

6.189 Exam Session 5

Administrivia

Name:

MIT ID:

Instructions:

1. Err..complete the questions :). Warning: they do increase in difficulty.
2. No calculators, no laptops, etc.
3. When we ask for output, you DON'T have to write the spaces/newlines in.

Program Text:

```
print "X",  
print "X",
```

Output:

```
XX
```

Day 1: Variables, Operators, and Expressions

Tips/Notes:

- Variables are **typed** (this is a string, that's an integer, etc.) *type(x)* is a function that returns the type of its parameter.
- You can convert information from one type to another using the built-in conversion functions: *int(x)*, *str(x)*, *float(x)*, *bool(x)*. These tend to fail for conversions that don't make sense, e.g. *int("hello")* crashes.
- Every variable you create should have a meaning. Give your variables names that match their meaning: don't just name all your variables *a*, *b*, *c* (exception: if you're writing a quiz to test your students, only use names like *a*, *b*, *c*)

Problem 1: Neophyte (hey, this was just 4 days ago)

What is the output of the following code?

Program Text:

```
a = 5  
b = a + 7  
a = 10  
print b
```

Output:

`12`**Problem 2: Type Theory**What is the **type** of each of the following expressions (within the type function)?

Program Text:

`print type(5)`

Output:

`INT`

Program Text:

`print type("abc")`

Output:

`STRING`

Program Text:

`print type(True)`

Output:

`BOOLEAN`

Program Text:

`print type(5.5)`

Output:

`FLOAT`

Program Text:

`print type(12/27)`

Output:

`INT`

Program Text:

`print type(2.0/1)`

Output:

`FLOAT`

Program Text:

`print type(12 ** 3)`

Output:

`INT`

Program Text:

`print type(5 == "5")`

Output:

`BOOLEAN`

Program Text:

`a = str((-4 + abs(-5) / 2 ** 3) + 321 - ((64 / 16) % 4) ** 2)
print type(a)`

Output:

`STRING`

Problem 3: Expressive Expressions

What is the output of the following code?

Program Text:

```
print 5 == 5.0
```

Output:

```
TRUE1
```

Program Text:

```
print float(1/2)
```

Output:

```
0.0
```

Program Text:

```
print float(1)/2
```

Output:

```
0.5
```

Program Text:

```
print 5 == "5"
```

Output:

```
FALSE
```

Program Text:

```
print "sdf" != "sdf"
```

Output:

```
FALSE
```

Program Text:

```
print True and (False or not True)
```

Output:

```
FALSE
```

Program Text:

```
print str(53) + str(True)
```

Output:

```
53TRUE
```

Program Text:

```
a = 20
print 15-(a-15), ",",
a = 10
print 15-(a-15),
```

Output:

```
10,20
```

Day 3: Conditionals**Tips/Notes:**

- The **if** statement executes a sequence of statements only if some condition is true. This condition can be anything.

¹Scheme (the initial language taught at MIT before Python) took type to an extreme. *5.0* and *5* were not considered equal because one was an integer and the other was a float. You weren't even allowed to do $(4 + 6.0)$. Python is more rational – it treats the two values as equal.

- **elif / else** is optional. Remember that at most one block of statements is executed. **else** occurs if none of the above conditions are satisfied.

Problem 4: Basics

Consider the following code

Program Text:

```
a = ?

if a > 10 and a % 6 = 3:
    print "a",
elif a > 10 and a < 20:
    print "b",
else:
    print "c",
```

Give a value for a that would produce the following outputs. If no value of a would produce that output, write **none**.

Value of a :

NONE

Output:

a b

Value of a :

15,21,27,...

Output:

a

Value of a :

11,12, ..., 19 (EXCEPT 15)

Output:

b

Value of a :

ANY OTHER

Output:

c

Value of a :

NONE

Output:

victory is mine!

Problem 5: Trickier Duet

Removed due to time constraints.

Day 3: While loop

Tips/Notes:

- A **while** loop allows us to repeat a sequence of statements many times.
- You can use almost *anything* for the condition. Not every while loop has to be a simple counter.

Problem 6: This is not a loopy title

What is the output of the following code? *If the code does not terminate, write error.*

Program Text:

```
a = 5
while a < 8:
    print "X",
```

Output:

LOOPS FOREVER

Program Text:

```
a = -1
while a < 3:
    print "X",
    a = a + 1
```

Output:

XXXX

Program Text:

```
a = 1
while a % 7 != 0:
    if a % 2 == 0:
        print "0"
    if a == 2:
        print "X"
    a = a + 1
```

Output:

OXOO

Problem 7: Subtle variants

We're going to show you variants of the same code. Write the output of each code snippet.

Program Text:

```
keep_going = True
a = 0
b = 0
while keep_going:
    print "0"
    a = a + 5
    b = b + 7
    if a + b >= 24:
        keep_going = False
```

Output:

00

We rearranged the code within the while loop here.

Program Text:

```
keep_going = True
a = 0
b = 0
while keep_going:
    print "0"
    if a + b >= 24:
        keep_going = False
    a = a + 5
    b = b + 7
```

Output:

000

The remaining two variants are duplicates of the first two with `>=` replaced by `>`.

