## Python Exercises

## CS 361: Programming Languages Principles and Implementations

## Weili Zhong

December 17, 2018

**Python**

**Exercises**

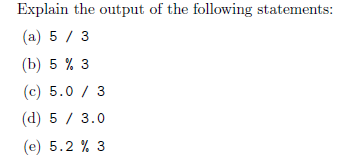
The code must be submitted under your name in GitHub in a repository called Python. Work individually.

Create one file with all your work and name it: cs361python.py or cs 631python.py.

Do not commit code that does not compile. The code that you commit should have been tested. -10 points for each exercise for code that does not compile on the top of your grade.

You will provide a hardcopy with your code to Dr. Scharff on 12/17.

**Exercise 1**



print("(a). 5 / 3 =", 5 / 3)

print("Prints out the exact value of 5 / 3 \n")

print("(b) 5 % 3 =", 5 % 3)

print("Prints out the remainder of 5 / 3 \n")

print("(c) 5.0 / 3 =", 5.0 / 3)

print("Prints out the exact value of 5.0 / 3, shows you can perform operations with different data types \n")

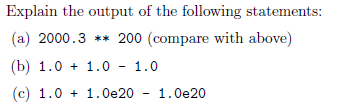
print("(d) 5 / 3.0 =", 5 / 3.0)

print("Prints out the exact value of 5 / 3.0, shows you can perform operations with different data types \n")

print("(e) 5.2 % 3 =", 5.2 % 3)

print("Prints out the exact remainder of 5.2 / 3, shows you can perform operations with different data types \n")

**Exercise 2**



try:

print("(a) 2000.3 \*\* 200=", 2000.3 \*\* 200)

except OverflowError:

print("Causes OverFlowError, the result is too large \n")

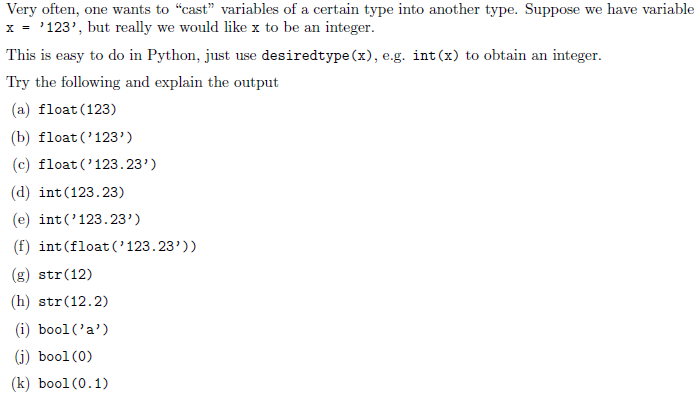
print("(b) 1.0 + 1.0 - 1.0=", 1.0 + 1.0 - 1.0)

print("Prints out the exact value of 1.0 + 1.0 - 1.0 \n")

print("(c) 1.0 + 1.0e20 - 1.0e20=", 1.0 + 1.0e20 - 1.0e20)

print("Prints out 0 \n")

**Exercise 3**



1. 123.0: casts 123 to a float
2. 123.0: casts string 123 to float
3. 123.23: casts string 123.23 to float
4. 123: casts 123.23 to an int
5. 123: Error, passes a string which is 123.23 (float) to be cast as int
6. 123.23: casts string 123.23 to a float first then float 123.23 to int
7. 12: cast int 12 to string
8. 12.2: cast float 12.2 to string
9. True: casts string a to true
10. False: int 0 casted into a bool is false
11. True: anything other than 0 is true

print("(a)", float(123))

print("(b)", float('123'))

print("(c)", float('123.23'))

print("(d)", int(123.23))

try:

print("(e)", int('123.23'))

except ValueError:

print("(e) Value Error, passing a float as string to be cast to int")

print("(f)", int(float('123.23')))

print("(g)", str(12))

