ENGL 516X: Methods of Formal Linguistic Analysis

Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

1 Feb 2018

Class outline

- Program from Tuesday + Questions
- Some new stuff about the topics we discussed so far
- Practice exercises
- Reminder: Assignment 2 due on 3rd! Submit on time!

Tuesday's programming Exercise

- Ask the user to enter a number first (integer). Assign it to a variable n.
- Now, take input from the user n number of times after this. These have to be numbers.
- Once the input taking is done, you have print the following back to the user: sum of the these numbers, and average.
- Example interaction with your program:

```
> Enter the number of numbers you want to enter:
5
> Enter a number: 2
> Enter a number: 6
> Enter a number: 5
> Enter a number: 3
> Enter a number: 8
> The sum of these numbers is: 24
> The average of these numbers is: 4.8
```

- Assume for now that the user is following your directions, and there are no errors to handle.

Tuesday's problem: Extension

Add exception handling to this program, to address the following conditions:

- n is a integer between 2 and 100
- Each subsequent number is a integer in the range 0 to 10000
- If the user enters a string or floating point or any non-integer for n, print an error message using try and except and stop.
- If the user enters anything other than a number after n, detect their mistake using if and else, and print an error message and move on to take next input number.
- Note: This is Similar to Final exercise in Chapter 5 in the textbook
- Note 2: I am not asking you to organize this program into functions - but think if you can.

New Solution

ExtendLoopQuestion.py

Topics covered so far

- Basic building blocks of Python programming: variables, expressions, operators
- Conditional statements
- data types, converting between them
- Exception handling (try, except)
- Writing our own functions
- Loops (for, while)
- Breaking a loop execution: break and continue statements

Some new stuff within these topics

main() function

- We can define an optional function with name main() in Python.
- good programming practice
- You just name it main - rest is same as other functions.
- We use it primarily to bring a logical structure to your program
- This kind of function is mandatory in some other languages, and program execution starts at main() function.

more info in the second textbook: `https://goo.gl/rbhDcd`

main() function - somewhat advanced stuff

- Python has a internal variable called __name__, which is automatically set to the string __main__ When we run the program just by itself

main() function - somewhat advanced stuff

- Python has a internal variable called __name__, which is automatically set to the string __main__ When we run the program just by itself
- It is also possible to "import" one program into another. In such a case, __name__ is set to the name of that program.
- Typically, we add this in the program:

  ```
  if __name__ == "__main__":
      main()
  ```

  -to tell it to look for the main() function and start running from there, when we execute the program.

Factorial program with main() function

```
def factorial(n):
   fact = 1
   for i in range(1,n+1): #why not start at 0??
      fact = fact*i
   return fact

def main():
   num = int(input("Enter a number: "))
   print(factorial(num))

if __name__ == "__main__":
    main()
```

A recursive program

- A recursive function is something that calls itself in its definition.
- How can a function call itself? See this example below:
-
```
def factorial(number):
    if number == 0 or number == 1:
        return number
    else:
        return number*factorial(number-1)
        #What is happening????
print(factorial(3))
print(factorial(1))
print(factorial(2))
```

A recursive program

- A recursive function is something that calls itself in its definition.
- How can a function call itself? See this example below:

-
  ```
  def factorial(number):
      if number == 0 or number == 1:
          return number
      else:
          return number*factorial(number-1)
          #What is happening????
  print(factorial(3))
  print(factorial(1))
  print(factorial(2))
  ```
- It is a way of programming. For every recursive program, there is always a non-recursive version.

