

NAME: ANSWER KEY

The rules:

- Relax!
- Open book.
- Closed notes.
- All work must be your own. Merely *looking* at the work of others is cheating and may carry all the consequences associated with cheating.
- Generally the interactive prompt will not be shown even if the code is assumed to have been entered interactively.
- Neatness counts. If I can't easily read it, you won't get credit. Pay attention to clear indentation where indentation is necessary.
- For questions whose answer is either a `float` or an `int`, you must indicate the type of the answer by either having or not having a decimal point.
- You may use the "standard" scientific functions of a calculator. Programmed calculations are not permitted. (A programmable calculator is allowed, but you are not allowed to program it or use pre-installed programs.)
- This is not a race! **If you submit your test before 50 minutes from the start of the exam, the number of incorrect points will be multiplied by three.** Take your time and check your work! (If you think you have time to kill, why not read the book?)
- The value of each question is indicated within brackets, e.g., [10]. (Questions with equal value are not necessarily of equal difficulty.)
- **Note!** Problems that I think require you to be cautious (or even extremely cautious) are marked with a "!" within the brackets.

1. [4] The following code is executed. What is the output of the `print` statement?

```
print(3 + 5 , "3" + "5")
```

- (a) `3 + 5 , "3" + "5"`
 - (b) `8 35` ← **ANSWER**
 - (c) `8 8`
 - (d) This produces an error.
2. [4] The following command is executed

```
a = eval(input("Enter: "))
```

In response to this the user types:

```
3 * 5
```

What would be the value of `a`?

- (a) `'3 + 5'`
 - (b) `3`
 - (c) `15` ← **ANSWER**
 - (d) This code produces an error.
3. [4] To what value does the following code set `a`?

```
a = list(range(3, -1, -1))
```

- (a) `[3, 2, 1, 0]` ← **ANSWER**
 - (b) `[3, 2, 1, 0, -1]`
 - (c) `[3, -1, -1]`
 - (d) This code produces an error.
4. [4] Consider the assignment statements:

```
a = [1, 2, 3, 4, 5]
```

```
b = a[1] ** a[3]
```

What is the value of `b`?

- (a) `1`
- (b) `3`
- (c) `16` ← **ANSWER**
- (d) This produces an error.

5. [4] What is the purpose of the following code?

```
for i in range(2, 10, 2):  
    print(i)
```

- (a) Prints all numbers (inclusively) in the range [2, 10].
- (b) Prints all even numbers less than 10. \Leftarrow **ANSWER** This answer is also acceptable
- (c) Prints all odd numbers in range [2, 10].
- (d) This produces an error.
- (e) None of the above. \Leftarrow **ANSWER** Technically, this is the correct answer

6. [4!] The following commands are executed

```
def cube(x):  
    print(x ** 3)
```

```
b = cube(3.0) + 2
```

What is the resulting value of b?

- (a) 'x ** 3 + 2'
- (b) '3.0 ** 3 + 2'
- (c) 27.0
- (d) 29.0
- (e) This produces an error. \Leftarrow **ANSWER** Note that None cannot be added to a number

7. [4] The following commands are executed

```
def cube(x):  
    return(x ** 3)
```

```
b = cube(2.0) + 2
```

What is the resulting value of b?

- (a) x ** 3 + 2
- (b) '2.0 ** 3 + 2'
- (c) 8.0
- (d) 10.0 \Leftarrow **ANSWER**
- (e) This produces an error.

8. [4] Suppose that the integer variable x currently has the value 46. What statement would cause the value of x to become 15.0?

- (a) x = x / 3
- (b) x = x // 3.0 \Leftarrow **ANSWER**
- (c) y = x // 3
- (d) None of the above.

9. [4] To what value is the variable a set by the following code?

```
def f(my_list):  
    sum = 0  
    for element in my_list:  
        sum = sum + element  
    return sum / len(my_list)
```

```
a = f([1, 5, 7, 11])
```

- (a) 24
- (b) 6.0 ← **ANSWER**
- (c) 1.5
- (d) None

10. [4] The following code is executed. What is the output of the print statement in the following code?

```
def add(a, b):  
    return a + b  
  
print(add(3, add(4, 5)))
```

- (a) 7
- (b) 12 ← **ANSWER**
- (c) 'add(3, 9)'
- (d) This produces an error.

11. [4] The following code is executed. What is the output of the print statement in the following code?

```
def f(x, y):  
    return x * y  
  
a = f(2, 3)  
b = f("2", 3)  
print(a, b)
```

- (a) 6 6
- (b) 6 '2 * 3'
- (c) 6 222 ← **ANSWER**
- (d) This produces an error.

12. [4] The following code is executed. What is the output of the `print` statement in the following code?

```
def f(x, y):
    x = x + 1
    y = y + 1
    return x * y

x = 2
y = 3
print(f(x, y), x, y)
```

- (a) 6 3 4
- (b) 12 3 4
- (c) 12 2 3 **← ANSWER**
- (d) This produces an error.

