

## CSCI 145 Exam 1 Practice — Lectures 1-4

Algorithms, variables & expressions, functions, scope & the call stack, modules & strings. 50 minutes, 50 points. You may use the Exam 1 reference sheet; no other notes, devices, or calculators. When asked for a value, write it the way Python would show it (for example, `3.0` vs. `3`, `'hi'` for a string).

---

### PART 1 — ALGORITHMS, TYPES & EXPRESSIONS

---

1. These steps compute the average of a list of test scores, but they are scrambled. Write the letters in the correct order. (3 pts)

- A. Divide total by count to get the average.
- B. Set total to 0 and count to 0.
- C. Report the average.
- D. Add each score to total, and add 1 to count.

Correct order: \_\_\_\_\_

2. Evaluate these Python expressions, writing both the value and its type in each blank (for example: `6.0`, `float`): (4 pts)

`7 * 3.0 - 4`      value: \_\_\_\_\_ type: \_\_\_\_\_

`-7 // 2`      value: \_\_\_\_\_ type: \_\_\_\_\_

`2 + 3 ** 2 * 2`      value: \_\_\_\_\_ type: \_\_\_\_\_

`9 / 3`      value: \_\_\_\_\_ type: \_\_\_\_\_

3. Evaluate each expression (value only). (3 pts)

`3 + 4 * 2 > 10 and not (5 == 5)`      \_\_\_\_\_

`5 == "5"`      \_\_\_\_\_

`"apple" < "Banana"`      \_\_\_\_\_

4. Evaluate each expression. If it causes an error, write Error. (3 pts)

`int(2.5)`      \_\_\_\_\_

`str(4) + "5"`      \_\_\_\_\_

`"Age: " + 18`      \_\_\_\_\_

## PART 2 — VARIABLES & FUNCTIONS

---

5. Consider each code snippet below and evaluate the asked-for variable. (2 pts)

```
x = 10
y = x
x = x + 5
```

The value of y is \_\_\_\_\_

```
name = "Sam"
name = name + "my"
greeting = name
name = "Al"
```

The value of greeting is \_\_\_\_\_

6. For each variable name, write Legal or Illegal. If it is legal, also write whether it follows this course's style guidelines (Good or Poor). (4 pts)

2nd\_place \_\_\_\_\_

class \_\_\_\_\_

l \_\_\_\_\_

is\_ready \_\_\_\_\_

7. Consider this code: (3 pts)

```
def double_print(n):
    print(n * 2)

def double_return(n):
    return n * 2

a = double_return(4) + 1
b = double_print(4)
```

a) What does this code print? \_\_\_\_\_

b) What is the value of a? \_\_\_\_\_

c) What is the value of b? \_\_\_\_\_

8. Write a function named `area_of_circle` that takes one parameter, `radius`, and returns the area of a circle with that radius. Use `math.pi` (assume `import math` has been done). Include a docstring that follows course conventions (what it does, its parameter, what it returns). (4 pts)

```
>>> area_of_circle(2)
12.566370614359172
```

9. Consider this code: (3 pts)

```
def modify(x):  
    bonus = 100  
    x = x + bonus  
    return x  
  
x = 5  
result = modify(x)
```

a) Value of x in the global scope after this runs: \_\_\_\_\_

b) Value of result: \_\_\_\_\_

c) What happens if we add `print(bonus)` as the last line of the program? Why?

10. Draw the complete call stack trace for this program: one frame for every function call, in the order the calls are made, from the start of the program until answer receives its final value. Label each frame with its call (for example, `f(3)`), fill in its parameter and variable values, and, once known, the value it returns. You may not need every box. (6 pts)

```
def half(x):  
    return x / 2  
  
def process(x):  
    y = half(x) + 1  
    return half(y)  
  
answer = process(20)
```

main

## PART 4 — MODULES & STRINGS

---

11. Evaluate each call (assume `import math` and `import random` have been done). For the `random` call, give the range of possible values. (4 pts)

`math.sqrt(81)` \_\_\_\_\_ `math.ceil(7.1)` \_\_\_\_\_

`math.log(100, 10)` \_\_\_\_\_ `random.randint(10, 20)` \_\_\_\_\_

12. Suppose `s = "BASKETBALL"`. Evaluate each expression. (3 pts)

`s[0:3] + s[-1]` \_\_\_\_\_ `s[6:] + s[:2]` \_\_\_\_\_

`s[1::2]` \_\_\_\_\_

13. Evaluate each expression. (3 pts)

`" Hi there ".strip().lower()` \_\_\_\_\_

`"Hello, World!".find("world")` \_\_\_\_\_

`"banana".replace("a", "o")` \_\_\_\_\_

14. Consider this code: (2 pts)

```
word = "cat"
word[0] = "b"
```

a) What happens when the second line runs?

b) Write one line of code that makes `word` refer to "bat" instead.

15. Write the exact output of each program. Show each newline as a line break, and show each tab as `→`. (3 pts)

```
print("Name:\tAda",
      "Language:\tPython",
      sep="\n")
```

```
print("a", "b", "c", sep="- ", end="!")
print("d")
```