Question 1: Multiple Choice Completely fill in the bubble next to your answer using a pencil. Each question should have exactly one filled-in bubble.

1.1. What is the *type* of the following expression?

```
1  1.5 + 2
```

- ○ `int`
- ● `float`
- ○ `str`
- ○ `bool`
- ○ `TypeError`

1.2. What is the *type* of the following expression?

```
1  len("cottage")
```

- ● `int`
- ○ `float`
- ○ `str`
- ○ `bool`
- ○ `TypeError`

1.3. What is the result of the following expression?

```
1  "110" + "110"
```

- ○ `220`
- ● `"110110"`
- ○ `TypeError`
- ○ `"220"`

1.4. What is the *result* of the following expression?

```
1  102 // 5
```

- ● `20`
- ○ `20.4`
- ○ `"20"`
- ○ `TypeError`
- ○ `21`

1.5. What is the *type* of this value in Python?

```
1  "True"
```

- ○ `bool`
- ● `str`
- ○ `TypeError`
- ○ `int`

1.6. What *value* will the following expression evaluate to?

```
1  "fox"[1]
```

- ○ `f`
- ○ `"f"`
- ○ `o`
- ● `"o"`
- ○ `TypeError`

1.7. What does the *len* function do in Python?

- ○ Converts a value to a string
- ○ Rounds a number to the nearest whole number
- ● Returns the length of a sequence
- ○ Converts a string to a number
- ○ Counts the digits in an int

1.8. What is a *bool* data type in Python?

- ○ Data type for storing text
- ○ Data type for storing numbers
- ● Data type for storing True/False values
- ○ Data type for storing any type of information
- ○ Data type for storing complex numbers

1.9. What is the indexing start position in Python sequences?

- ● `0`
- ◯ `1`
- ◯ `-1`
- ◯ `None`
- ◯ `TypeError`

1.10. Which of the following is a float in Python?

- ◯ `10`
- ● `10.0`
- ◯ `"10.0"`
- ◯ `True`

1.11. What does a docstring do in Python?

- ◯ `It performs calculations.`
- ◯ `It changes the value of a variable.`
- ● `It provides documentation for a function or module.`
- ◯ `It declares a new function.`
- ◯ `It calls a function.`

1.12. Is Python case-sensitive language?

- ● `Yes`
- ◯ `No`

1.13. What does the following Python expression evaluate to?

```
1 bool(0)
```

- ● `False`
- ◯ `True`
- ◯ `0`
- ◯ `1`

1.14. Which of the following is the correct way to concatenate two strings in Python?

- ◯ `"fox" , "hare"`
- ◯ `"fox" :  "hare"`
- ● `"fox" + "hare"`
- ◯ `"fox" "hare"`

1.15. What will the following Python expression evaluate to?

```
1 1 + True
```

- ◯ `True`
- ● `2`
- ◯ `1`
- ◯ `False`

1.16. What will the following Python expression evaluate to?

```
1 3.1415 * 2
```

- ● `6.283`
- ◯ `6`
- ◯ `5`
- ◯ `2`

1.17. Which of the following is a valid identifier name (e.g. function name) in Python?

- ◯ `123rabbit`
- ● `rabbit_123`
- ◯ `rabbit-123`
- ◯ `rabbit 123`

1.18. What is the result of evaluating the following Python expression?

```
1 2 ** 3
```

- ◯ `5`
- ● `8`
- ◯ `6`
- ◯ `4`

1.19. What is the result of the following operation?

```
1 110 + "110"
```

- ◯ 220
- ◯ "110110"
- ◯ "220"
- ● TypeError

1.20. What does this code evaluate to in Python?

```
1 int(5.75)
```

- ◯ 5.5
- ● 5
- ◯ 6
- ◯ TypeError

1.21. Suppose we have a `float` named `x`, use a constructor function call expressions to convert it into an `int`. Which of the following is correct?

