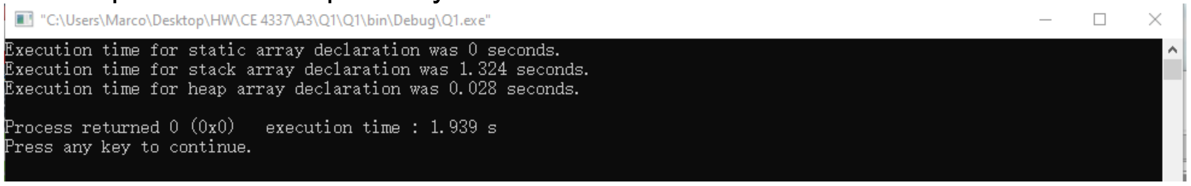
CS/CE 4337 - Assignment#3 Due Date: 10/20/19, 11:59 pm

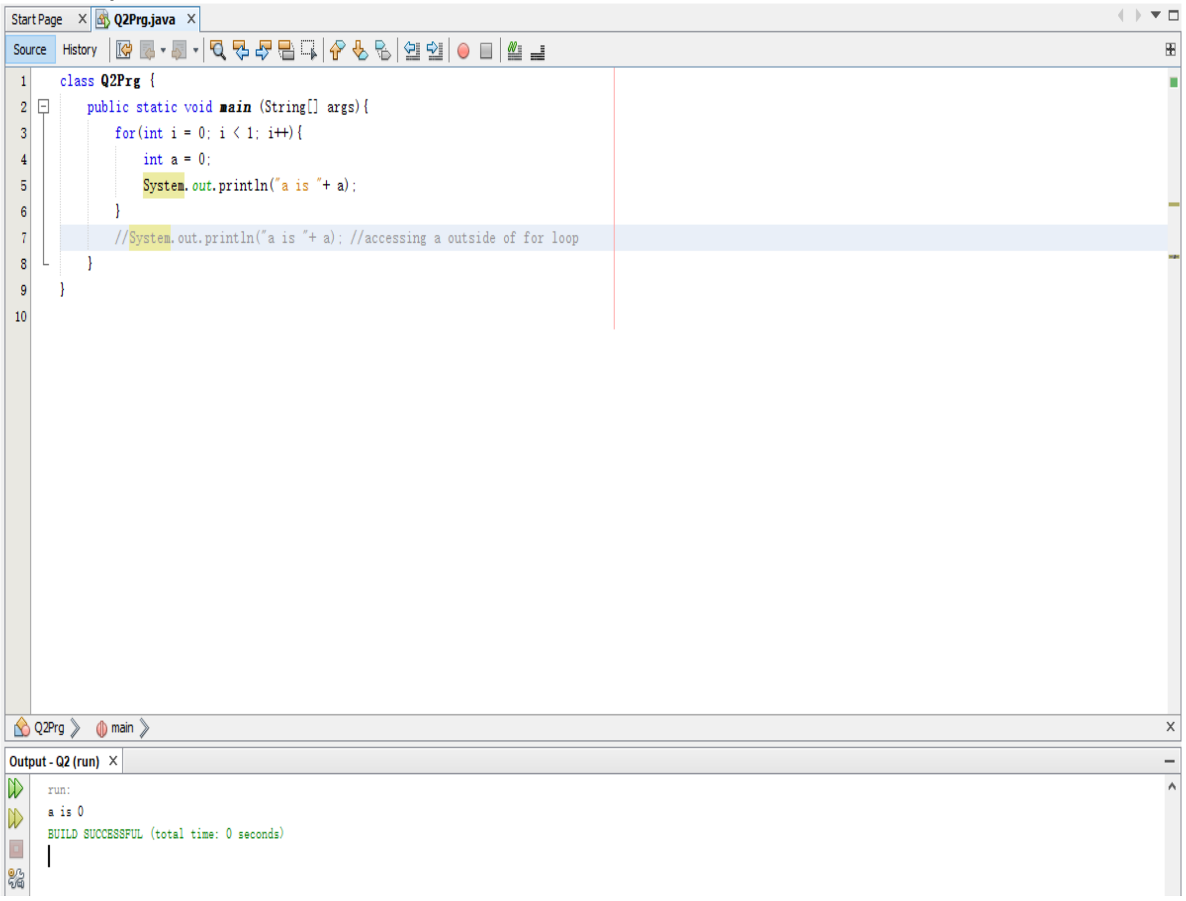
1- Write three functions in C++ : one that declares a large array statistically, one that declares the same large array on the stack, and one that creates the same large array from the heap . Call each of the subprograms a large number of times (at least 100000) and output the time required by each .



"C:\Users\Marco\Desktop\HW\CE 4337\A3\Q1\Q1\bin\Debug\Q1.exe"

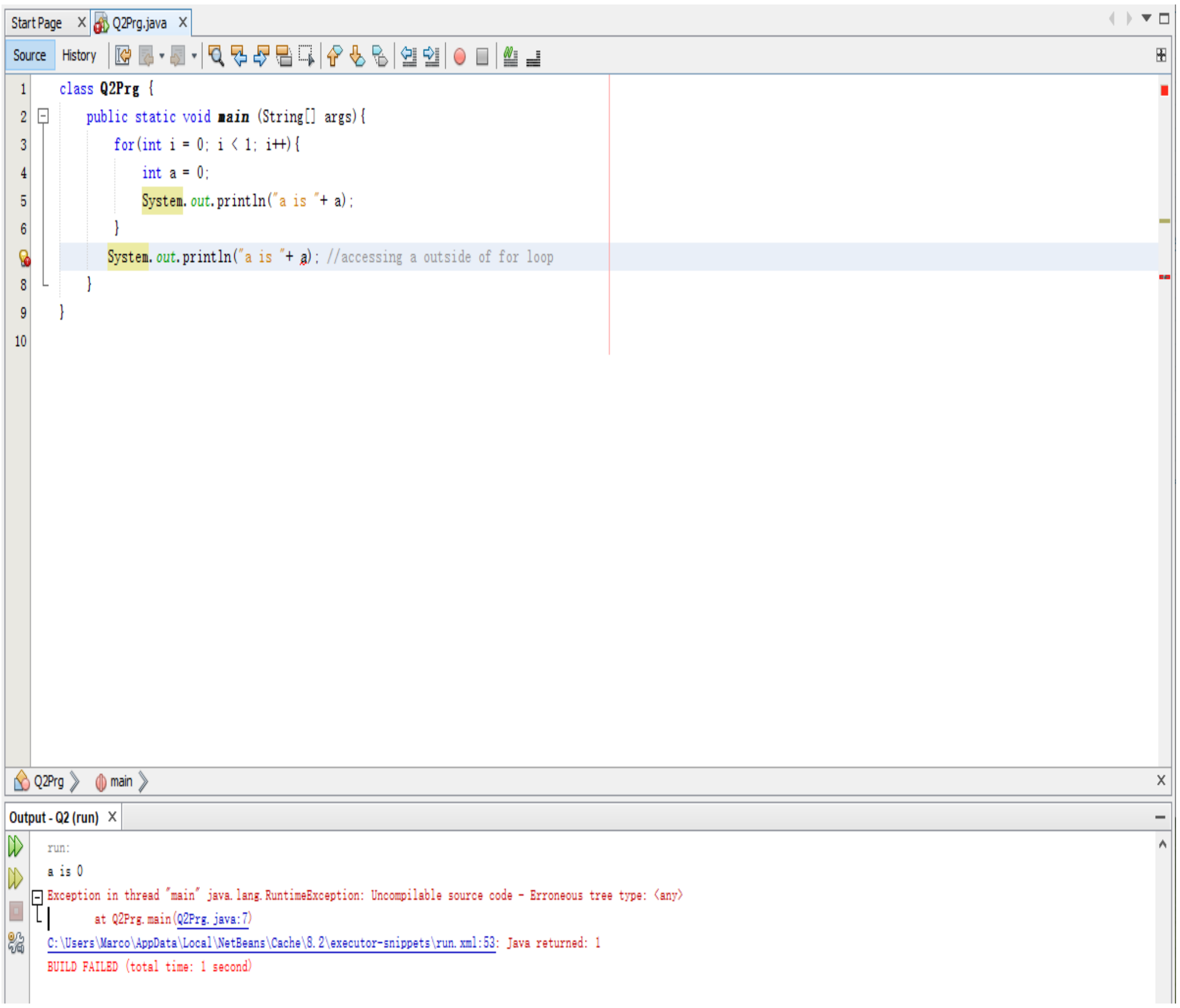
```
Execution time for static array declaration was 0 seconds.
Execution time for stack array declaration was 1.324 seconds.
Execution time for heap array declaration was 0.028 seconds.

Process returned 0 (0x0)   execution time : 1.939 s
Press any key to continue.
```

