Clearly print your email id:

Clearly print your name:

Pledge:

Part 1 answers

a)
b)
c)
d)
e)
f)
g)
h)
i)
j)

This page is almost blank

Part 1: short answers (20 points)

1. Consider the following function definitions

```
def f( ) :
   x = 1

def g( x ) :
   x = 1

def h( x, y ) :
   x, y = y, x
   return x, y
```

a) What is the return value of function f()? Your answer should not provide an explanation.

b) In function f() is x a variable? Your answer should be yes or no with no explanation given.

c) In function g() is x an argument? Your answer should be yes or no with no explanation given.

d) In function h() is x a parameter? Your answer should be yes or no with no explanation given.

e) What does the following code segment output()? Your answer should be a single value with no explanation given.

```
x = 0
f()
print( x )
```

f) What does the following code segment output()? Your answer should be a single value with no explanation given.

```
x = 0
g( x )
print( x )
```

g) What does the following code segment output()? Your answer should be a single value with no explanation given.

```
x = 0
x = g( x )
print( x )
```

h) What does the following code segment output()? Your answer should be two values with no explanation given.

```
x, y = 0, 1
h( x , y )
print( x, y )
```

i) Explain your answer to part (h). Be specific and terse.

j) What is the purpose of a return statement? Be specific and terse.

Part 2: Programming (80 points)

2. Implement module *cd.py*, which is concerned with carbon-14 dating. The module defines a function `sample()`. The function has a single decimal parameter `d`, which is a carbon 12 to carbon 14 decay ratio. The function returns an *integer* estimate of the age of a fossil with a such a ratio. The carbon-14 decay formula for estimating age is:

   $$age = \log(\, d\,) \cdot -8268.3982$$

   Because the age is an estimate, it is always truncated to *integer*. For your information, the `math.log()` function should prove useful.

   The output of the built-in tester is:

```
sample( 0.35 ) = 8680
sample( 0.005 ) = 43808
sample( 1.0 ) = 0
```

3. Implement module *lwv.py*. The module defines a function `less()`. The function has a single string parameter `w`. The function returns the logical value `True` if `w` contains a lowercase vowel; that is one of `'a'`, `'e'`, `'i'`, `'o'`, or `'u'`. If instead, `w` does not contain a lowercase vowel, the function returns the logical value `False`.

   The output of the built-in tester is:

```
less( oxen ) = True
less( urchin ) = True
less( mink ) = True
less( rabbit ) = True
less( lynx ) = False
```

4. Implement module *iee.py*. The module defines a function `process()`. The function has two integer parameters `x` and `y`. If both parameters are positive, the function returns the sum `x + y`; if instead, the parameters are both negative, the function returns the difference `x - y`; otherwise, the function returns the *integer* quotient `x // y`.

   The output of the built-in tester is:

```
process( 12 , 13 ) = 25
process( -9 , -2 ) = -7
process( 16 , -2 ) = -8
process( 0 , -5 ) = 0
```

5. Implement module *tobe.py*. The module defines a function `series()`. The function has a single integer parameter `n`. The function returns a list of `n` integer values. The values are respectively:

   $2^0, 2^1, 2^2, 2^3, \dots\ 2^{n-1}$.

   The output of the built-in tester is:

```
series( 0 ) = []
series( 1 ) = [1]
series( 4 ) = [1, 2, 4, 8]
series( 9 ) = [1, 2, 4, 8, 16, 32, 64, 128, 256]
```

6. Implement module *xmum.py*. The module defines a function maxi(). The function has two integer list parameters x and y of equal length. The function does not modify its parameters. The function returns a new list of n integer values, where the element value at index *i* in the new list is the maximum of the corresponding element values in x and y.

   The built-in tester runs four tests using the following to initialize parameters x and y respectively.

```
x1 = [5, 6, 6]   x2 = [7, 3, 5, 5]    x3 = [4, 7, 7, 8, 2, 3]    x4 = []
y1 = [1, 4, 6]   y2 = [4, 8, 2, 7]    y3 = [3, 8, 4, 4, 8, 5]    y4 = []
```

   The output of the built-in tester is:

```
maxi( x1, y1 ) = [5, 6, 6]
maxi( x2, y2 ) = [7, 8, 5, 7]
maxi( x3, y3 ) = [4, 8, 7, 8, 8, 5]
maxi( x4, y4 ) = []
```

7. Implement module *atse.py*. The module defines a function dt(). The function has one dataset parameters d. The rows of d are lists of integer values. The function does not modify its parameter. The function returns a new dataset with the same number rows as d. The *i*th row in the new dataset is a list of four values [v1, v2, v3, v4], where

   - v1 is the length of row *i* in d,
   - v2 is the minimum value in row *i* in d,
   - v3 is the integer average of the values in row *i* in d.
   - v4 the maximum value in row *i* in d.

   The built-in tester runs four tests using the following to initialize parameter d respectively.

```
d1 = [[5, 6, 5], [7, 3, 5, 5], [4, 7, 7, 8, 2, 3]]
d2 = [[1, 4, 6], [4, 8, 2, 7], [3, 8, 4, 4, 8, 5]]
d3 = [[1], [2, 4], [5, 3, 7, 7, 3, 3]]
d4 = [[3, 1, 4, 1, 5, 9]]
```

   The output of the built-in tester is:

```
dt( d1 ) = [[3, 5, 5, 6], [4, 3, 5, 7], [6, 2, 5, 8]]
dt( d2 ) = [[3, 1, 3, 6], [4, 2, 5, 8], [6, 3, 5, 8]]
dt( d3 ) = [[1, 1, 1, 1], [2, 2, 3, 4], [6, 3, 4, 7]]
dt( d4 ) = [[6, 1, 3, 9]]
```

Notices

- Based on your past educational achievements, I expect you to do well on this test.
- Answer the questions in any order that you want.

Test rules

- Before you leave the room, check that you uploaded all of your solutions. Do not ask afterwards whether you can submit a forgotten solution.
- This pledged exam is closed notes. The only device you may access during the test is your laptop.
- Uploading after you leave the room means you withdrawing from the class with a test score of 0.
- Any cheating can result in failing the class and the incident being referred to the Honor Committee.
- Do not access class examples artifacts, web solutions, or your own past assignments during the test; that is, the only code you may access or view are ones that you develop for this test.
- The only windows allowed on your laptop are PyCharm and a single browser with tabs reachable from class website.

PyCharm

- PyCharm can be used for developing the modules to be submitted. It ***cannot be used*** for the short answer questions of part 1.

Modules

- Modules should follow class programming practices; e.g., whitespace, identifier naming, and commenting if you think it is needed, etc.
- Whether a module function is runnable is important.
- None of your code should produce output. Comment out or delete all debugging `print()` statements before submitting.