

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] What is the value of a after the following has been executed?

```
j = 2
k = 3
a = 2 * k ** j
```

- (a) 36
- (b) 18 ← **ANSWER**
- (c) 12
- (d) 32

2. [4] What is the value of sum after the following has been executed?

```
sum = 0
for k in range(1, 3):
    sum = sum + 1
```

- (a) 2 ← **ANSWER**
- (b) 3
- (c) 5
- (d) 0

3. [4] What is the value of sum after the following has been executed?

```
sum = 0
k = 2
for k in range(3):
    sum = sum + k
```

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

4. [4] What is the value of sum after the following has been executed?

```
sum = 0
k = 2
for i in range(3):
    sum = sum + k
```

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

5. [4] To what value does the following code set a?

```
a = list(range(-2, 2, 2))
```

- (a) [-2, 0, 2]
- (b) [-2, -1, 0, 1]
- (c) [-2, 0] ← **ANSWER**
- (d) [-1, 1]
- (e) This code produces an error.

6. [4] The following command is executed

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

In response to this the user types:

```
2 + 3
```

What would be the value of a?

- (a) '2 + 3'
- (b) 2
- (c) 5 **⇐ ANSWER**
- (d) This code produces an error.

7. [4] Consider the assignment statement:

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

What is the value of a[2]?

- (a) 3 **⇐ ANSWER**
- (b) 2
- (c) 4
- (d) [2, 4, 6, 8, 10]
- (e) This produces an error.

8. [4] To what value is the variable a set by the following code?

```
def sum_list(my_list):  
    sum = 0  
    for i in my_list:  
        sum = sum + i  
    return sum
```

```
a = sum_list([1, 4, 7])
```

- (a) 7
- (b) 0
- (c) 12 **⇐ ANSWER**
- (d) 5

9. [4] The following commands are executed

```
def g(x):  
    return x ** 2
```

```
a = g(4)
```

What is the resulting value of a?

- (a) '4 ** 2'
- (b) 16 **⇐ ANSWER**
- (c) 16.0
- (d) None

10. [4!] The following commands are executed

```
def h(x):  
    print(x * 3)  
  
a = h(4.0)
```

What is the resulting value of a?

- (a) 'x * 3'
- (b) '4.0 * 3'
- (c) 12.0
- (d) 12
- (e) None \leftarrow **ANSWER** Note that `print()` doesn't return anything.

11. [4] What is the output produced by the following?

```
for i in range(5):  
    print(i * i, end=" ")
```

- (a) 1 4 9 16
- (b) 0 1 4 9 16 \leftarrow **ANSWER**
- (c) 1 4 9 16 25
- (d) 0 1 4 9
- (e) This code produces an error

12. [4!] To what value is the variable a set by the following code?

```
def multiply_list(start, stop):  
    product = 1  
    for element in range(start, stop):  
        product = product * element  
    return product  
  
a = multiply_list(1, 4)
```

- (a) 24
- (b) 6
- (c) 2
- (d) 1 \leftarrow **ANSWER** Note return statement is in for-loop.

13. [4!] What is the output produced by the `print()` function in the following code?

```
def diff(a, b):  
    return a - b  
  
a = 5  
b = 3  
print(diff(b, a))
```

Output: -2 \leftarrow **ANSWER**

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

```
def f(n):  
    for i in range(-n, n + 1):  
        print(i, end=" ")
```

f(3)

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

15. [4] In the following code, what value is assigned to a?

```
def add_them(n1, n2):  
    return n1 + n2
```

a = add_them(1, 2)

- (a) 'n1 + n2'
- (b) '1 + 2'
- (c) 3 ⇐ **ANSWER**
- (d) This code produces an error.

16. [4] What is the value of a after the following statements have been executed?

```
x = 2  
y = 3  
y = x  
x = y  
a = y ** x
```

- (a) 27
- (b) 4 ⇐ **ANSWER**
- (c) 9
- (d) 8

17. [4] What is the value of a after the following statements have been executed?

```
x = 2  
y = 3  
x, y = y, x  
a = y ** x
```

- (a) 27
- (b) 4
- (c) 9
- (d) 8 ⇐ **ANSWER**

18. [5] For each of the following, circle whether the identifier is a valid or invalid variable name:

- (a) Valid or Invalid: too-good-to-be-true *Hyphens not allowed.*
- (b) Valid or Invalid: 2good2bettrue *Cannot start with digit.*
- (c) Valid or Invalid: tooGoodToBeTrue
- (d) Valid or Invalid: too.good.to.be.true *Periods not allowed.*
- (e) Valid or Invalid: _2_good_to_be_true_

19–23: [10 (2 each)] Assume the following code has been executed:

```
a = 4.0
b = a + 1 / 2
c = a * 1 / 2
d = (a + 1) // 2
e = int((a + 1) / 2)
f = 1 // 2 * a
```

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.

19. b = 4.5

20. c = 2.0

21. d = 2.0

22. e = 2

23. f = 0.0

24–26. A programmer is asked to write a function called `five_multiples()` that prints the first five multiples of the argument. For example, if the argument was 3, the printed values the function should generate are 3, 6, 9, 12, and 15 (representing 3×1 , 3×2 , 3×3 , 3×4 , and 3×5). Shown below is the programmer’s first attempt at writing this function.

```
def five_multiples(x)
    for i in range(5)
        print x ** i
```

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

- 24. [3] Line 1: `def five_multiples(x): # Add colon.`
- 25. [3] Line 2: `for i in range(1, 6): # Add colon; change range() arguments.`
- 26. [3] Line 3: `print(x * i) # Add parentheses; multiplication instead of exponentiation.`

ALTERNATE SOLUTION:

- 24. [3] Line 1: `def five_multiples(x): # Add colon.`
- 25. [3] Line 2: `for i in range(5): # Add colon.`
- 26. [3] Line 3: `print(x * (i + 1)) # Add parentheses; multiplication instead of exponentiation; adjust index.`

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

```
>>> five_multiples(3)    # First five multiples of 3.
3
6
9
12
15
>>> five_multiples(7)    # First five multiples of 7.
7
14
21
28
35
```

27. [8] Write a function called `multiples()` that is somewhat similar to the function described in the previous problem. This function takes two arguments (i.e., two parameters). The first argument is the number of multiples we wish to obtain (we only want positive multiples starting from one). The second argument is the value for which we want to generate multiples. We want the output for each multiple to show the multiplier, the second argument, and the result of the multiplication. Please see the sample output shown below. Additionally, the function should *return* the highest product displayed (which is simply the product of the two arguments).

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

```
def multiples(n, x):
    for i in range(1, n + 1):
        print(i, '*', x, '=', i * x)
    return i * x
```

Sample output demonstrating correct behavior of the function:

```
>>> multiples(4, 7)      # First four multiples of 7.
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
28
>>> a = multiples(3, 5) # First three multiples of 5. Assign return value to a.
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
>>> a                    # Show what a is.
15
```