2- Write test programs in Java to determine the scope of a variable declared in a `for` statement. Specifically, the code must determine whether such a variable is visible after the body of the `for` statement



```java
class Q2Prg {
    public static void main (String[] args){
        for(int i = 0; i < 1; i++){
            int a = 0;
            System.out.println("a is "+ a);
        }
        //System.out.println("a is "+ a); //accessing a outside of for loop
    }
}
```

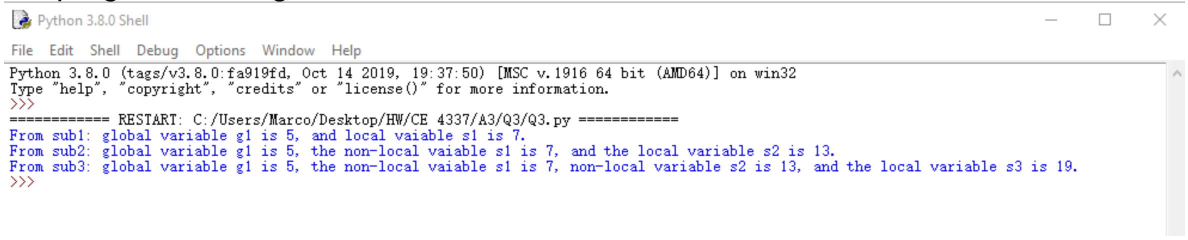
```
run:
a is 0
BUILD SUCCESSFUL (total time: 0 seconds)
```



```java
class Q2Prg {
    public static void main (String[] args){
        for(int i = 0; i < 1; i++){
            int a = 0;
            System.out.println("a is "+ a);
        }
       System.out.println("a is "+ a); //accessing a outside of for loop
    }
}
```

```
run:
a is 0
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - Erroneous tree type: <any>
	at Q2Prg.main(Q2Prg.java:7)
C:\Users\Marco\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 1 second)
```

3- Write a Python program that has subprograms nested three deep and in which each nested subprogram references local variables, variables defined in all of its enclosing subprograms, and global variables.

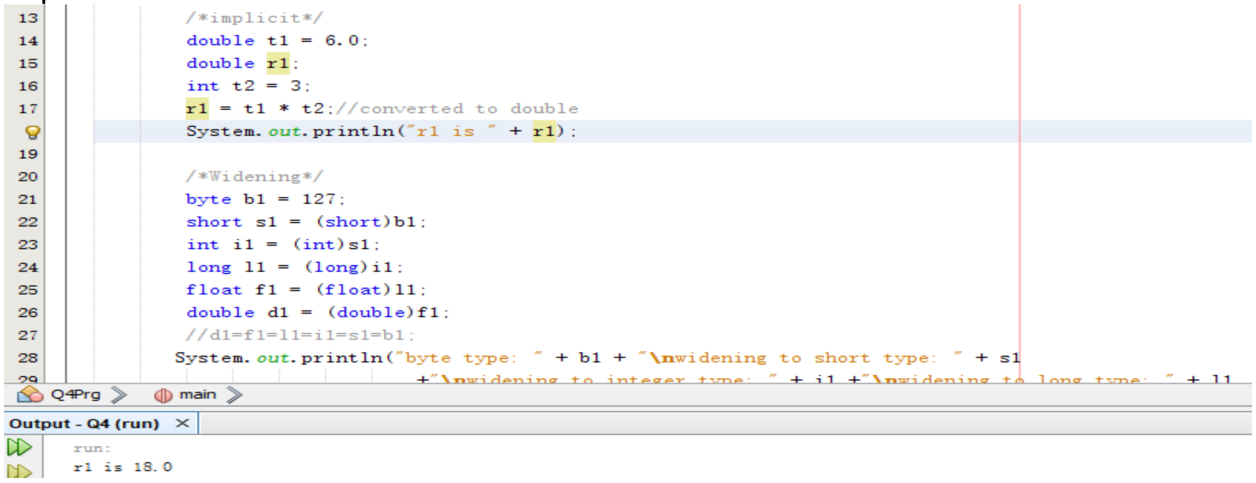


```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/Marco/Desktop/HW/CE 4337/A3/Q3/Q3.py ============
From sub1: global variable g1 is 5, and local vaiable s1 is 7.
From sub2: global variable g1 is 5, the non-local vaiable s1 is 7, and the local variable s2 is 13.
From sub3: global variable g1 is 5, the non-local vaiable s1 is 7, non-local variable s2 is 13, and the local variable s3 is 19.
>>>
```

4- Design a set of simple test programs to determine the type compatibility rules of a java compiler to which you have access. Write a report (7-15 lines)

In java, there is implicit and explicit type compatibility rules. Values are converted implicitly to the value of next level during multiplication and division of an integer and a float or double value. Their result will always be float or double. Explicit conversion happens when casting a value to another type. The order of widening in explicit conversion is byte, short, int, long, float, and double. As a variable gets widen, its information will still retain. On the other hand, when narrowing a variable, the insignificant part of the variable will be truncated during the process. The narrowing order is the exact opposite of the widening order. A char variable can be converted to double, float, long, and int. During the conversion, only the ASCII code of the char variable will retain. It is not possible to convert from numerical variables to char variables, char variables or numerical variables to Boolean variables, or from Boolean variables to char variables or numerical variables.

