1 Functions

Questions

1.1 Determine what the Python interpreter will output given the following lines of code.

```python
>>> from operator import add, mul
>>> mul(add(5, 6), 8)

>>> print('x')

>>> y = print('x')

>>> print(y)

>>> print(add(4, 2), print('a'))
```

1.2 Determine what the Python interpreter will output given the following lines of code.

```python
>>> def foo(x):
...     print(x)
...     return x + 1

>>> def bar(y, x):
...     print(x - y)

>>> foo(3)

>>> bar(3)

>>> bar(6, 1)

>>> bar(foo(10), 11)
```
2 Control

Questions

2.1 Which numbers will be printed after executing the following code?

```python
n = 0
if n:
    print(1)
elif n < 2
    print(2)
else:
    print(3)
print(4)
```

2.2 WWPD (What would Python Display) after evaluating each of the following expressions?

```python
>>> 0 and 1 / 0

>>> 6 or 1 or "a" or 1 / 0

>>> 6 and 1 and "a" and 1 / 0

>>> print(print(4) and 2)

>>> not True and print("a")
```

2.3 Define a function, count_digits, which takes in an integer, n, and counts the number of digits in that number.

```python
def count_digits(n):
    ... 
    >>> count_digits(4)
    1
    >>> count_digits(12345678)
    8
    >>> count_digits(0)
    0
    ...
```

*Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.*
2.4 Define a function, count_matches, which takes in two integers n and m, and counts the number of digits that match.

```python
def count_matches(n, m):
    
    >>> count_matches(10, 30)
    1
    >>> count_matches(12345, 23456)
    0
    >>> count_matches(121212, 123123)
    2
    >>> count_matches(111, 11) # only one’s place matches
    2
    >>> count_matches(101, 10) # no place matches
    0
    
```

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.
3 Environment Diagrams

Questions

3.1 Draw the environment diagram for evaluating the following code

```python
def f(x):
    return y + x
y = 10
f(8)
```

3.2 Draw the environment diagram for evaluating the following code

```python
def dessef(a, b):
    c = a + b
    b = b + 1
b = 6
dessef(b, 4)
```
3.3 Draw the environment diagram for evaluating the following code

```python
def foo(x, y):
    foo = bar
    return foo(bar(x, x), y)

def bar(z, x):
    return z + y

y = 5
foo(1, 2)
```

3.4 Draw the environment diagram for evaluating the following code

```python
def spain(japan, iran):
    def world(cup, egypt):
        return japan-poland
    return iran(world(iran, poland))

def saudi(arabia):
    return japan + 3

japan, poland = 3, 7
spain(poland+1, saudi)
```
3.5 Draw the environment diagram for evaluating the following code

cap = 9
hulk = 3

def marvel(cap, thor, avengers):
    marvel = avengers
    iron = hulk + cap
    if thor > cap:
        def marvel(cap, thor, avengers):
            return iron
    else:
        iron = hulk
    return marvel(thor, cap, marvel)

def iron(man):
    hulk = cap - 1
    return hulk

marvel(cap, iron(3), marvel)
4 Higher Order Functions

Questions

4.1 What do lambda expressions do? Can we write all functions as lambda expressions? In what cases are lambda expressions useful?

4.2 Determine if each of the following will error:

```python
>>> 1/0
>>> boom = lambda: 1/0
>>> boom()
```

4.3 Express the following lambda expression using a `def` statement, and the `def` statement using a lambda expression.

```python
pow = lambda x, y: x**y
```

```python
def foo(x):
    def f(y):
        def g(z):
            return x + y * z
        return g
    return f
```

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.
4.4 Draw Environment Diagrams for the following lines of code

```python
square = lambda x: x * x
higher = lambda f: lambda y: f(f(y))
higher(square)(5)

a = (lambda f, a: f(a))(lambda b: b * b, 2)
```

*Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.*
4.5 Write **make_skipper**, which takes in a number n and outputs a function. When this function takes in a number x, it prints out all the numbers between 0 and x, skipping every nth number (meaning skip any value that is a multiple of n).

```python
def make_skipper(n):
    "":
    >>> a = make_skipper(2)
    >>> a(5)
    1
    3
    5
    "":
```
5 Extra Questions

Questions

5.1 Define a function, ordered_digits, which takes in a positive integer, \( x \), and returns True if the (base 10) digits of \( x \) are in non-decreasing order, and False otherwise.

```python
def ordered_digits(x):
    
    >>> ordered_digits(5)
    True
    >>> ordered_digits(11)
    True
    >>> ordered_digits(127)
    True
    >>> ordered_digits(1357)
    True
    >>> ordered_digits(21)
    False
    >>> result = ordered_digits(1375) # Return, don't print
    >>> result
    False
    
```

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.
Define a function, cycle, which takes in three functions, f1, f2, f3, and returns a function that takes in an integer n and returns a function that takes in an integer x, and returns the result of f1(x) the first time it’s called, f2(x) the second time it’s called, f3(x) the third time it’s called, and then cycles back to f1(x) the fourth time it’s called, and so on.

```python
def cycle(f1, f2, f3):
    """Returns a function that is itself a higher-order function."
    def add1(x):
        return x + 1
    def times2(x):
        return x * 2
    def add3(x):
        return x + 3
    my_cycle = cycle(add1, times2, add3)
    identity = my_cycle(0)
    identity(5)
    add_one_then_double = my_cycle(2)
    add_one_then_double(1)
    do_all_functions = my_cycle(3)
    do_all_functions(2)
    do_more_than_a_cycle = my_cycle(4)
    do_more_than_a_cycle(2)
    do_two_cycles = my_cycle(6)
    do_two_cycles(1)
    """
```

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.
Define a function, is_palindrome, which takes in an integer, \( n \), and returns True if \( n \) is a palindrome and False otherwise.

```python
def is_palindrome(n):
    """
    Fill in the blanks '_____' to check if a number
    is a palindrome.
    >>> is_palindrome(12321)
    True
    >>> is_palindrome(42)
    False
    >>> is_palindrome(2015)
    False
    >>> is_palindrome(55)
    True
    """
    x, y = n, 0
    f = lambda: _______________________________
    while x > 0:
        x, y = _____________, f()
    return y == n
```