413-513 Notes- Week 3

**[ String formatting docs ]**

<https://docs.python.org/2.5/lib/typesseq-strings.html>

**[ Loops ]**

uses:

-salary (use of forever loop, use of **break** statement)

-prime num tester (use of loop)

* **The range function simplifies the process of writing a for loop**
  + range returns an iterable object
    - Iterable: contains a sequence of values that can be iterated over
* **range characteristics:**
  + One argument: used as ending limit
  + Two arguments: starting value up to but *not* including ending limit
  + Three arguments: third argument is step value

ex. **range (10, 0, -1)**

Fixed loop example --*For loop from 0 to 2, therefore running 3 times.*

for x in range(0, 3):

print "We're on time %d" % (x)

*While loop from 1 to infinity, therefore running infinity times.*

x = 1

while True:

print ("To infinity and beyond! We're getting close, \

on %d now!" % (x))

x += 1

#salary problem

Sammy begins a new job with an annual salary of $ 52,500 for his first year of employment, a guarantee of a 5 % salary increase each for his second, third and

fourth year and a guarantee of a 9.5 % increase for his fifth and sixth year. Write a code that will output Sammy’s salary schedule for his first six years of employment. Round all amounts to two decimal places. Write your program such that your output will appear similar to the tabular format shown as follows….

|  |  |
| --- | --- |
| **Year** | **Salary** |
| 1 | $ **52500.00** |
| 2 | $ |
| 3 | $ |
| 4 | $ |
| 5 | $ |
| 6 | $ |

'''salary problem'''

#housekeeping - declare & initialize vars

salary = 52500

#print iniital header

print ("\t", "Year".ljust(16), "Salary".ljust(16))

#print 1st year salary

print ("\t", "1".center(3).ljust(16), "${:,.2f}".format(salary).ljust(16))

#set loop range to years (2-6) of the salary increases

for x in range(2,7):

if x < 5:

salary \*= 1.05

print ("\t", str(x).center(3).ljust(16), "${:,.2f}".format(salary).ljust(16))

else:

salary \*= 1.095

print ("\t", str(x).center(3).ljust(16), "${:,.2f}".format(salary).ljust(16))

**[ Handling programming errors ]**

Ex. input validation, check for errors –

ex. password incorrect allow for 3 tries, etc.

Problem: (Registration Input verification checker) program.

a. Using various 'built-in' string methods, write code to check a user input for a

registration form.

<http://www.codeproject.com/Articles/2782/Credit-Card-Validator-control-for-ASP-NET>

Verify a persons:

username (email address?)

password (at least 6 digits, mask password?)

address (city,state,zip (digits only))

b. Include user defined methods as well to validate input data.

c. Error trap any erroneous input data.

Error trap suggestions ~meeting specialty criteria

-General checking!

1. input is not null (for ANY input)

-Numeric checking

2. input is a number

-Character length checking

3. check input len (password, zip)

#Number validator

**while True:** #use as do loop logic, need break to quit

try:

**x = int(input("Please enter a number: "))**

**break**

**except ValueError:**

**print "Oops! That was no valid number. Try again..."**

**[ Functions – intro ]**

Builtin – math, random num generators, etc.

Keywords passed (given in line after [positional args])

def counter(start=0, step=1)

Block variables , local v. scope level vs global variables/constants

Random num generation

Use cases:

**-One line or multi lines at bank?**

**-Card deck shuffle**

<https://docs.python.org/3/library/random.html>

Difference between randrange and randint

**randrange([start], stop[, step])** you can use the step

random.randrange(0,1) will **not** consider the last item

**randint(0,1)** returns a choice **inclusive** of the last item.

**import random**

**for x in range(10):**

**x = random.randrange(1,5)**

**#x = random.randint(1,5)**

**print (x)**

-uniform ex.

**for i in range(5):**

**print ('%04.3f' % random.uniform(1, 100))**

The method **seed()** sets the integer starting value used in generating random numbers. Call this function before calling any other random module function.

random.seed(10)

random.seed(1)

**for** i **in** range(5):

**print** ('**%04.3f**' % random.random())

**import random**

**#gen keygen**

**strv="" #must define**

**for x in range(1,26):**

**strv += chr(random.randint(65,90)) #convert to chars**

**if x % 5 == 0 and x != 25:**

**strv += '-' #sep value pairs by 5, excl. last set**

**print (strv)**

**#bonus convert letter to num**

**x = ord('A')**

**print (x)**

>>>

**UWFIB-SKUNF-ENOQU-TJOAT-TFWQQ**

**65**

>>>

**Next week solid coverage**:

creating modules

function overloading / argument passing args, kwargs

multiple returns on functions

lambda functions (one liners!)

Recursive functions