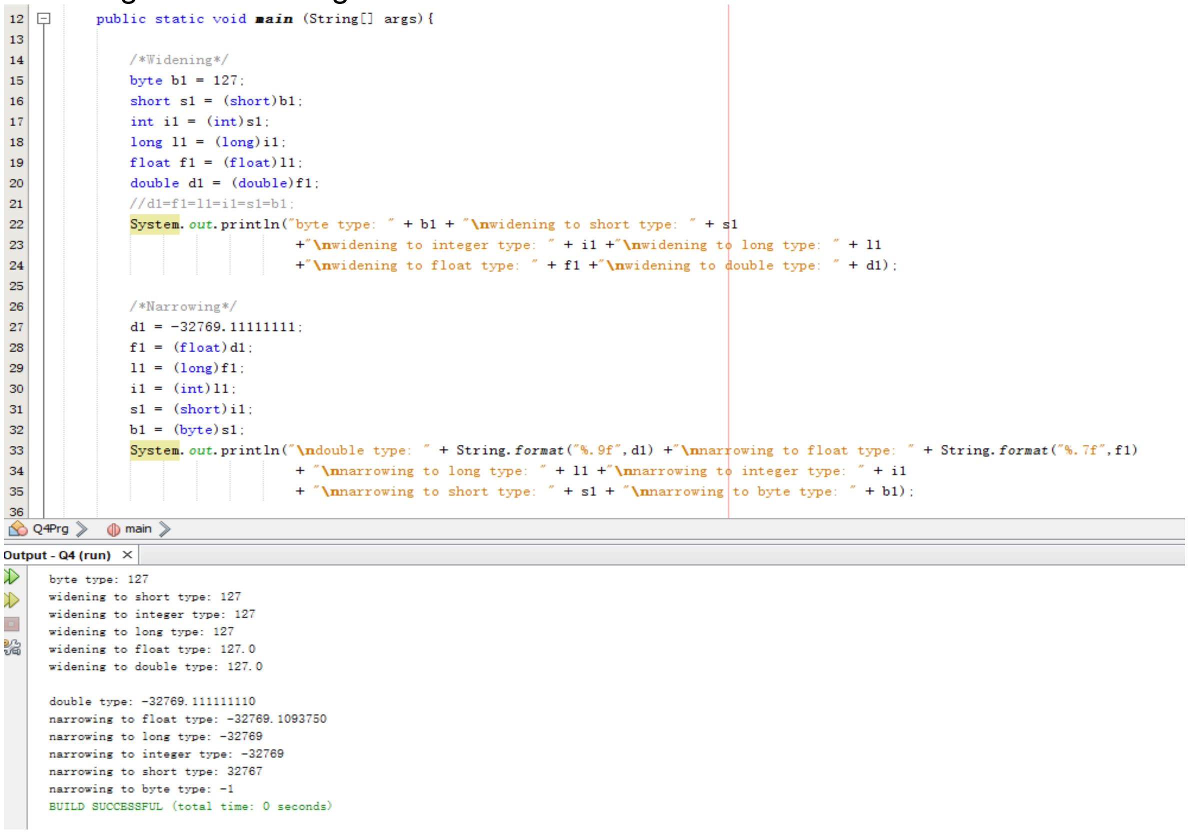
Implicit:



```
        /*implicit*/
        double t1 = 6.0;
        double r1;
        int t2 = 3;
        r1 = t1 * t2;//converted to double
        System.out.println("r1 is " + r1);

        /*Widening*/
        byte b1 = 127;
        short s1 = (short)b1;
        int i1 = (int)s1;
        long l1 = (long)i1;
        float f1 = (float)l1;
        double d1 = (double)f1;
        //d1=f1=l1=i1=s1=b1;
       System.out.println("byte type: " + b1 + "\nwidening to short type: " + s1
```

```
run:
r1 is 18.0
```

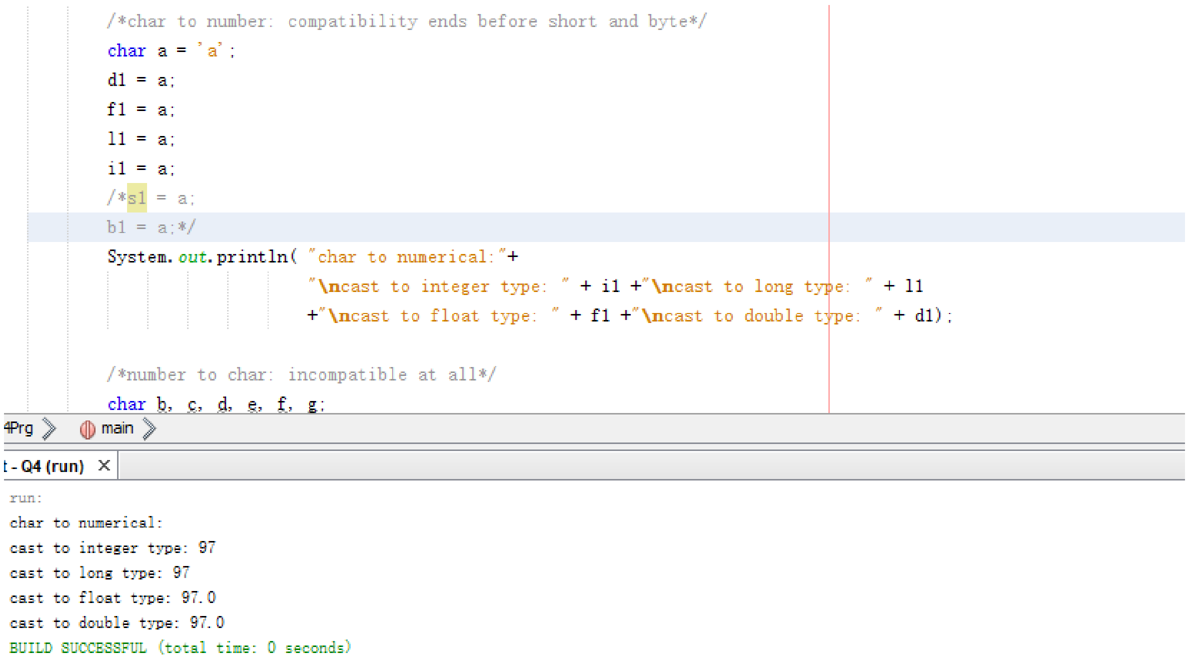
Widening and narrowing:



```
    public static void main (String[] args){

        /*Widening*/
        byte b1 = 127;
        short s1 = (short)b1;
        int i1 = (int)s1;
        long l1 = (long)i1;
        float f1 = (float)l1;
        double d1 = (double)f1;
        //d1=f1=l1=i1=s1=b1;
        System.out.println("byte type: " + b1 + "\nwidening to short type: " + s1
                    +"\nwidening to integer type: " + i1 +"\nwidening to long type: " + l1
                    +"\nwidening to float type: " + f1 +"\nwidening to double type: " + d1);

        /*Narrowing*/
        d1 = -32769.11111111;
        f1 = (float)d1;
        l1 = (long)f1;
        i1 = (int)l1;
        s1 = (short)i1;
        b1 = (byte)s1;
        System.out.println("\ndouble type: " + String.format("%.9f",d1) +"\nnarrowing to float type: " + String.format("%.7f",f1)
                    + "\nnarrowing to long type: " + l1 +"\nnarrowing to integer type: " + i1
                    + "\nnarrowing to short type: " + s1 + "\nnarrowing to byte type: " + b1);
```

```
byte type: 127
widening to short type: 127
widening to integer type: 127
widening to long type: 127
widening to float type: 127.0
widening to double type: 127.0

double type: -32769.111111110
narrowing to float type: -32769.1093750
narrowing to long type: -32769
narrowing to integer type: -32769
narrowing to short type: 32767
narrowing to byte type: -1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Char to num:

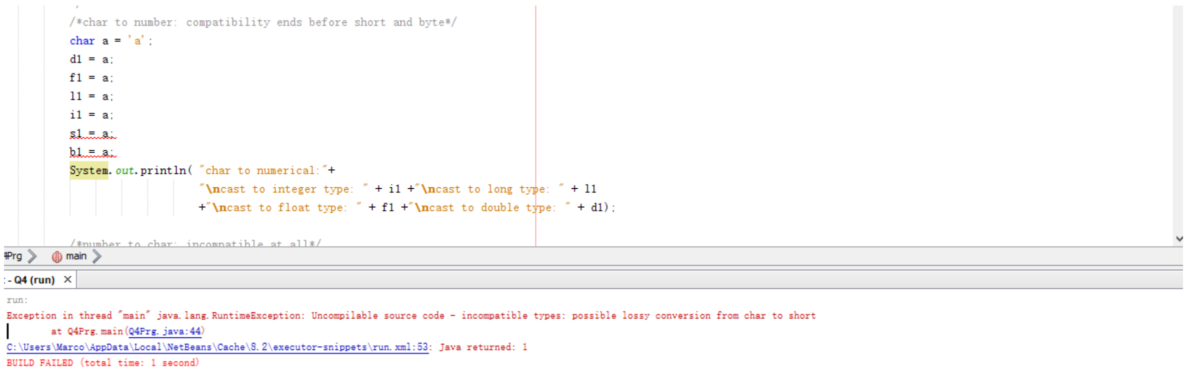


```
/*char to number: compatibility ends before short and byte*/
char a = 'a';
d1 = a;
f1 = a;
l1 = a;
i1 = a;
/*s1 = a;
b1 = a;*/
System.out.println( "char to numerical:"+
                    "\ncast to integer type: " + i1 +"\ncast to long type: " + l1
                    +"\ncast to float type: " + f1 +"\ncast to double type: " + d1);