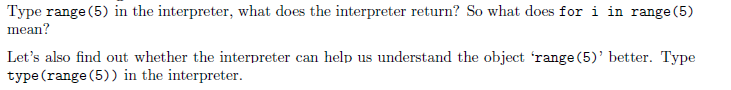
print("(h)", str(12.2))

print("(i)", bool('a'))

print("(j)", bool(0))

print("(k)", bool(0.1))

**Exercise 4**



for i in range(5) means any number from 0 to 5

type(range(5)) produces class 'range' which tells you range() is of class 'range'

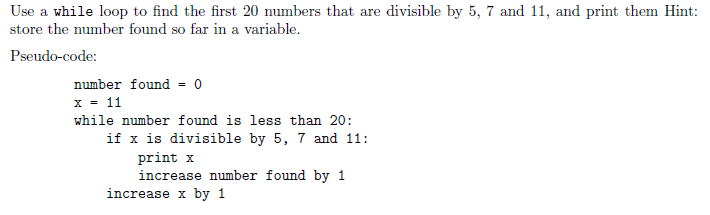
print(range(5))

print("for i in range(5) means any number from 0 to 5")

print(type(range(5)))

print("type(range(5)) produces class 'range' which tells you range() is of class 'range'")

**Exercise 5**



def primeNumber():

number\_found = 0

x = 11

while number\_found < 20:

test\_5 = (x % 5 == 0)

test\_7 = (x % 7 == 0)

test\_11 = (x % 11 == 0)

if test\_5 != 0 and test\_7 != 0 and test\_11 != 0:

print(x)

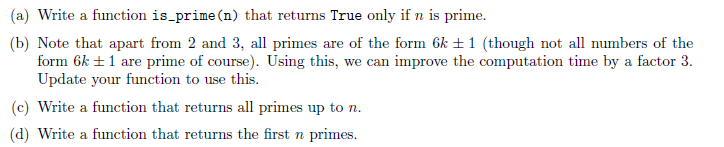
number\_found += 1

x += 1

print("The first 20 prime numbers \n")

primeNumber()

**Exercise 6**



#-----------Part A-----------

def is\_prime(n):

if n == 2:

return True

if n == 3:

return True

if n % 2 == 0:

return False

if n % 3 == 0:

return False

i = 5

w = 2

while i \* i < n:

if n % i == 0:

return False

i += w

w = 6 - w

return True

#-----------Part C-----------

def primes\_up\_to(n):

counter = 2

list\_of\_primes = []

while counter <= n:

if is\_prime(counter):

list\_of\_primes.append(counter)

counter += 1

return(list\_of\_primes)

#-----------Part D-----------

def first\_n\_primes(n):

num\_primes = n

counter = 2

list\_of\_primes = []

while num\_primes > 0:

if is\_prime(counter):

num\_primes = num\_primes - 1

list\_of\_primes.append(counter)

counter += 1

return(list\_of\_primes)

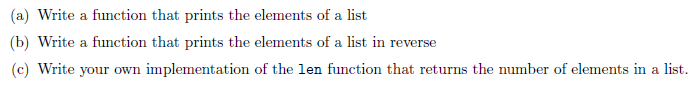
#-----------Tests-----------

print("Check if 230 is prime:", is\_prime(230))

print("Prints all primes up to 100:", primes\_up\_to(100))

print("Prints first 50 primes", first\_n\_primes(50))

**Exercise 7**



print("Exercise 7 \n")

#-----------Part A-----------

def printList(list):

iterator = 0

while iterator < len(list):

print(list[iterator])

iterator += 1

print("Finished Print List")

#-----------Part B-----------

def reverseprintList(list):

iterator = len(list) - 1

while iterator >= 0:

print(list[iterator])

iterator = iterator - 1

print("Finished Reverse Print")

#-----------Part C-----------

def countElements(list):

counter = 0;

while counter < len(list):

counter = counter + 1

return print("List Element Count", counter)

#-----------Tests-----------

list = [2,3,5,123,4,1,23]

