CIS122 Summer

WELCOME BACK!
Questions?
Test Results

Mean 9
Median 9.5
Mode 10
Test feedback

Problem # - Most common error on the problem
0 - Most common incorrect answer was 0, 1, 2. So, 5 students thought that x+1 modified x's value

1 - Most common incorrect answer was 4, 4, 2. So, 13 students didn't know that int/int = float

2 - Most common error 3.0, 3.0, float, float. Many students miscalculated the values, some got incorrect types.
<table>
<thead>
<tr>
<th>Types</th>
<th>Functions</th>
<th>Flow Control</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>print()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>type()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FINALLY!
Remember when I bragged that Python has lots of built in tools and libraries?

Some built in functions:

print()
type()
help()  
min() / max()
bin() / hex() / oct()
id()
input()
int() / float()
pow()
round()
abs()  all()  any()  ascii()  bin()  bool()  bytearray()  bytes()  callable()  chr()  classmethod()  compile()  complex()  delattr()  dict()  dir()  divmod()  enumerate()  eval()  exec()  filter()  frozenset()  getattr()  globals()  hasattr()  hash()  help()  hex()  id()  input()  int()  isinstance()  issubclass()  iter()  len()  locals()  map()  max()  memoryview()  min()  next()  object()  oct()  open()  ord()  pow()  print()  property()  range()  repr()  reversed()  round()  set()  slice()  sorted()  str()  super()  tuple()  type()  vars()  zip()  __import__()
Not that kind of argument

An argument is something passed to a function, it’s what you want the function to work on.

Functions can be thought of as black boxes
help()
min() / max()

phew! for a second I thought I might have had to do math.
remember variables have a value, type, and id
int() / float()

introduction to *casting*
not that useful *yet* but will be when we get to strings

Not exactly rounding…more like truncation
pow()

an example of a function behind an operator
round() often nested:

print( round( min(2.5, 3.1, -7.9) ) )
WELCOME BACK!
Questions?
<table>
<thead>
<tr>
<th>Types</th>
<th>Functions</th>
<th>Flow Control</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>print()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>type()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>help()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min() / max()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>int() / float()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>round()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
they’re okay but what if I want something that isn’t built in, something that isn’t even in a library
User Defined Functions
def times_two(num):
    return num * 2

def is a key word that tells python you are starting the definition of a function

times_two is the name of my function

num is a parameter (or argument), it is an input passed to the function, not all functions require arguments

return is what the function is going to give back when finished

Let’s try out our code.
def times_two(num):
    return num * 2

Usually Python is flexible with regards to whitespace. The big exception is indents.
import java.util.Scanner;

class AddNumbers
{
    public static void main(String args[])
    {
        int x, y, z;
        System.out.println("Enter two integers to calculate their sum ");
        Scanner in = new Scanner(System.in);
        x = in.nextInt();
        y = in.nextInt();
        z = x + y;
        System.out.println("Sum of entered integers = "+z);
    }
}
User Defined Functions, Indent

```python
def times_two(num):
    return num * 2
```

so python uses indents to tell what code goes together

when the code stops being indented then python knows the function is complete

so
```python
def times_two(num):
    return num * 2
```

won’t work because the function times_two has no code
def times_two(num):
    return num * 2
	num is a parameter (or argument), it is an input passed to the function, not all functions require arguments

What exactly is “num”?

It’s essentially a variable, but one that only lives inside the function.

if we call times_two(4) then the first thing this code does is num = 4

*Arguments are what let us call functions on a variety of inputs*
def three_times_two():
    return 3 * 2

We've written a version of times_two that doesn't take an argument and instead is hardcoded for a specific value (i.e. fixed, not variable).

This works the same way as times_two(3) would but is obviously much less useful.
def times_two(num):
    return num * 2

def ptimes_two(num):
    print(num*2)

Do these do the same thing?  Hint: NO.
Note the color differences,
orange is a keyword,
purple is a built in function

What does the second function return?
print() exists to give information to a human being

returns exist to pass data around between parts of the program

x = max(2,3) works because the max function returns the largest of it’s arguments, that value (3 in this case) is then set as the value of the variable x

if max(2,3) printed a 3 rather than returning it then the above code wouldn’t work.