/*number to char: incompatible at all*/
char b, c, d, e, f, g;
```

4Prg › main ›

t - Q4 (run) ×

```
run:
char to numerical:
cast to integer type: 97
cast to long type: 97
cast to float type: 97.0
cast to double type: 97.0
BUILD SUCCESSFUL (total time: 0 seconds)
```



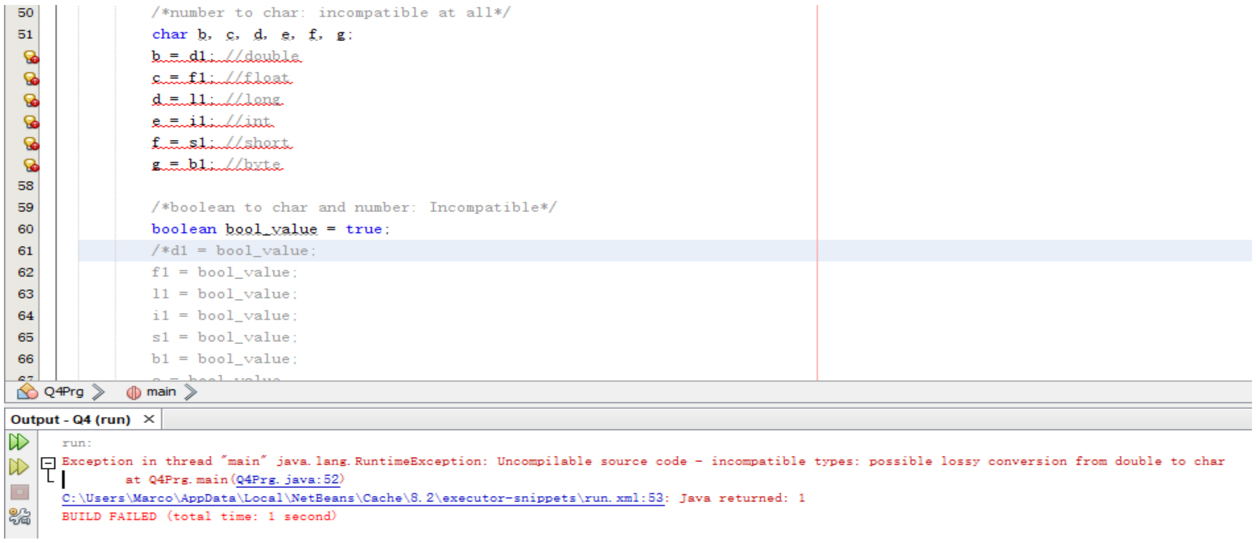
```
/*char to number: compatibility ends before short and byte*/
char a = 'a';
d1 = a;
f1 = a;
l1 = a;
i1 = a;
s1 = a;
b1 = a;
System.out.println( "char to numerical:"+
                    "\ncast to integer type: " + i1 +"\ncast to long type: " + l1
                    +"\ncast to float type: " + f1 +"\ncast to double type: " + d1);

/*number to char: incompatible at all*/
```

4Prg › main ›

: - Q4 (run) ×

```
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - incompatible types: possible lossy conversion from char to short
	at Q4Prg.main(Q4Prg.java:44)
C:\Users\Marco\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 1 second)
```

Num to char:



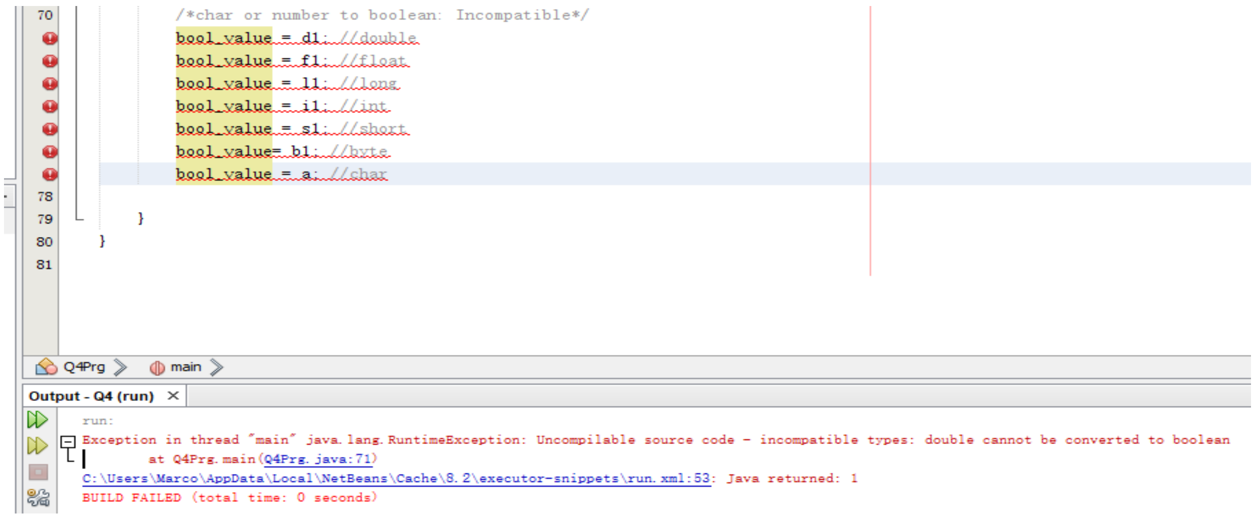
```
/*number to char: incompatible at all*/
char b, c, d, e, f, g;
b = d1; //double
c = f1; //float
d = l1; //long
e = i1; //int
f = s1; //short
g = b1; //byte

/*boolean to char and number: Incompatible*/
boolean bool_value = true;
/*d1 = bool_value;
f1 = bool_value;
l1 = bool_value;
i1 = bool_value;
s1 = bool_value;
b1 = bool_value;
```

Q4Prg › main ›

Output - Q4 (run) ×

```
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - incompatible types: possible lossy conversion from double to char
	at Q4Prg.main(Q4Prg.java:52)
C:\Users\Marco\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 1 second)
```

Char or num to bool:



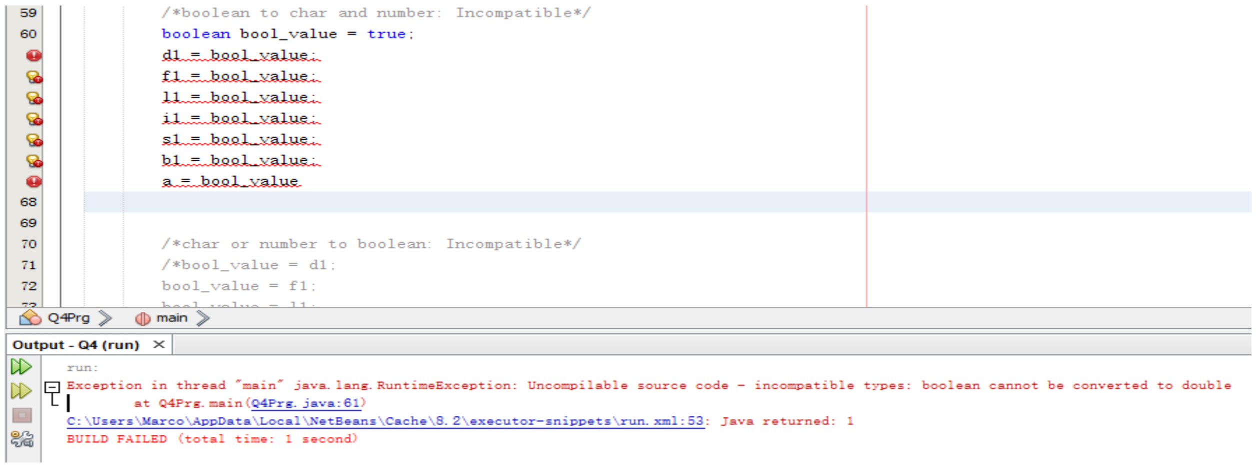
```
        /*char or number to boolean: Incompatible*/
        bool_value = d1; //double
        bool_value = f1; //float
        bool_value = l1; //long
        bool_value = i1; //int
        bool_value = s1; //short
        bool_value= b1; //byte
        bool_value = a; //char