13. [4] Consider the following code where a simultaneous assignment is used in the for-loop. What is the output produced by this code?

```
x = 0
y = 0
for i in range(3):
    x, y = x + 1, x + y
    print(y, end=" ")
```

(Be sure you think about this!)

- (a) 0 1 3 **← ANSWER**
- (b) 1 2 4
- (c) 1 3 6
- (d) 0 1 2

14. [4] What is the output produced by the following?

```
result = 1
for i in [2, 4, 6]:
    result = result * i
print(result)
```

- (a) 0
- (b) 48 **← ANSWER**
- (c) 1
- (d) None of the above.

15–22: [16 (2 each)] Assume the following code has been executed:

```
a = 2.0
b = a + 2 * 2
c = a * 2 / 2 ** 2
d = (a - 1) // 4
e = int((a - 7) / 3)
f = int(a) / 4
g = float(a // 4)
h = int(2.5) + int(a) // 4
```

To what values are the following variables set? Note: When the result is a `float`, show the decimal point in your answer regardless of whether the fractional part is zero or not. If the result is an `int`, your answer should not have a decimal point.

15. `b` = 6.0

16. `c` = 1.0

17. `d` = 0.0

18. `e` = -1

19. `f` = 0.5

20. `g` = 0.0

21. `h` = 2

22–25. A programmer is asked to write a function that displays the x and y coordinates of five points that are on a line. As arguments to this function one specifies the slope (m) of the line and the y -intercept (b). (Recall that the equation for a line is $y = mx + b$ where y and x are the “dependent” and “independent” variables, respectively.) The programmer is told to write the function so that it prints the x - y coordinates of points on this line when x has integer values in the range $[1, 5]$ (inclusively). Let’s call this function `five_points_on_line()`.

For example, if the arguments were 3 and 4 (meaning the equation for the line is $y = 3x + 4$), the function should print the following pairs of numbers (corresponding to the values of x and y at each point): 1 7, 2 10, 3 13, 4 16, and 5 19. Note that the first number in the pair (x) has integer values from 1 to 5 (inclusively), while the corresponding y -values are calculated from $y = 3x + 4$. More specifically, the y values in this example are given by: $1 \times 3 + 4$, $2 \times 3 + 4$, $3 \times 3 + 4$, $4 \times 3 + 4$, and $5 \times 3 + 4$.

Shown below is the programmer’s first attempt at writing this function.

```
def five_points_on_line(m b)
    for x in range(5)
        y = mx + b
        print x y
```

Unfortunately, although the intended purpose of each line can be deciphered, each of these lines is flawed in some way (there is at least one mistake per line and possibly more than one). Write the correct code corresponding to each line:

- 22. [4] Line 1: `def five_points_on_line(m, b):` # Add comma, colon.
- 23. [4] Line 2: `for x in range(1, 6):` # Add colon; change `range()` arguments.
- 24. [4] Line 3: `y = m * x + b` # Add multiplication operator.
- 25. [4] Line 4: `print(x, y)` # Add parentheses.

ALTERNATE SOLUTION:

- 22. [4] Line 1: `def five_points_on_line(m, b):` # Add comma, add colon.
- 23. [4] Line 2: `for x in range(5):` # Add colon.
- 24. [4] Line 3: `y = m * (x + 1) + b` # Add multiplication operator, adjust index.
- 25. [4] Line 4: `print((x + 1), y)` # Add parentheses, adjust index.

When the function is behaving properly, the following demonstrates the output it will produce:

```
>>> five_points_on_line(3, 4)    # Five points from line y = 3x + 4
1 7
2 10
3 13
4 16
5 19
```

```
>>> five_points_on_line(2, 2)    # Five points from line  $y = 2x + 2$ 
1 4
2 6
3 8
4 10
5 12
```

26. [12] Write a function called `display_line()` that is somewhat similar to the function described in the previous problem. This function has arguments for the slope (m) and y -intercept (b) of a line (where the line is again given by $y = mx + b$). The function prints the x - y values for points on this line when x takes on integer values in the range $[1, n]$ (inclusively). The value of n is provided to the function as the third parameter/argument. Notice that, n was fixed to 5 in the previous problem. Now this value is determined via the third parameter of the function.

Additionally, the function should *return*, as a two-element tuple, the range of values y takes on over the n points. More precisely, the two returned values in the tuple should be: the first calculated y value (which is $1 \times m + b$) and the last calculated y value (which is $n \times m + b$). Please see the sample output shown below.

There are more than enough lines provided for you to write your function.

```
def display_line(m, b, n):
    for x in range(1, n + 1):
        y = m * x + b
        print(x, y)
    return m + b, m * n + b
```

Sample output demonstrating correct behavior of the function:

```
>>> display_line(3, 4, 2)    # 2 points from  $y = 3x + 4$ .
1 7
2 10
(7, 10)
>>> a = display_line(2, 2, 8) # 8 points from  $y = 2x + 2$ . Assign return value to a.
1 4
2 6
3 8
4 10
5 12
6 14
7 16
8 18
>>> a                        # Show what a is.
(4, 18)
```