**Python Exercises**

**12/14/18**

**Jackie Shao**

**Due 12/17/18**

**Python**

**Exercises**

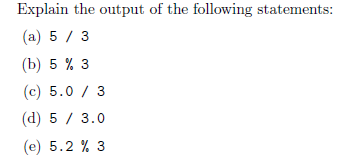
The code must be submitted under your name in GitHub in a repository called Python. Work individually.

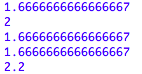
Create one file with all your work and name it: cs361python.py or cs 631python.py.

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**

****

****

1. **1.6666666666666667**

**This is an int divided by int. In the older versions of Python, float is not returned. But in Python 3.7.1, the float is kept.**

1. **2**

**This is an int modulo int. Modulo displays the remainder when the two numbers are divided. When 5 is divided by 3, there is a remainder of 2.**

1. **1.6666666666666667**

**This is a float divided by int. When the dividend is a float, the answer is outputted as a float as well.**

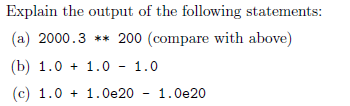
1. **1.6666666666666667**

**This is an int divided by a float. When the divisor is a float, the answer is outputted as a float as well.**

1. **2.2**

**This is a float modulo int. When 5.2 is divided by 3, there is a remainder of 2.2 and since the dividend is a float, a float answer is returned.**

**Exercise 2**

****

****

1. **Traceback (most recent call last):**

**File "<pyshell#10>", line 1, in <module>**

**2000.3 \*\*200**

**OverflowError: (34, 'Result too large')**

**2000.3 is 2000.3 to the 200th power, and cannot be held within a float**

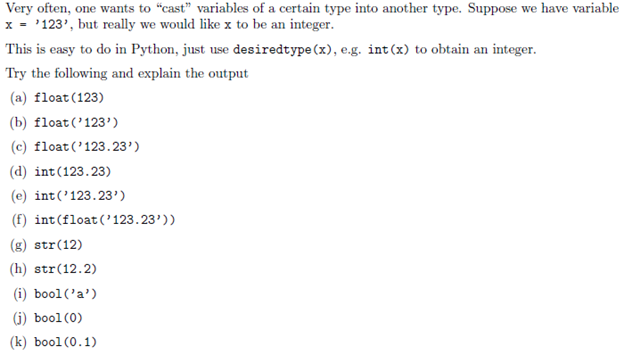
1. **1.0**

**1.0+ 1.0 = 2.0, then 2.0-1.0= 1.0. The problem is read from left to right, and then outputted.**

1. **0.0**

**Whatever you change the first 1.0 to, the answer is always 0.0. Floats store values as approximations, then later scale with the exponent. In this problem, these values are too large, causing the incorrect output.**

**Exercise 3**

****

****

1. **123.0**

**123 is originally an int, but it is being casted as a float so a float is outputted.**

1. **123.0**

**‘123’ is originally a literal, but it is being casted as a float so a float is outputted.**

1. **123.23**

**‘123.23’ is originally a literal, but is being casted as a float so a float is outputted.**

1. **123**

**123.23 is originally a float, but is being casted as an int so the decimal is dropped.**

1. **traceback (most recent call last):  
    File "<pyshell#19>", line 1, in <module>  
    int('123.23')  
   ValueError: invalid literal for int() with base 10: '123.23'**

**since the literal (a type of string) is a float, and casted as an int, an error is thrown as the value cannot be manipulated.**

1. **123**

**‘123.23’ is originally a literal, but is being casted as a float first, resulting in a float output. Then it is casted as an int, resulting in the decimal places being dropped.**

