More About Python quiz

TOTAL POINTS 12

1. Please decide whether the following statements are true or false

1 point

An “if” statement must be indented, by 4 spaces.

- (●) F
- ( ) T

2. If k is integer, how many times will the following “while” loop be executed?

1 point

```
1  k = 50
2  while k > 1:
3      print(k)
4      k = k // 2
```

- ( ) 6
- ( ) 3
- ( ) 4
- (●) 5

3. Which of the following code snippets can print “rest apples are less than 9”once and only once?

1 point

- ( )
```
1  apples = 100
2  while True:
3      if apples < 9:
4          continue
5          print("rest apples are less than 9")
6      apples -= 9
```

- (●)
```
1  apples = 100
2  while apples >= 1:
3      if apples < 9:
4          print("rest apples are less than 9")
5          break
6      apples -= 9
```

```
apples = 100
for a in reversed(range(apples)):
    if a < 9:
        print("rest apples are less than 9")
        continue
        apples -= 9
```

```
apples = 100
while True:
    if apples < 9:
        break
        print("rest apples are less than 9")
    apples -= 9
```

4. Which of the following statements about the flow control of Python functions are correct? 1 point

- [ ] In “while” and “for” loops, a “continue” statement serves to stop the current loop and continues to enter the statement(s) below the loop body.
- [x] One of the characteristics of an “if” statement is: it makes judgment from top to bottom, and if one judgment is True, the statement corresponding to that judgment will be executed, ignoring the remaining “elif” and “else”.
- [x] Boolean operators have a very interesting short-circuit logic behavior: for an expression “x and y”, when “x” is false, it directly returns “False”, without calculating the value of “y”.
- [ ] In “while” and “for” loops, a “break” statement serves to end the current loop and re-starts the loop.

5. Which of the following statements about the flow control of Python functions are correct? 1 point

- [x] As Boolean expressions, the values of None, 0, [] and {} would be regarded as “False” by the interpreter.
- [ ] When “is” is a comparison operator, the meaning of “x is y” is to compare whether “x” is a sub-class of “y”.

- [x] Standard Boolean values are 0 (representing False) and 1 (representing True); in fact, the result of the statement True==1 is True.

- [ ] Comparisons of incompatible types, like integers and strings, is meaningless in mathematics. It is no longer supported in Python 3.x.

6. What's the result of the following program? 1 point

```
s = 0
for i in range(1, 11):
    if i % 2 == 0:
        continue
    if i % 10 == 5:
        break
    s = s + i
print(s)
```

4

7. Regarding the function below: 1 point

```
def location(city, province):
    print('%s belongs to %s province' % (city, province))
```

Which of the following statement has a different result comparing with others?

- ○ location(province = 'Jiangsu', city = 'Nanjing')
- ○ location('Jiangsu', 'Nanjing')
- ○ location(city = 'Nanjing', province = 'Jiangsu')
- ◉ location('Nanjing', 'Jiangsu')

8. Define a function as below with f as the function parameter, 1 point

```
def test(f, a, b):
    print(f(a, b))
```

Which of the following options will be the result of test((lambda x,y: x ** 3 + y), 2, 3)?

- 10
- 11
- 8
- 9

9. Define a function as below: 1 point

```
def my_power(x, n = 2):
    s = 1
    while n > 0:
        n -= 1
        s = s * x
    return s
```

What's the result of passing my_power(-3) and my_power(3, 3) respectively?

- 9 and -27
- -9 and -27
- 9 and 27
- -9 and 27

10. Which of the following is correct about the program below? 1 point

```
def f(x):
     a = 7
     print(a + x)
a = 5
f(3)
print(a)
```

- [x] The result of the program is 10 and 5.
- [ ] The result of the program is 10 and 7.
- [ ] The program cannot be executed normally.
- [ ] The result of the program is 8 and 5.

11. What kind of exception will be generated when executing the following code snippet? 1 point

```
1  >>> a = 3
2  >>> print(a ** b)
```

- [ ] IndexError
- [ ] TypeError
- [ ] ValueError
- [x] NameError

12. If the code snippet always generates a random number in [0, 1.0), what's the possible function from library random used here? 1 point

```
1  >>> import random
2  >>> random._____()
```

- [ ] random
- [x] randint
- [ ] shuffle
- [ ] uniform