- ◯ `x("int")`
- ● `int(x)`
- ◯ `(int)x`
- ◯ `float_to_int(x)`

1.22. Suppose we have the following literal expression `"3.14"`. What is the type of this expression?

- ◯ `int`
- ◯ `float`
- ● `str`
- ◯ `bool`

1.23. Which of the following is a literal expression for a string in Python?

- ◯ `string("Hello")`
- ◯ `"Hello"{}`
- ● `"Hello"`
- ◯ `print("Hello")`

1.24. Which are valid `bool` literals in Python?

- ● `True` / `False`
- ◯ Yes / No
- ◯ 1 / 0
- ◯ On / Off

1.25. What function would you use to get the data type of an object?

- ◯ data_type()
- ◯ get_type()
- ◯ typeof()
- ● type()

Question 2: Multiple Choice Completely fill in the bubble next to your answer using a pencil. Each question should have exactly one filled-in bubble.

2.1. A function call expression's evaluated value is determined by _____.

- ● the first return statement evaluated in the function definition
- ○ the last return statement evaluated in the function definition
- ○ each and every return statement evaluated in the function definition

2.2. Below is a properly defined Python function. What is the the role of the "beverage" parameter?

```
1 def order_beverage(beverage: str) -> str:
2   """This function orders a beverage"""
3   return "Your " + beverage + " is ready!"
```

- ○ The return value
- ● An input to the function
- ○ The function's name
- ○ The external variable

2.3. What will be the result of the following Python function?

```
1 def evaluate_length(name: str) -> int:
2   """This function returns the length of the name"""
3   return len(name)
```

`evaluate_length("Foxglove")`

- ○ `7`
- ● `8`
- ○ `"8"`
- ○ `"Foxglove"`

2.4. Consider the function declared below. What value is returned when `fairytale_winter(coziness=3, days=5)` is called?

```
1 def fairytale_winter(coziness: int, days: int) -> float:
2   """This function estimates the enjoyment during winter days."""
3   return coziness * days / 2.0
```

- ○ `15.0`
- ● `7.5`
- ○ `7`
- ○ `"7.5"`

2.5. What will be the *printed output* of the following Python function call?

```
1 def say_hello(name: str) -> None:
2   """This function prints a greeting"""
3   print("Hello, " + name + "!")
```

`say_hello("Doe")`

- ● Hello, Doe!
- ○ `"Hello, Doe!"`
- ○ Nothing
- ○ TypeError

Question 3: Evaluate and Respond to the following questions.

3.1. What is the return type of the following function?

```
1 def acorn_count(tree_count: int, acorns_per_tree: int) -> int:
2   """Returns the total number of acorns in the woodland."""
3   return tree_count * acorns_per_tree
```

- ● `int`
- ○ `str`
- ○ `float`
- ○ `bool`

3.2. Complete the following code to call `acorn_count` function such that 110 is printed to the screen.

```
1 print(acorn_count(____________________))
```

Solution: tree_count=11, acorns_per_tree=10

3.3. What value and type does the following expression evaluate to: `int("1" + "2")`

Solution: 12, int

3.4. What value and type does the following expression evaluate to: `3 + 4 * 5`

Solution: 23, int

3.5. What value and type does the following expression evaluate to?

```
1 len(str(10 // 3))
```

Solution: 1, int

3.6. What value and type does the following expression evaluate to?

```
1 str(10 % 3)
```

Solution: "1", str

3.7. Fill in the blank. Given the below definition, what value does the following function call evaluate to: `sum_length(recipe_str="PumpkinPie", ingredient_str="SugarBeet")`

```
1 def sum_length(recipe_str:str, ingredient_str:str) -> int:
2   """Returns the sum of the length of a recipe and an ingredient"""
3   return len(recipe_str) + len(ingredient_str)
```

Solution: 19

Question 4: Identification Given the following code listing, identify lines which contain the following concepts.

```
1  def total_feet(sparrows: int, rabbits: int) -> int:
2    """Returns the total number of feet among the woodland creatures"""
3    return bird_feet(birds=sparrows) + rabbit_feet(rabbits=rabbits)
4
5
6  def bird_feet(birds: int) -> int:
7    """Returns the total number of bird feet given a number of birds"""
8    return 2 * birds
9
10
11 def rabbit_feet(rabbits: int) -> int:
12   """Returns the total number of rabbit hindfeet and forefeet."""
13   return 4 * rabbits
14
15
16 print(total_feet(sparrows=3, rabbits=2))
```