    }
}
```

Q4Prg > main >

Output - Q4 (run)

```
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - incompatible types: double cannot be converted to boolean
        at Q4Prg.main(Q4Prg.java:71)
C:\Users\Marco\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 0 seconds)
```

Bool to char or num:



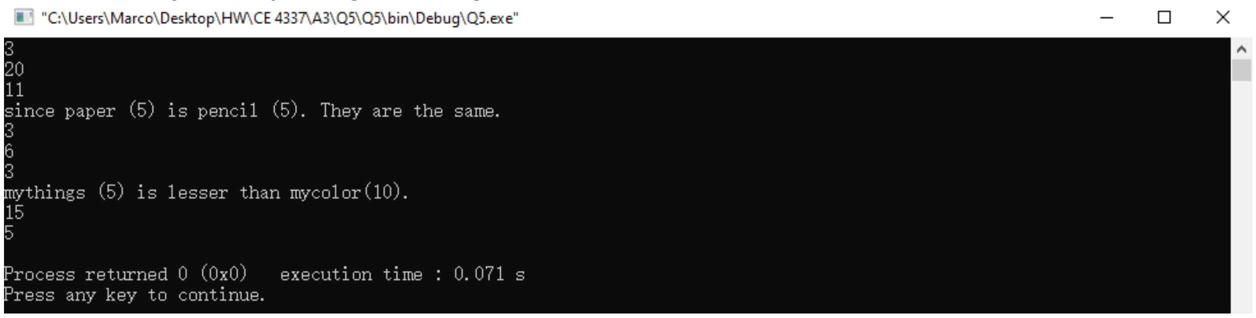
```
        /*boolean to char and number: Incompatible*/
        boolean bool_value = true;
        d1 = bool_value;
        f1 = bool_value;
        l1 = bool_value;
        i1 = bool_value;
        s1 = bool_value;
        b1 = bool_value;
        a = bool_value


        /*char or number to boolean: Incompatible*/
        /*bool_value = d1;
        bool_value = f1;
```

Q4Prg > main >

Output - Q4 (run)

```
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - incompatible types: boolean cannot be converted to double
        at Q4Prg.main(Q4Prg.java:61)
C:\Users\Marco\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 1 second)
```

5- Write a simple program in C++ to investigate the safety of its enumeration types. Include at least 10 different operations on enumeration types to determine what incorrect or just silly things are legal

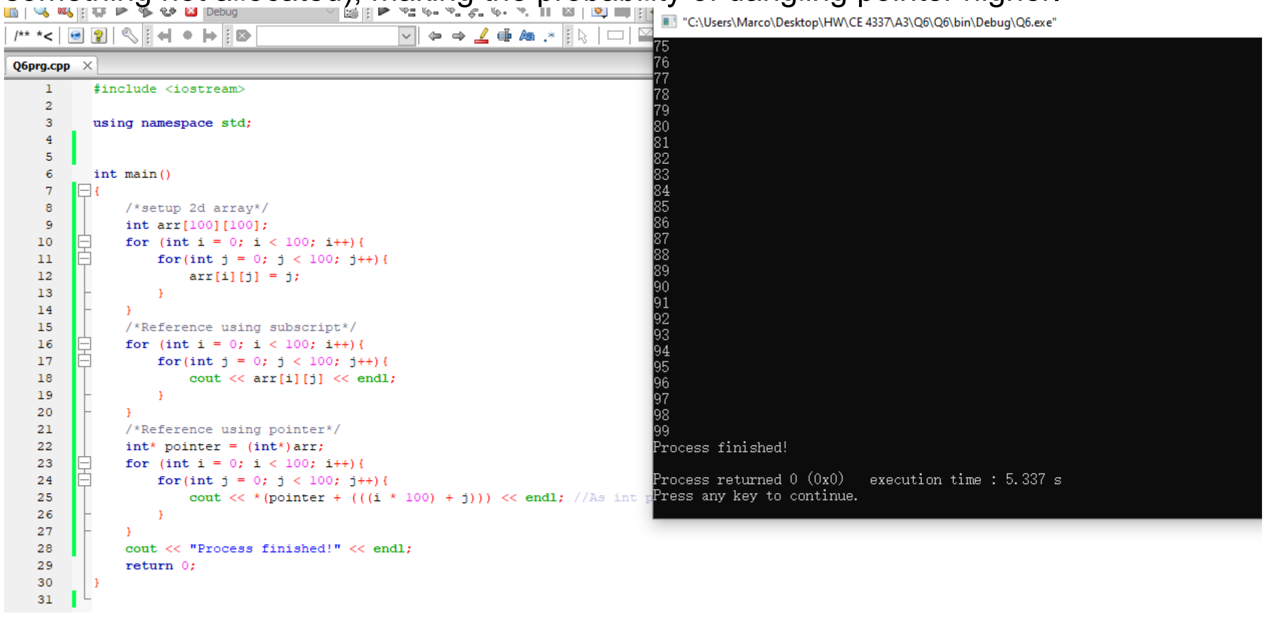


"C:\Users\Marco\Desktop\HW\CE 4337\A3\Q5\Q5\bin\Debug\Q5.exe"

```
3
20
11
since paper (5) is pencil (5). They are the same.
3
6
3
mythings (5) is lesser than mycolor(10).
15
5

Process returned 0 (0x0)   execution time : 0.071 s
Press any key to continue.
```

6- Write a C++ program that does a large number of references to elements of two-dimensioned arrays, using only subscripting. Write a second program that does the same operations but uses pointers and pointer arithmetic for the storage-mapping function to do the array references. Which of the two programs is likely to be more reliable? Why? (3- 6 lines)

Using subscripting is more reliable. Subscripting ties closely to the 2D array. As a result, it will not be destroyed until the end of execution. Subscripting also automatically calculates the correct address of each slot in the 2D array, avoiding the problem of mis-referencing. Using pointers not only goes over boundaries of each slot of the array easily, but also goes out of bound of the array easily (referencing something not allocated), making the probability of dangling pointer higher.


Q6prg.cpp

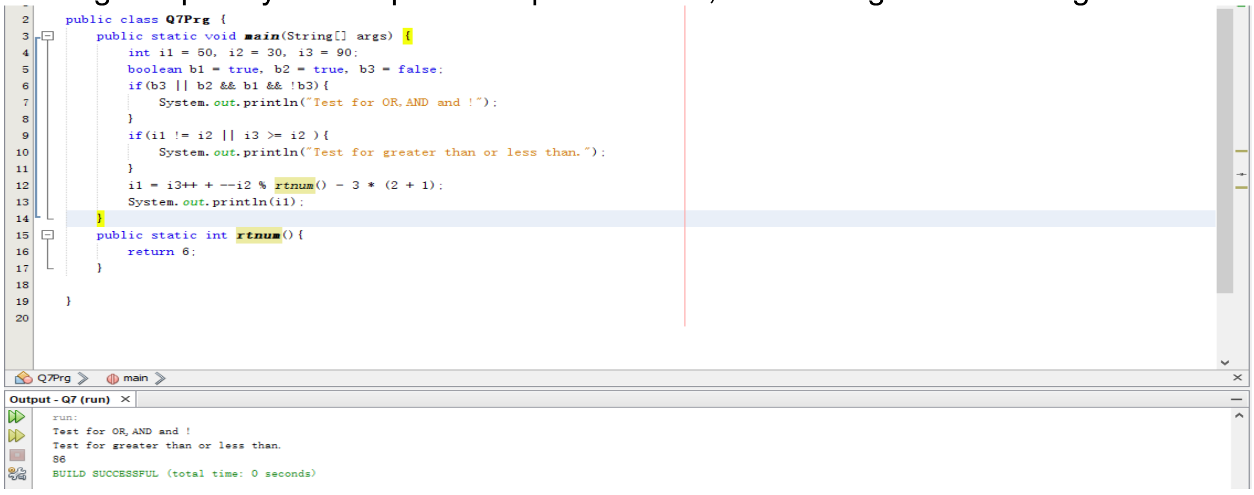
```cpp
#include <iostream>