1. **‘12’**

**12 is originally an int, but is being casted as a string so a string is outputted.**

1. **‘12.2’**

**Similar to above, the float 12.2 is converted to a string, ‘12.2’, and thus outputted.**

1. **True**

**bool() returns true for any value besides 0.**

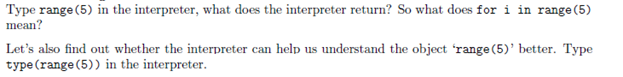
1. **False**

**bool() returns true for any value besides 0, which outputs false.**

1. **True**

**0.1 is not 0, therefore it returns true**

**Exercise 4**

****

****

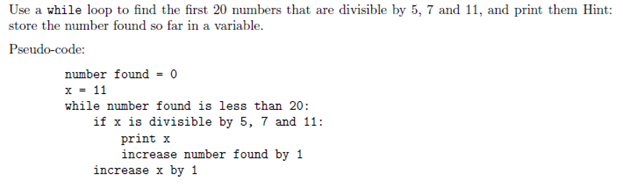
**Typing range(5) produces:**

**range(0,5)**

**It generates a list of 5 integers, starting from 0.**

**Typing type(range(5)) produces:**

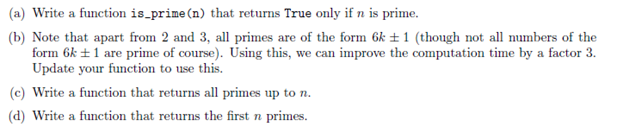
**<class 'range'>**

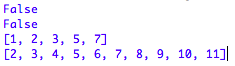
**Exercise 5** 

****

**numFound = 0;  
x = 11;  
while (numFound < 20):  
 if (x % 5 == 0) & (x % 7 == 0) & (x % 11 == 0):  
 print(x)  
 numFound += 1  
 x += 1**

**Exercise 6**

****

****

1. **def is\_prime(n):  
    if n>1:  
    for x in range(2,n):  
    if (n%x) ==0:  
    print(False)  
    break  
    else:  
    print(True)  
    else:  
    print(False)  
     
   is\_prime(6) ← to test it**

**------------------------------------------------------**

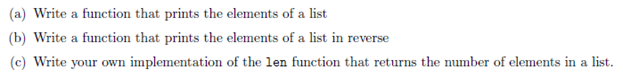
1. **def is\_prime\_faster(n):  
    if n==2 or n ==3:  
    print(True)  
    if n%2==0 or n%3==0:  
    print(False)  
    i = 5  
    x = 2  
    sq = int(n\*\*.5) + 1  
    while i<= sq:  
    if n%i ==0:  
    print(False)  
    i = i+x  
    x = 6-x  
    print(True)**

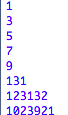
**is\_prime\_faster(6) ← to test it**

**------------------------------------------------------**

1. **def prime\_upto(n):  
    list=[]  
    for num in range(1,n+1):  
    if all(num%i !=0 for i in range(2, num)):  
    list.append(num)  
    print(list)  
     
   prime\_upto(10) ← to test it**
2. **def first\_n(n):  
    prime=[]  
    i = 2  
    while len(prime) != n:  
    for num in range(2, 1//2+1):  
    if i%num == 0:  
    break  
    else:  
    prime.append(i)  
    i = i+1  
    print(prime)  
     
   first\_n(10) ← to test it**

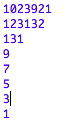
**Exercise 7**

****

1. **L=[1, 3, 5, 7, 9, 131, 123132, 1023921]  
   def list\_print(list):  
    for items in list:  
    print(items)  
   list\_print(L)  
   **

**------------------------------**

1. **def reverse\_print(list):  
    for items in reversed(list):  
    print(items)  
   reverse\_print(L)  
   print()**

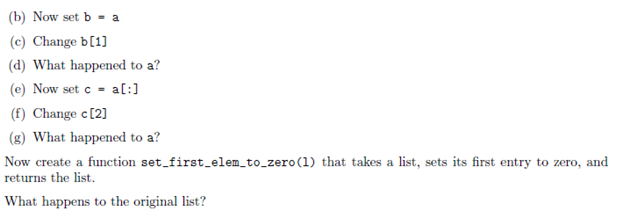
****

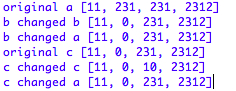
1. **def len(list):  
    elements = 0  
    for num in list:  
    elements = elements + 1  
    print('You have', elements, 'elements.')  
   len(L)**