4.1. Identify the line number where a function definition signature is found.

- ◯ `Line 2`
- ◯ `Line 3`
- ● `Line 6`
- ◯ `Line 9`
- ◯ `Line 10`

4.2. Identify the line number where a docstring is written.

- ◯ `Line 1`
- ● `Line 2`
- ◯ `Line 4`
- ◯ `Line 5`
- ◯ `Line 6`

4.3. Identify the line number where an expression is found.

- ◯ `Line 1`
- ◯ `Line 2`
- ◯ `Line 5`
- ● `Line 8`
- ◯ `Line 10`

4.4. What is `-> int` an example of?

- ◯ `parameter type`
- ● `return type`
- ◯ `expression`
- ◯ `type conversion`

4.5. Identify the line number where a function call is made.

- ◯ `Line 1`
- ◯ `Line 2`
- ● `Line 3`
- ◯ `Line 4`
- ◯ `Line 5`

4.6. Which of the following is a parameter name?

- ◯ `bird_feet`
- ◯ `print`
- ● `birds`
- ◯ `bunnies`

4.7. What would be the printed result of the code listing?

- ◯ `5`
- ◯ `10`
- ◯ `12`
- ● `14`
- ◯ `20`

4.8. Which function definition is jumped into *second*?

- ◯ `print`
- ◯ `total_feet`
- ● `bird_feet`
- ◯ `rabbit_feet`

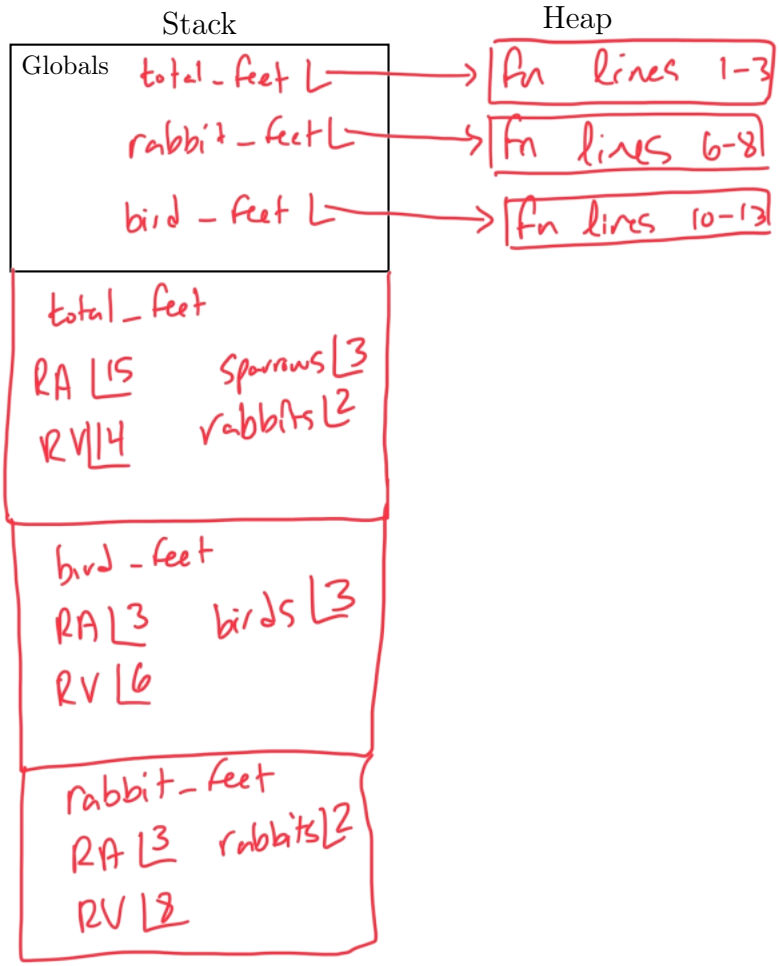
Question 5: Memory Diagram Trace a memory diagram of the following code listing and then answer the sub-questions. You do not need to diagram the sub-questions.

```
1  def total_feet(sparrows: int, rabbits: int) -> int:
2    """Returns the total number of feet among the woodland creatures"""
3    return bird_feet(birds=sparrows) + rabbit_feet(rabbits=rabbits)
4                    6                +              8
5                                     14
6  def rabbit_feet(rabbits: int) -> int:
7    """Returns the total number of rabbit hindfeet and forefeet."""
8    return 4 * rabbits
9
10
11 def bird_feet(birds: int) -> int:
12   """Returns the total number of bird feet given a number of birds"""
13   return 2 * birds
14
15 print(total_feet(sparrows=3, rabbits=2))
```

we will not look at scratch work!

