COMP1405C, Midterm Exam, Fall 2012, 1h20m

All code on this exam is written in the Python programming language and you may assume that it is being run with the parameters `-Q new` or, equivalently, that it has been preceded with `from __future__ import division`.

Answer all questions on the Scantron sheet. Select exactly one answer for each question. In cases where there is more than once correct answer, select the most correct/precise answer. Write your name and student number on the top of this question sheet. Submit both this question sheet and your Scantron answer sheet.

1. What does the code `2 + 2 * 3` evaluate to?
   (a) 4 (b) 8 (c) 12 (d) 16 (e) 64

2. What does the code `2/3 == 2//3` evaluate to?
   (a) True (b) False (c) 1.5 (d) No (e) Yes

3. If `a` and `b` are both of type `float`, then the expression `a * b` is of type
   (a) `float` (b) `int` (c) `string` (d) `bool` (e) `object`

4. If `a` and `b` are both of type `float`, then the expression `a < b` is of type
   (a) `float` (b) `int` (c) `string` (d) `bool` (e) `object`

5. Consider the following code:

```
def format_name(fn, ln, ap):
    return "%s, %s, %s" % (ln, fn, ap)
```

   Calling `format_name('Pat', 'Morin', 'Dr.')` will return
   (a) the triple `('Dr.', 'Pat', 'Morin')`
   (b) the triple `('Morin', 'Pat', 'Dr.')`
   (c) the string `'Dr. Pat Morin'`
   (d) the string `'Morin, Pat, Dr.'`
   (e) none of the above

6. Consider the following code:

```
def in2ftin(i):
    return i//12, i%12
```

   Calling `in2ftin(74)` will return
   (a) the pair `(6, 2)`
   (b) the string `'6, 2'`
   (c) the number `6.2`
   (d) the number `6.166666667`
   (e) none of the above

7. For any integer `n` $\geq 0$ following two functions have the same return value:

```
def stars(n):
    s = ""
    for i in xrange(0,n):
        s += "*"
    return s

def stars(n):
    return n * '*'
```

(a) true (b) false

8. For any integer `n` $\geq 0$ following two functions have the same return value:

```
def factorial2(n):
    return reduce(lambda x,y: x*y, range(2,n+1))

def factorial2(n):
    return reduce(lambda x,y: x*y, range(1,n+1))
```

(a) true (b) false

9. Consider the following code:

```
def foo(a):
    return reduce(lambda x,y: str(x)+str(y), a)
```

(a) This function takes a list and return an list
(b) This function takes a list of numbers and returns their sum (a number)
(c) This function takes a list of strings and returns their concatenation (a string)
(d) This function takes a list of anything, converts each element to a string, and returns their concatenation (a string)

10. Consider the following code:

```
def bar(a):
    return [ x**3 for x in a ]
```

Calling `bar([1, 2, 3, 4])` will return
(a) the integer 100
(b) the list [1, 4, 9, 16]
(c) the integer 100
(d) the list [1, 8, 27, 64]
(e) a list containing 64 entries

11. Consider the following code:

```
def nice_hour(h):
    if h == 12: return "noon"
    names = [ "midnight", "one", "two", "three", "four", "five",
              "six", "seven", "eight", "nine", "ten", "eleven" ]
    return names[h%12]
```

Calling `nice_hour(17)` will

(a) return the number `5`
(b) return the string `'five'`
(c) return an array
(d) generate a `NameError`
(e) generate an `IndexError`

12. Calling `nice_hour(29)` will

(a) return the number `5`
(b) return the string `'five'`
(c) return the string `'seven'`
(d) generate a `NameError`
(e) generate an `IndexError`

⋆ The next few questions build upon the following code:

```
class Person(object):
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name

    def __eq__(self, other):
        return self.first_name == other.first_name \
            and self.last_name == other.last_name

    def __str__(self):
        return "%s %s" % (self.first_name, self.last_name)
```

13.
```
pm1 = Person('Pat', 'Morin')
pm2 = Person('Pat', 'Morin')
print pm1 == pm2
print pm1 is pm2
```

(a) prints `True` then `True`
(b) prints `False` then `True`
(c) prints `True` then `False`
(d) prints `False` then `False`

14.
```
pm1 = Person('Pat', 'Morin')
pm2 = pm1
print pm1 == pm2
print pm1 is pm2
```
(a) prints `True` then `True`
(b) prints `False` then `True`
(c) prints `True` then `False`
(d) prints `False` then `False`

⋆ The next few questions add the following class:

```
class Musician(person):
    def __init__(self, first_name, last_name, instrument):
        super(Musician, self).__init__(first_name, last_name)
        self.instrument = instrument

    def __str__(self):
        return "%s %s (%s player)" \
                % (self.first_name, self.last_name, self.instrument)
```

15.
```
pm1 = Person('Pat', 'Morin')
pm2 = Musician('Pat', 'Morin', 'trombone')
print pm1
print pm2
```
(a) prints `Pat Morin` then `Pat Morin`
(b) prints `Pat Morin (trombone player)` then `Pat Morin`
(c) prints `Pat Morin` then `Pat Morin (trombone player)`
(d) prints `Pat Morin (trombone player)` then `Pat Morin (trombone player)`
(e) generates a `NameError`

16.
```
pm1 = Person('Pat', 'Morin')
pm2 = Musician('Pat', 'Morin', 'trombone')
print pm1 == pm2
print pm1 is pm2
```
(a) prints `True` then `True`
(b) prints `False` then `True`
(c) prints `True` then `False`
(d) prints `False` then `False`
(e) generates a `NameError`

17.
```
def negated(a):
    """Return a list having only the values x in a where -x is also in a"""
    return [x for x in a if -x in a]
```
(a) This code is incorrect — it doesn't do what its comment suggests
(b) This code is correct but slow, it will take a long time if `a` is a long list
(c) This code is correct and fast, even when `a` is a long list

18.
```
def unique(a):
    return list(set(a))
```

(a) This code is fast, even when `a` is a long list
(b) This code returns a list containing each distinct element of `a` exactly once
(c) The elements in the returned list appear in the order of their first occurrence in `a`
(d) Both (a) and (b)
(e) All of the above

19.
```
def ananyms1(words):
    return { w for w in words if w[::-1] in words }

def ananyms2(words):
    wordset = set(words)
    return { w for w in wordset if w[::-1] in words }
```

Suppose `words` is a list containing 100 words. Then

(a) `ananyms1(words)` will take a fraction of a second to execute and `ananyms2(words)` will take a fraction of a second to execute
(b) `ananyms1(words)` will take more than a minute to execute and `ananyms2(words)` will take a fraction of a second to execute
(c) `ananyms1(words)` will take more than a minute to execute and `ananyms2(words)` will take more than a minute to execute
(d) `ananyms1(words)` will take a fraction of a second to execute and `ananyms2(words)` will take more than a minute to execute

20. Suppose `words` is a list containing 100000 words. Then

(a) `ananyms1(words)` will take a fraction of a second to execute and `ananyms2(words)` will take a fraction of a second to execute
(b) `ananyms1(words)` will take more than a minute to execute and `ananyms2(words)` will take a fraction of a second to execute
(c) `ananyms1(words)` will take more than a minute to execute and `ananyms2(words)` will take more than a minute to execute
(d) `ananyms1(words)` will take a fraction of a second to execute and `ananyms2(words)` will take more than a minute to execute

21. (Bonus) If you choose an answer to this question at random what is the chance you will be correct?
(a) 20% (b) 40% (c) 60% (d) 80% (e) 20%