Same thing with print( max(2,3) )
The print knows what to print because max returned the value 3 to it when called.
remember the id() function?

what if we call
print( id(times_two) )
Programming as Data

A function is essentially a variable whose “value” is a series of steps on some input. This was a HUGE conceptual breakthrough.
WELCOME BACK!
Questions?
<table>
<thead>
<tr>
<th>Types</th>
<th>Functions</th>
<th>Flow Control</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>print()</td>
<td></td>
<td>def</td>
</tr>
<tr>
<td>float</td>
<td>type()</td>
<td></td>
<td>return</td>
</tr>
<tr>
<td></td>
<td>help()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min()/max()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>int()/float()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>round()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Familiar Refrain

This code

```python
def foo(a):
    return a * a
```

is a lot less easy to understand than this

```python
def square(num):
    return num * num
```

Just like with variables giving functions and arguments good names is a very good idea (which makes sense since arguments and functions really are sorts of variables)
Python visualizer:
http://www.pythontutor.com/visualize.html#mode=edit

def foo (A):
    doubleA = 2* A

print( doubleA )

Does this code work?
Things Go Wrong

Bugs!
Quick Intro to Entymology

1) Syntax Errors
2) Logic Errors
3) Runtime Errors

This list is in ascending order of suck.
Syntax Errors

```python
def times_two(num):
    return num * 2

def two():
    return 2

max(2 3)
```

Syntax error = your code sucks (or a typo)

Good news- easy to catch, easy to fix
def times_two(num):
    return num * 3

Logic error = your *computational thinking* sucks (or a typo)

May be easy or hard to spot, often frustrating to fix
“good” runtime error

```python
def times_two(nam):
    return num * 2
```

bad runtime error

```python
def divide_ten(num):
    return 10 / num
```

Runtime error = you didn’t think of an important case, or you referenced non existing variables

Can be nearly impossible to find without very good test cases. Often not that hard to fix.
Idle is an integrated Development Environment

IDEs are your friend

Generally Idle will point you to the vicinity of what’s wrong with Syntax and Runtime errors. It can’t do that with logic errors.

Testing your code is key.

We’ll talk more about testing and debug in week 5. So, you know, look forward to that.

No debug stuff on this quiz, this intro is meant to help you work on your homework.
Let’s build a function that evaluates the quadratic equation:
\[-3x^2 – 5x + 7\] for \(x = -2\)

FYI: In elementary algebra, a **quadratic equation** (from the Latin *quadratus* for "square") is any equation having the form
\[ax^2 + bx + c\]
where \(x\) represents an unknown, and \(a\), \(b\), and \(c\) are **constants** with \(a\) not equal to 0. If \(a = 0\), then the equation is **linear**, not quadratic. The parameters \(^{[1]}\) \(a\), \(b\), and \(c\) are called, respectively, the quadratic **coefficient**, the linear coefficient and the constant or free term. (wikipedia)
Okay, now let’s build a function that evaluates $-3x^2 - 5x + 7$ for any $x$.

Actually we just have to modify our previous function.

FYI: In elementary algebra, a **quadratic equation** (from the Latin *quadratus* for "square") is any equation having the form $ax^2 + bx + c$ where $x$ represents an unknown, and $a$, $b$, and $c$ are **constants** with $a$ not equal to 0. If $a = 0$, then the equation is **linear**, not quadratic. The parameters [1] $a$, $b$, and $c$ are called, respectively, the quadratic **coefficient**, the linear coefficient and the constant or free term. (wikipedia)
Okay, now let’s build a function that evaluates any quadratic equation: \( ax^2 + bx + c \) for any \( x, a, b, c \)

Actually we just have to modify our previous function

FYI: In elementary algebra, a quadratic equation (from the Latin quadratus for "square") is any equation having the form \( ax^2 + bx + c \) where \( x \) represents an unknown, and \( a, b, \) and \( c \) are constants with \( a \) not equal to 0. If \( a = 0 \), then the equation is linear, not quadratic. The parameters [1] \( a, b, \) and \( c \) are called, respectively, the quadratic coefficient, the linear coefficient and the constant or free term. (wikipedia)
Remember to Test

Let’s test our generic solution against the specific solution we first tried, it should match.

FYI: In elementary algebra, a **quadratic equation** (from the Latin *quadratus* for "square") is any equation having the form $ax^2 + bx + c$ where $x$ represents an unknown, and $a$, $b$, and $c$ are **constants** with $a$ not equal to 0. If $a = 0$, then the equation is **linear**, not quadratic. The parameters [1] $a$, $b$, and $c$ are called, respectively, the quadratic **coefficient**, the linear coefficient and the constant or free term. (wikipedia)