Output

```
Solution: 14
14
```



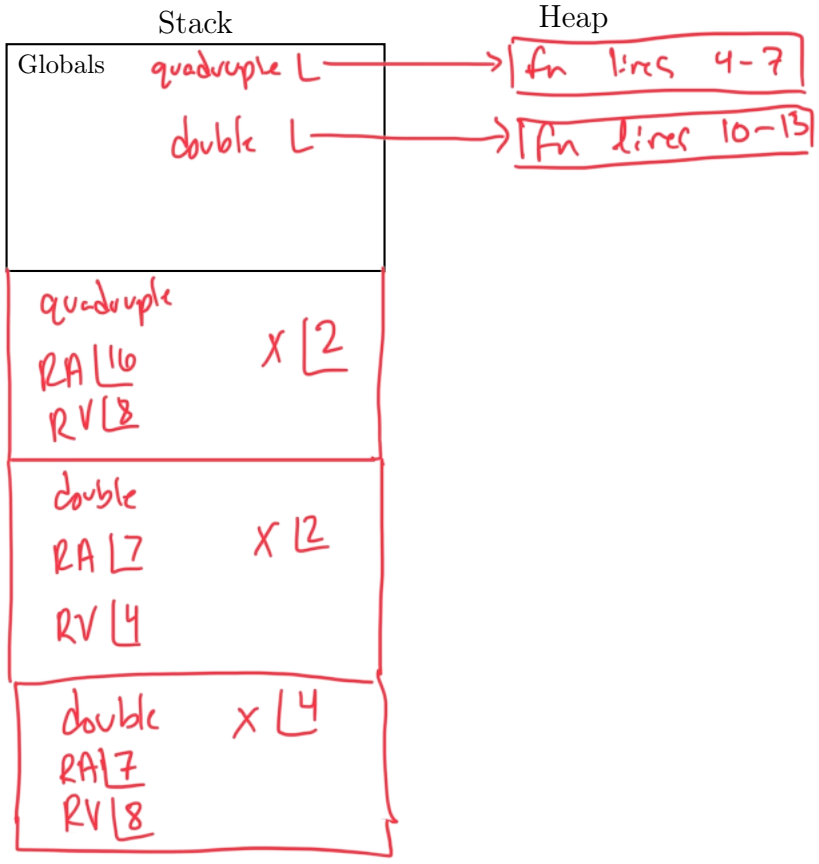
Question 6: Memory Diagram Trace a memory diagram of the following code listing and then answer the sub-questions. You do not need to diagram the sub-questions.

```
1  """Some fun functions..."""
2
3
4  def quadruple(x: int) -> int:
5    """Quadruple an int!"""
6    print("quadruple(" + str(x) + ")")
7    return double(x=double(x=x))
8
9
10 def double(x: int) -> int:
11   """Double an int!"""
12   print("double(" + str(x) + ")")
13   return 2 * x
14
15
16 print(quadruple(x=2))
```

Output

Solution:

quadruple(2)
double(2)
double(4)
8



Question 7: Memory Diagram Trace a memory diagram of the following code listing and then answer the sub-questions. You do not need to diagram the sub-questions.

```
1  """Functions of a circle..."""
2
3
4  def main() -> None:
5    """Entrypoint of Program"""
6    print(perimeter(radius=1.0))
7    print(area(radius=1.0))
8    return None
9
10
11 def area(radius: float) -> float:
12   """Calculate area of a circle"""
13   return 3.14 * radius**2
14
15
16 def perimeter(radius: float) -> float:
17   return 2 * 3.14 * radius
18
19
20 main()
```

Output

Solution:

6.28

3.14