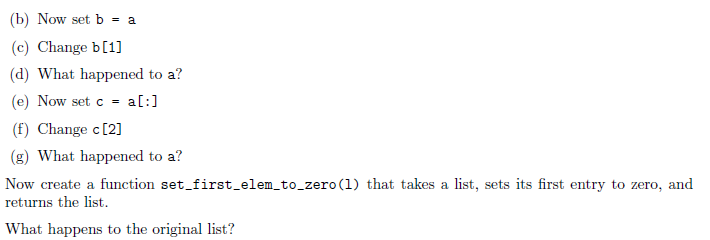
printList(list)

reverseprintList(list)

countElements(list)

**Exercise 8**





a = [5, 10, 15, 20, 25]

print("(a)", a, "Created a list")

print("(b) Set b = a" )

b = a

print("(c) Set b[1]")

b[1] = 88

print("(d)", a, "Changes made in list b are also made in list a")

print("(e) Set c = a[:]")

c = a[:]

print("(f) Set c[2]")

c[2] = 77

print("(g)", a, "Nothing has happened to a after changes in list c")

def set\_first\_elem\_to\_zero(list):

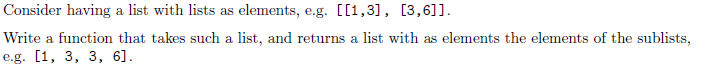
list[0] = 0

return 1

set\_first\_elem\_to\_zero(a)

print("After running set\_first\_elem\_to\_zero on list a the first element changes to 0")

**Exercise 9**



def sublist\_elements(list):

final\_list = []

iterator = 0

while iterator < len(list):

counter = 0

sublist = list[iterator]

while counter < len(sublist):

final\_list.append(sublist[counter])

counter += 1

iterator += 1

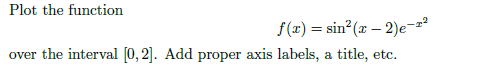
return(print(final\_list))

listA = [[1,3],[3,6]]

sublist\_elements(listA

**Exercise 10**

Use mathplotlib



import math

import matplotlib.pyplot as plt

def f(x):

result = math.pow(math.sin(x-2),2) \* math.pow(math.e,- (math.pow(x,2)))

return result

input = []

output = []

iter = 0

while iter <= 2:

input.append(iter)

iter+= 0.08

iter = 0

while iter < len(input):

output.append(f(input[iter]))

iter += 1

print(output)

print(math.pi)

plt.xlabel("X-Axis")

plt.ylabel("Y-Axis")

plt.title("sin^2(x-2)e^(-x^2)")

plt.ylim(0,1.5)

plt.xlim(0,2.5)

plt.plot(input,output)

plt.show()

**Exercise 11**



def iteration\_sum\_list(list):

iterator = 0

sum = 0

while iterator < len(list):

sum += list[iterator]

iterator += 1

return sum

def recursive\_sum\_list(list):

sum = 0

if len(list) == 0:

return sum

else:

sum = list[0]

del list[0]

return sum + recursive\_sum\_list(list)

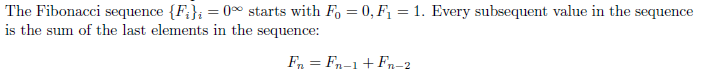
a = [8, 9, 7, 2, 1]

print("Sum of all elements in list a using iteration:", iteration\_sum\_list(a))

print("Sum of all elements in a list using recursion:", recursive\_sum\_list(a))

n

**Exercise 12**



def fib(n):

if n <= 1:

return n

else:

return fib(n-1) + fib(n-2)

print("5th Fibonacci Number:", fib(5))

print("10th Fibonacci Number:", fib(10))

print("15th Fibonacci Number:",fib(15))

print("20th Fibonacci Number:", fib(20))

**Exercise 13**

Write a Python program that extracts the email addresses of a file. An email file emails.txt is provided to test your program.

<http://rubular.com/> is a site that can be useful to get familiar with regular expressions.

import re

file = open('email.txt', 'r')

file = file.read()

email = re.findall(r'([^ ]+[@][^ ]+[.][a-z]+)', file)

print(email)

**References**

Stanford courses on Python <https://web.stanford.edu/~schmit/cme193/exercises.html>