

Name: \_\_\_\_\_ Section: 1 2 3 4

Use this quiz to help you prepare for the Paper-and-Pencil portion of Test 1. **Answer all questions.** Make additional notes as desired. **Not sure of an answer?** Ask your instructor to explain in class and revise as needed then. *Please print two-sided if practical.*

Throughout, where you are asked to “circle your choice”, you can circle or underline it (whichever you prefer).

1. Consider the **secret** function defined to the right. What are the values of:

```
def secret(x):  
    y = (x + 1) ** 2  
    return y
```

a. **secret(2)** \_\_\_\_\_

b. **secret(secret(2))** \_\_\_\_\_

2. Consider the **mystery** function defined to the right. What are the values of:

```
def mystery(x, y):  
    result = x + (3 * y)  
    return result
```

a. **mystery(5, 2)** \_\_\_\_\_

b. **mystery(2, 5)** \_\_\_\_\_

3. Consider the code snippets defined below. They are contrived examples with poor style but will run without errors. For each, what does it print when *main* runs? (Each is an independent problem. Pay close attention to the order in which the statements are executed.)

```
def main():  
    x = 5  
    foo(x)  
    print(x)  
  
def foo(x):  
    print(x)  
    return x ** 3
```

Prints: \_\_\_\_\_

\_\_\_\_\_

```
def main():  
    x = 5  
    y = foo(x)  
    print(y)  
  
def foo(x):  
    x = 10  
    print(x)  
    return x ** 3
```

\_\_\_\_\_

\_\_\_\_\_

```
def main():  
    x = 5  
    x = foo(x)  
    print(x)  
  
def foo(x):  
    print(x)  
    return x ** 3
```

\_\_\_\_\_

\_\_\_\_\_

4. What is the value of each of the following expressions?

`7 // 4`

`3.0 // 4.0`

`3 / 4`

`7 % 2`

`7 ** 2`

`'fun' + 'ny'`

`'hot' * 5`

5. For each of the following code snippets, what does it print?  
(Write each answer directly below its code snippet.)

```
for j in range(0, 8, 2):
    print(j)
```

```
a = 10
for k in range(3, 6):
    a = a + k
    print(a, k)
```

```
b = 0
for k in range(10, 2, -1):
    if (k % 3) == 2:
        b = b + 1
        print(b, k)
print(b)
```

6. For each of the following Boolean expressions, indicate whether it evaluates to **True** or **False** (circle your choice):

<i>True</i>	<i>False</i>	<code>not (5 &lt; 7)</code>
<i>True</i>	<i>False</i>	<code>(7 &lt; 5) or not (5 &lt; 7)</code>
<i>True</i>	<i>False</i>	<code>(3 != 4) and (3 == 3)</code>
<i>True</i>	<i>False</i>	<code>(6 &lt;= 6) or (3 == 2)</code>
<i>True</i>	<i>False</i>	<code>(6 &lt;= 6) and (3 == 2)</code>
<i>True</i>	<i>False</i>	<code>not not False</code>

7. For each of the following, write a **range** expression that produces the given sequence:

4, 5, 6, 7, 8

40, 50, 60, 70, 80

-6, -5, -4

-6, -7, -8

8. What gets printed when *main* is called in the program shown to the right? (Pay close attention to the order in which the statements are executed. **Write the output in a column to the left of the program.**)

Output

```
def main():  
    a = 2  
    b = 3  
  
    foo1()  
    print(a, b)  
  
    foo2(a, b)  
    print(a, b)  
  
    foo3(a, b)  
    print(a, b)  
  
def foo1():  
    a = 88  
    b = 99  
  
def foo2(a, b):  
    a = 400  
    b = 500  
  
def foo3(x, y):  
    x = 44  
    y = 55
```

9. True or False: As a **user** of a function (that is, as someone who will **call** the function), you don't need to know how the function is **implemented**; you just need to know the **specification** of the function. **True False** (circle your choice)

10. List **two** reasons why functions are useful and important.

Reason 1: \_\_\_\_\_

Reason 2: \_\_\_\_\_

11. **float** versus **int**:

a. Write two Python constants – one an integer (**int**) and one a floating point number (**float**) – that clearly shows the difference between the **int** and **float** types.

b. A Python **int** can have an arbitrarily large number of digits.    **True**    **False**  
(circle your choice)

c. A Python **float** can represent an arbitrarily large number.    **True**    **False**  
(circle your choice)

d. There is a limit to the number of significant digits a Python **float** can have.    **True**    **False**  
(circle your choice)

12. **int** versus **str**: What does each of the following code snippets print or cause to happen if the user types **5** in each case? (Write each answer to the side of its code snippet.)

```
x = input('Enter an integer: ')
print(x * 3)
```

```
y = int(input('Enter an integer: '))
print(y * 3)
```

```
z = input('Enter an integer: ')
print(z / 3)
```

13. Does the following function meet its specification? If not, why not?

```
def get_number(x):  
    """  
    Returns x squared plus x cubed, for the given x.  
    For example, if x is 5, returns (5 ** 2) + (5 ** 3), which is 150.  
    """  
    answer = (x ** 2) + (x ** 3)  
    print(answer)
```

14. Does the following function meet its specification? If not, why not?

```
def get_number(x):  
    """  
    Returns x squared plus x cubed, for the given x.  
    For example, if x is 5, returns (5 ** 2) + (5 ** 3), which is 150.  
    """  
    answer = (x ** 2) + (x ** 3)  
    print(answer)  
    return answer
```

15. Does the following function meet its specification? If not, why not?

```
def test_get_number(x):  
    """ Tests the get_number function. """  
    answer1 = get_number(5)  
    answer2 = get_number(1)  
    answer3 = get_number(2)
```

16. Consider a function whose name is ***print\_string*** that takes two arguments as in this example:

```
print_string('Robots rule!', 4)
```

The function should print the given string the given number of times. So, the above function call should produce this output:

```
Robots rule!
```

```
Robots rule!
```

```
Robots rule!
```

```
Robots rule!
```

Write (in the space below) a complete implementation, *including the header (def) line*, of the above ***print\_string*** function.