
my_list = ['a',1,True]
my_list.append('z')
```

the object that
we are calling the
method with

the name of
the method

arguments to
the method

## SOME NEW METHODS

- A list is mutable and can change:

```
my_list[0]='a'  #index assignment
my_list.append(), my_list.extend()
my_list.pop()
my_list.insert(), my_list.remove()
my_list.sort()
my_list.reverse()
```

- Most of these methods ***do not return a value***
- This is because lists are mutable, so the methods modify the list directly. No need to return anything.

## UNUSUAL RESULTS

```
my_list = [4, 7, 1, 2]
my_list = my_list.sort()
my_list ⇒ None      # what happened?
```

What happened was the sort operation changed the order of the list in place (right side of assignment). Then the sort method returned `None`, which was assigned to the variable. The list was lost and `None` is now the value of the variable.

## SPLIT

- The string method split generates a sequence of characters by splitting the string at certain split-characters.
- You can specify whatever character you want (space is default).
- ***It returns a list*** (we didn't mention that before)

```
split_list = 'this is a test'.split()
print(split_list)
['this', 'is', 'a', 'test']
```

## SORTING

Only lists have a built in sorting method. Thus you often convert your data to a list if it needs sorting

```
my_list = list('xyzabc')
print(my_list)
['x','y','z','a','b','c']
my_list.sort()    # no return
print(my_list)
['a', 'b', 'c', 'x', 'y', 'z']
```

## LISTS – HANDS ON

- Open the script, “lists3.py” and observe the code.
  - Or, open a new file and type the code.

```
# Working with lists - part 3
done = False
myList = []
while not done:
    value = input("Please enter a value for the list or "
                  "press ENTER to end the list: ")
    if value:
        myList.append(value)
    else:
        done = True
print("The unsorted list: {0}".format(myList))
myList.sort()
print("The sorted list: {0}".format(myList))
value = input("Enter a value to add to the front of the list: ")
myList.insert(0, value)
print("Updated list: {0}".format(myList))
```

## LISTS – HANDS ON RESULTS

- You should see something simliar to the following when you run the script.

```
$ python3 ./lists3.py
Please enter a value for the list or press ENTER to end the list: 123
Please enter a value for the list or press ENTER to end the list: 555
Please enter a value for the list or press ENTER to end the list: 432
Please enter a value for the list or press ENTER to end the list:
The unsorted list: ['123', '555', '432']
The sorted list: ['123', '432', '555']
Enter a value to add to the front of the list: 33
Updated list: ['33', '123', '432', '555']
```

## 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 #4 - REVIEW

- This is the correct solution to the modified dice simulator assignment:

```
# Yahtzee dice roller example
import random
print("Welcome to the Yahtzee dice roller!")
dice = [0,0,0,0,0]
repeat = 'y'
while (repeat == 'y') or (repeat == 'Y'):
    print("Rolling the dice...")
    for i in range(0, 5):
        dice[i] = random.randint(1, 6)
    print("You rolled the following dice:")
    dice.sort()
    print(dice)
    print()
    repeat = input("Would you like to roll again? [Y/n]: ")
    print()
print("bye!")
```

## HOMEWORK #3 - REVIEW

```
$ python3 ./yahtzee1.py
Welcome to the Yahtzee dice roller!
Rolling the dice...
You rolled the following dice:
[1, 2, 3, 4, 6]

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

Rolling the dice...
You rolled the following dice:
[1, 1, 4, 4, 5]

Would you like to roll again? [Y/n]: n
bye!
```

## HOMEWORK ASSIGNMENT #5

- (2 points) Modify the yahtzee1.py script (rename to yahtzee2.py) to store each set of rolls in a list inserting the last roll at the front of the list (this is called a stack – a fundamental data structure in computer science - LIFO)
- (2 bonus points) Change the stack to a queue.
    - Hint: a queue is a FIFO data structure.
- (1 bonus point) Store the roll number with the dice rolls as a tuple.
    - Hint: a tuple is formed with parenthesis and values separated by commas.

```
myTuple = (3, [1,2,3])    # a tuple with one integer and a list
```

## QUESTIONS OR COMMENTS?