Revision - some code analysis and some coding practice

functions vs methods

For the question about isxxx() methods posted in the forum

- We use len(”python”) but ”python”.isalpha() - what is the difference?
- The first one is called a ”function”, second one is a ”method” that works for strings.

functions vs methods

For the question about isxxx() methods posted in the forum

- We use len("python") but "python".isalpha() - what is the difference?
- The first one is called a "function", second one is a "method" that works for strings.
- simple difference: functions - may work with several kinds of data types (e.g., print() works with integers, strings, floats, lists etc).
- methods are tied to specific data objects (i.e., .isalpha() works only for string variables.

functions vs methods

For the question about isxxx() methods posted in the forum

- We use len("python") but "python".isalpha() - what is the difference?
- The first one is called a "function", second one is a "method" that works for strings.
- simple difference: functions - may work with several kinds of data types (e.g., print() works with integers, strings, floats, lists etc).
- methods are tied to specific data objects (i.e., .isalpha() works only for string variables.
- complex difference: There is something called "object oriented programming" - which is beyond the scope of this class.

comments in python

- # are used to write single line comments in your program.
- Anything after that symbol will be ignored by python interpreter.
- they are for our own use - commenting a program is a good practice both for you, and anyone who wants to use your program.

comments in python

- # are used to write single line comments in your program.
- Anything after that symbol will be ignored by python interpreter.
- they are for our own use - commenting a program is a good practice both for you, and anyone who wants to use your program.
- Multi-line comments start and end with triple quotes (single or double)

Exercise: Write a program

Write a program that generates 10 random integers between 1 and 1000, and prints the sum of these 10 numbers. What happens if you run again? Do you see the same result?

Exercise: Write a program

Write a program that generates 10 random integers between 1 and 1000, and prints the sum of these 10 numbers. What happens if you run again? Do you see the same result?

```
import random
i = 1
sum = 0
while i<=10:
    randNum = random.randint(1,1000)
    print(randNum)
    sum += randNum
    i = i+1
print("Sum of all the 10 generated numbers so far is: " + str(sum))
```

Exercise: Write another program

Write a program that prompts the user to keep entering strings. It should stop when the user enters "done" or something and print the lengths of the longest and shortest strings entered so far.

Exercise: Write another program

Write a program that prompts the user to keep entering strings. It should stop when the user enters "done" or something and print the lengths of the longest and shortest strings entered so far.

```
minlength = 9999999999
maxlength = 0
try:
   while True:
     inputString = input("Enter a string: ")
     if inputString == "done":
         print("Min length of strings you entered so far: " + str(minlength))
         print("Max length of strings you entered so far: " + str(maxlength))
         break
     lenString  = len(inputString)
     if lenString < minlength:
         minlength = lenString
     if lenString >= maxlength:
         maxlength = lenString

except Exception as E:
    print("Something really unpredictable happened! Here is the description:")
    print(E)
```

Exercise: Write a program with functions

Write a program with the following functions:

- OddEven(integer): This function takes a positive whole number as an argument, and returns a string which is either "Odd" or "Even".
- LogNum(integer): Takes a positive whole number and returns the logarithm of this number.
- RandNum(integer): Takes a positive whole number and returns a random number between 0 and this number.
- main(): A main function, that prompts a user for a number, and returns the output of all the above functions one by one.
- Make sure your program actually runs!
- Note: program should ask for input only once, and give that number as argument to all functions!

Post your solution on the forum.

Last class' additional exercise

- Write a program that takes a number and prints multiplication table for that number (n*1 to n*10, one number per line)
- Expected input/output:

```
> Enter a number: 5
> 5*1 = 5
  5*2 = 10
  5*3 = 15
  .. ...
  5*10 = 50
```

- post solution in today's discussion forum.

Is this a familiar feeling now?

Next Week

- Topics: Strings, String manipulations, Regular expressions
- Readings:
  - For Tuesday: Chapter 6; `https://goo.gl/DU4aSQ`
  - For Thursday: Chapter 11.
- Optional reading: "The Joys (and Woes) of the Craft of Programming" by Frederick P.Brooks `http://home.adelphi.edu/sbloch/class/adages/joy.html`
- Optional exercise: Do the two exercises at the end of Chapter 5 in the textbook.
- Mandatory: Submit Assignment 2.