****

**Exercise 8**

****

****

****

1. **a = [11, 231, 231, 2312]**
2. **b = a**
3. **b[1] = 0**
4. **B[1] is a pointer pointing to a[1]. By tracing the pointer, a[1] is also changed when the value of b[1] is changed.**
5. **c = a[:]**
6. **c[2] = 10**
7. **C[] is a copy of the list a[] so when any element of c[] is changed, the original list (a[]) is untouched.**

****

**The new function created is a pointer pointing to the original list (a[]) so when an element of the pointer list is changed, the corresponding element of the original list also changes.**

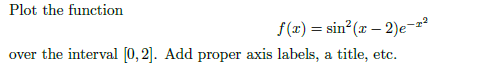
**Exercise 9**

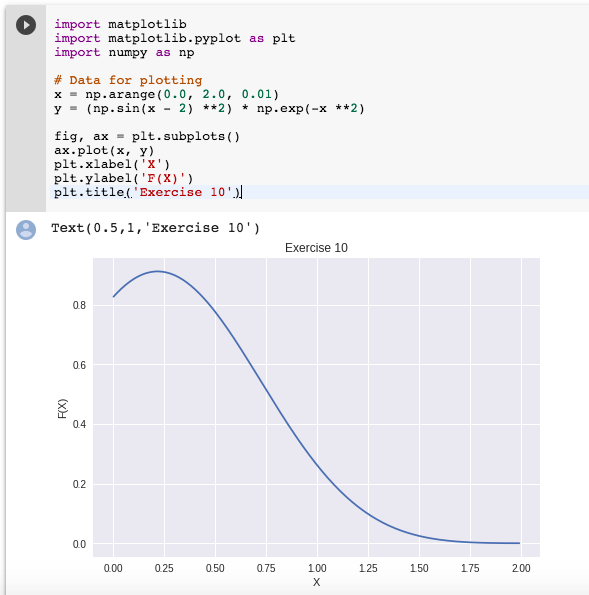


1. **def combine():  
    list = [[1,3], [3,6]]  
    newList = []  
    for i in list:  
    newList = list[0] + list[1]  
    print(newList)  
   combine()**

**Exercise 10**

Use mathplotlib





**Exercise 11**

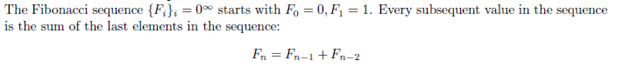
****

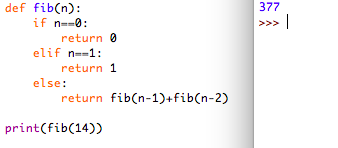
****

**List = [2, 4, 6, 8, 10]  
  
def iterative(L):  
 if len(L) == 0:  
 return 0  
 product = 1  
 for elements in L:  
 product = product \* elements  
 return product**

**def recurs(L):  
 if len(L) == 0:  
 return 0  
 if len(L) == 1:  
 return L[0]  
 else:  
 return recurs([L[0]]) \* recurs(L[1:])  
print('The product is', iterative(List))  
print('Recursively, the product is', recurs(List))**

**Exercise 12**

****

****

**def fib(n):  
 if n == 0:  
 return 0  
 elif n == 1:  
 return 1  
 else:  
 return fib(n-1) + fib(n-2)  
print(fib(14)) ← to test it**

**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('emails.txt', "r")  
file = file.read()  
  
eAddress = re.findall(r'[\w\"\.-@]\*[\w\"\w.-]+@[\w\.-]+\.[\w]+', file)  
print (eAddress)**