using namespace std;


int main()
{
    /*setup 2d array*/
    int arr[100][100];
    for (int i = 0; i < 100; i++){
        for(int j = 0; j < 100; j++){
            arr[i][j] = j;
        }
    }
    /*Reference using subscript*/
    for (int i = 0; i < 100; i++){
        for(int j = 0; j < 100; j++){
            cout << arr[i][j] << endl;
        }
    }
    /*Reference using pointer*/
    int* pointer = (int*)arr;
    for (int i = 0; i < 100; i++){
        for(int j = 0; j < 100; j++){
            cout << *(pointer + (((i * 100) + j))) << endl; //As int p
        }
    }
    cout << "Process finished!" << endl;
    return 0;
}
```

"C:\Users\Marco\Desktop\HW\CE 4337\A3\Q6\Q6\bin\Debug\Q6.exe"

```
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
Process finished!

Process returned 0 (0x0)   execution time : 5.337 s
Press any key to continue.
```

7- Write a Java program that exposes Java's rule for operand evaluation order when one of the operands is a method call. Write a report (5 to 10 lines)

The lowest priority is the assignment symbol. The next symbol in the order of low to high is logical OR, then logical AND, then logical equality and logical inequality, then logical greater than along with logical less than, then addition and subtraction, including string concatenation, then multiplication, division and mods, then unary plus and minus, unary logical not, pre-increment and decrement, then post increment and decrement. The second highest priority will be a method call.
The highest priority in the operand is parentheses, associating from left to right.



```java
public class Q7Prg {
    public static void main(String[] args) {
        int i1 = 50, i2 = 30, i3 = 90;
        boolean b1 = true, b2 = true, b3 = false;
        if(b3 || b2 && b1 && !b3){
            System.out.println("Test for OR, AND and !");
        }
        if(i1 != i2 || i3 >= i2 ){
            System.out.println("Test for greater than or less than.");
        }
        i1 = i3++ + --i2 % rtnum() - 3 * (2 + 1);
        System.out.println(i1);
    }
    public static int rtnum(){
        return 6;
    }

}
```

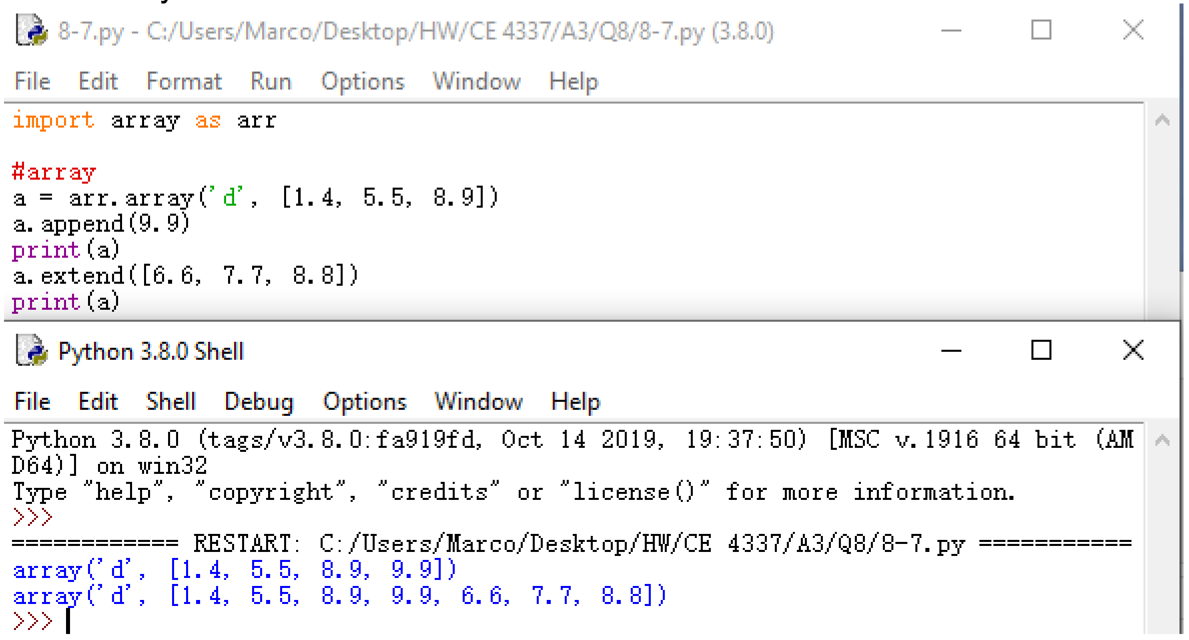
Q7Prg > main >

Output - Q7 (run)

```
run:
Test for OR, AND and !
Test for greater than or less than.
86
BUILD SUCCESSFUL (total time: 0 seconds)
```

8- Answer the following Python Interview questions

- How is Python an interpreted language?
    - Because Python is not in machi

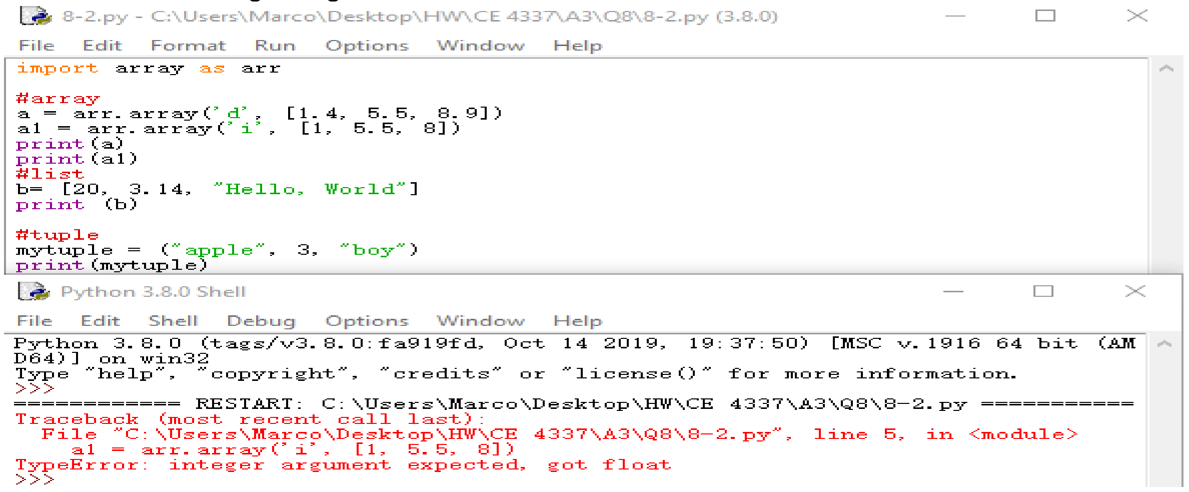

8-7.py - C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-7.py (3.8.0)
File Edit Format Run Options Window Help
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help

```
import array as arr

#array
a = arr.array('d', [1.4, 5.5, 8.9])
a.append(9.9)
print(a)
a.extend([6.6, 7.7, 8.8])
print(a)
```

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-7.py ===========
array('d', [1.4, 5.5, 8.9, 9.9])
array('d', [1.4, 5.5, 8.9, 9.9, 6.6, 7.7, 8.8])
>>> 
```