Program Text:

```
keep_going = True
a = 0
b = 0
while keep_going:
    print "0"
    a = a + 5
    b = b + 7
    if a + b > 24:
        keep_going = False
```

Output:

OOO

Program Text:

```

keep_going = True
a = 0
b = 0
while keep_going:
    print "0"
    if a + b > 24:
        keep_going = False
    a = a + 5
    b = b + 7

```

Output:

OOOO

Day 3: Nested loops

Tips/Notes:

- This isn't anything new, but we can put loops inside other loops. We can also do fairly crazy things: nest a **while** in an **if** in a **while** in another **while**.
- The **break** keyword exits the innermost loop.

Problem 8: Freebie!

What is the output of the following code? *If the code does not terminate, write error.*

Program Text:

```

a = 0
while a < 3:
    while True:
        print "X",
        break
    print "0",
    a = a + 1

```

Output:

XOXOXO

Problem 9: (insert evil laugh) ..is what I'd like to say. Still not that bad, though

What is the output of the following code? *If the code does not terminate, write error.*

Program Text:

```
a = 1
while a < 3:
    while a < 3:
        print "0",
        a = a + 1
```

Output:

LOOPS FOREVER

Program Text:

```
a = 1
while a < 3:
    if a % 2 == 0:
        b = 1
        while b < 3:
            print "X",
            b = b + 1
    print "0",
    a = a + 1
```

Output:

OXXO

Extra Credit (mainly due to time constraints.) Solve this if you finish early!

Program Text:

```
a = 1
while a < 3:
    b = 1
    while b < 3:
        if a == 2:
            print "X",
            print "0",
            b = b + 1
        print "0",
```

Output:

LOOPS FOREVER (DUE TO TYPO.) FIXED, IT WOULD BE
OOOXOXOO

Day 2: Functions

Tips/Notes:

- A **function** is just a named sequence of statements. We usually *define* functions at the beginning of code – definitions just associate the name with the sequence of statements.
- Functions can take parameters (within the parenthesis suffix) and can return information via **return**
- **return** is NOT a function. Like **if**, **while**, .. its a *keyword*: a basic command of the language.
- You can find out more information about functions using the *help(x)* function, e.g. *help(sqrt)*.
Remember to write *from math import ** first.

Problem 10: Sanity Check

What is the output of the following code? *If the code does not terminate, write error.*

Program Text:

```
def f(a):  
    a = a + 5  
    return a  
  
b = 0  
f(b)  
print b, ",",  
b = f(b)  
print b
```

Output:

0,5

Problem 11: Last but not least (somewhere in the middle)

You know that functions can call other functions, right? Here's an interesting fact – functions can call themselves!

Program Text:

```
def f(x):  
    print "X",  
    if x <= 1:  
        return 1  
    else:  
        return x+f(x-1)
```

Fill out the following table for the return value and output of each function call.

| | | |
|-----------------------------------|---------------------------------|-----------------------------------|
| Function call: | Return value: | Output: |
| <input type="text" value="f(1)"/> | <input type="text" value="1"/> | <input type="text" value="X"/> |
| Function call: | Return value: | Output: |
| <input type="text" value="f(2)"/> | <input type="text" value="3"/> | <input type="text" value="XX"/> |
| Function call: | Return value: | Output: |
| <input type="text" value="f(3)"/> | <input type="text" value="6"/> | <input type="text" value="XXX"/> |
| Function call: | Return value: | Output: |
| <input type="text" value="f(4)"/> | <input type="text" value="10"/> | <input type="text" value="XXXX"/> |

Extra stuff

If you were reasonably comfortable with this test, here is some extra stuff that you might find useful (you can rip this page out if you like.)

- Don't forget about using `#` to write notes/comments in your code!
- Instead of always writing `a = a + 5` or `a = a / 7`, you can use the shorthand `a += 5` and `a /= 7`. Be careful not to get confused by this notation:

Program Text:

```
-
_a = 5
-b = a
-a += 3
-print b #still 5
```

- String stuff

- You can use either a single quotation mark (`'`) or a double quotation mark for strings. The only difference is either one can contain the other (`'This is a "test"'` is valid, `"This is a 'test'"` is valid, `"This is a ""test""` is not valid.)
- You can use `"\n"` to insert a newline (Enter key) in a string.
- You can define multi-line strings using the triple-quotation `"""` operator.

Program Text:

```
-
_print """This is a sentence.\nThis sentence is on the second line.
-This sentence is on the third line.
-This is on the fourth."""
-
```

- You can write a description of a function by putting a string as the first line of the function.

Program Text:

```
-
def hypo(a,b):
- """This function returns the length of the hypotenuse defined by
-   the right triangle with side lengths a,b"""
- return sqrt(a*a + b*b)
```

Try calling `help(hypo)` on the function you just wrote!

- **print** is a keyword, not a function (like **if**, **while**, **return**.) There's no particularly good reason why it's a keyword – future versions of Python are changing it into a function.