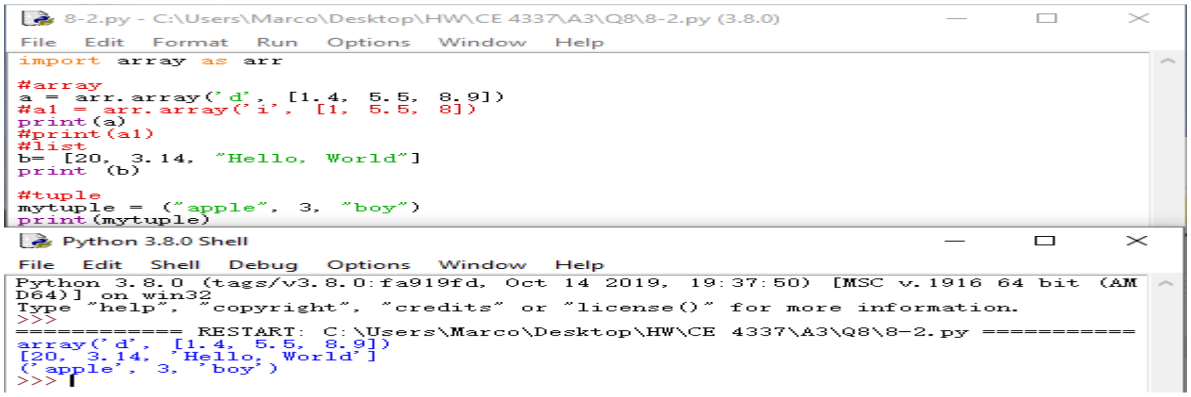
ne code prior to runtime. Its translation happens as the program is executing. Its program runs directly from the source code.

- What is the difference between Python Arrays, lists, tuples, and records? Explain it with examples
    - Python array is a contiguous allocation for storing same type data
    - Python lists is a collection of data with no restriction of data type
    - Python tuple is an immutable list of unnamed data accessed through offsets from the beginning of the structure.


8-2.py - C:\Users\Marco\Desktop\HW\CE 4337\A3\Q8\8-2.py (3.8.0)
File Edit Format Run Options Window Help
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help

```
import array as arr

#array
a = arr.array('d', [1.4, 5.5, 8.9])
a1 = arr.array('i', [1, 5.5, 8])
print(a)
print(a1)
#list
b= [20, 3.14, "Hello, World"]
print (b)

#tuple
mytuple = ("apple", 3, "boy")
print(mytuple)
```

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:\Users\Marco\Desktop\HW\CE 4337\A3\Q8\8-2.py ===========
Traceback (most recent call last):
  File "C:\Users\Marco\Desktop\HW\CE 4337\A3\Q8\8-2.py", line 5, in <module>
    a1 = arr.array('i', [1, 5.5, 8])
TypeError: integer argument expected, got float
>>>
```

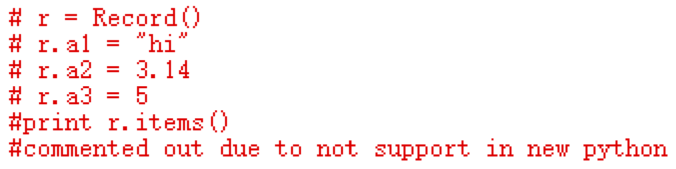

8-2.py - C:\Users\Marco\Desktop\HW\CE 4337\A3\Q8\8-2.py (3.8.0)
File Edit Format Run Options Window Help

```
import array as arr

#array
a = arr.array('d', [1.4, 5.5, 8.9])
#a1 = arr.array('i', [1, 5.5, 8])
print(a)
#print(a1)
#list
b= [20, 3.14, "Hello, World"]
print (b)

#tuple
mytuple = ("apple", 3, "boy")
print(mytuple)
```

Python 3.8.0 Shell
File Edit Shell Debug Options Window Help

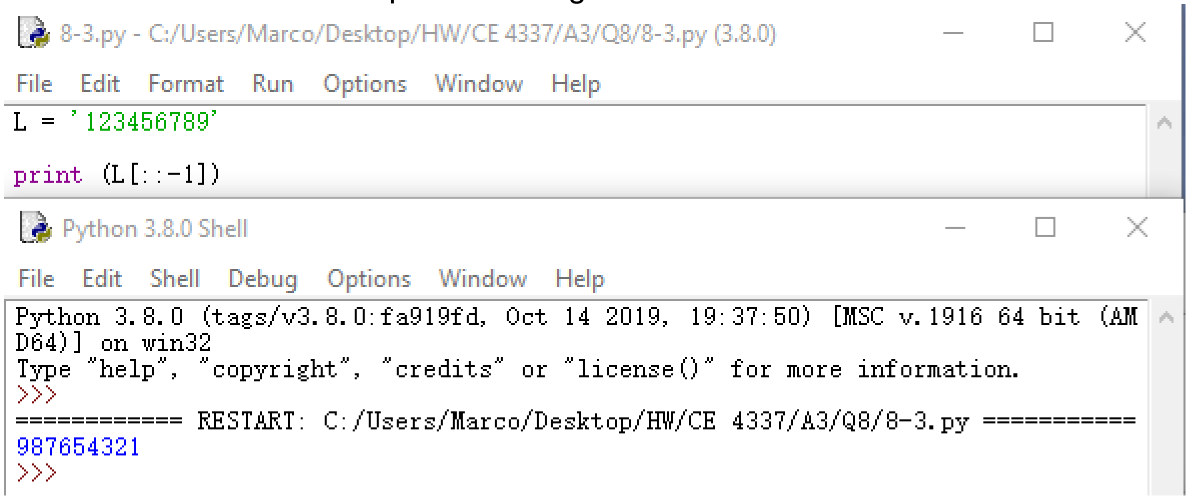
```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:\Users\Marco\Desktop\HW\CE 4337\A3\Q8\8-2.py ===========
array('d', [1.4, 5.5, 8.9])
[20, 3.14, 'Hello, World']
('apple', 3, 'boy')
>>>
```

- Python record is an aggregate of data elements in which the individual elements are identified by names and accessed through offsets from the beginning of the structure.



```
# r = Record()
# r.a1 = "hi"
# r.a2 = 3.14
# r.a3 = 5
#print r.items()
#commented out due to not support in new python
```

- What does [::-1} do? Explain it with an example
  - It reverse the order of the specified string.


8-3.py - C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-3.py (3.8.0)
File Edit Format Run Options Window Help

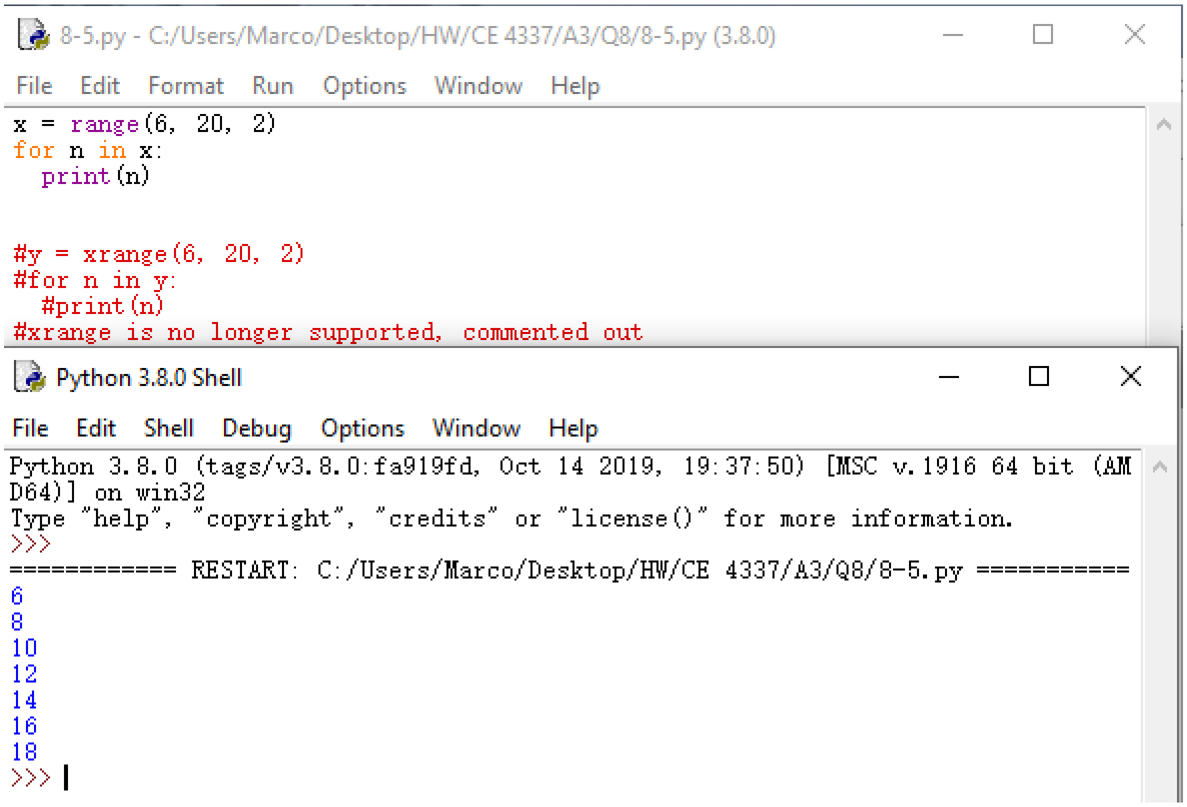
```
L = '123456789'

print (L[::-1])
```

Python 3.8.0 Shell
File Edit Shell Debug Options Window Help

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-3.py ===========
987654321
>>>
```

  -

- How can you randomize the items of a list in place in Python?
  - First import random, then use command random.shuffle(List)
- What is the difference between range & xrange? Explain it with an example
  - They both generate a list of integers, but range outputs python list object, but xrange returns xrange object that generate its own integer object as the user access it. Xrange use less memory.
  - range generates a static list that is ready to use in memory when accessing. xrange has to generate an integer object every time when accessing an index.

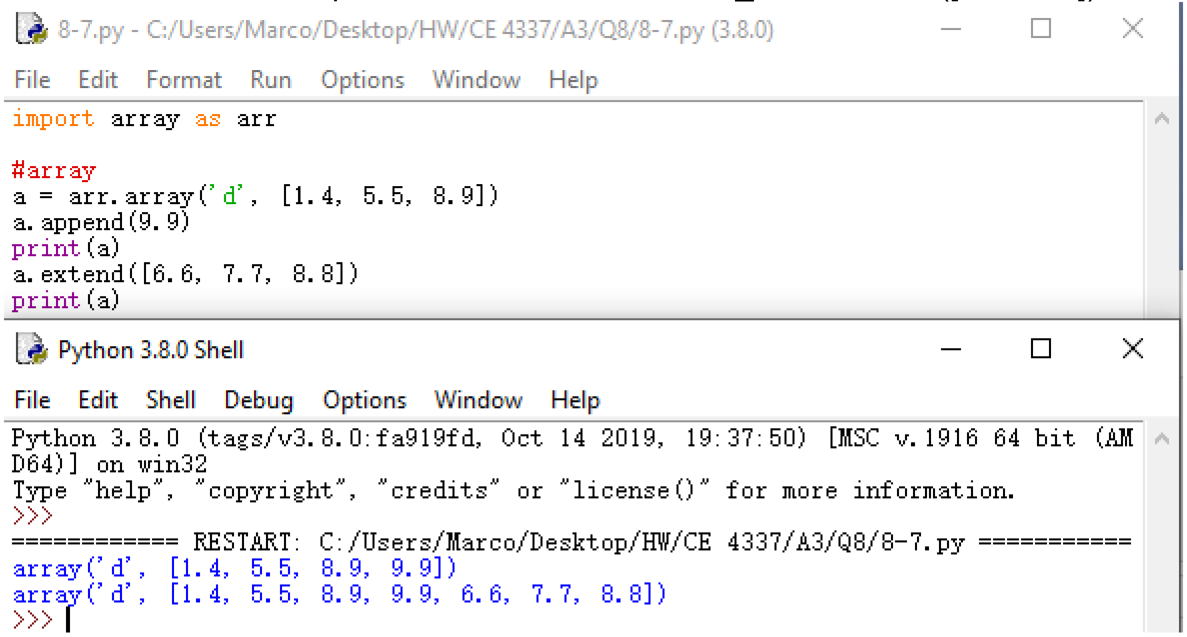

8-5.py - C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-5.py (3.8.0)
File Edit Format Run Options Window Help

```
x = range(6, 20, 2)
for n in x:
  print(n)


#y = xrange(6, 20, 2)
#for n in y:
  #print(n)
#xrange is no longer supported, commented out
```

Python 3.8.0 Shell
File Edit Shell Debug Options Window Help

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-5.py ===========
6
8
10
12
14
16
18
>>>
```

- What advantages do NumPy arrays offer over (nested) Python lists?
    - NumPy runs faster than nested list and permit efficient manipulation of homogeneous numerical data in Python.
    - Its speed boost makes manipulation of elements in multi-dimensional data more convenient.
- How to add values to a python array? Explain it with an example
    - To add a value to the end, use arr_name.append(value)
    - To add multiple values to the end use arr_name.extend([v1,v2,v3])

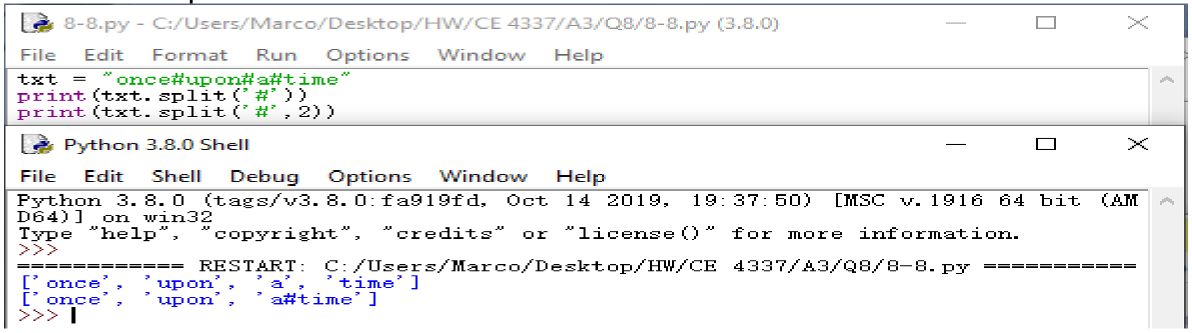

8-7.py - C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-7.py (3.8.0)
File Edit Format Run Options Window Help

```
import array as arr

#array
a = arr.array('d', [1.4, 5.5, 8.9])
a.append(9.9)
print(a)
a.extend([6.6, 7.7, 8.8])
print(a)
```

Python 3.8.0 Shell
File Edit Shell Debug Options Window Help

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-7.py ===========
array('d', [1.4, 5.5, 8.9, 9.9])
array('d', [1.4, 5.5, 8.9, 9.9, 6.6, 7.7, 8.8])
>>>
```

- What is split used for? Explain it with an example
    - When there is a string, str_name.split(‘symbol’) splits the string and gives every element separated by the symbol a slot in the new list.
    - Using str_name.split(‘symbol’, n-1) controls there are only n elements in the list. If n is smaller than the element separable, then the remainder of the list is put as the last element.



8-8.py - C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-8.py (3.8.0)

```
txt = "once#upon#a#time"
print(txt.split('#'))
print(txt.split('#',2))
```

Python 3.8.0 Shell

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/Marco/Desktop/HW/CE 4337/A3/Q8/8-8.py ===========
['once', 'upon', 'a', 'time']
['once', 'upon', 'a#time']
>>> 
```