I. Strings

Given the following string, answer the questions below.

my_string = “a0:12:90:00:80:43”

1. What character is returned by **my_string[0]** ?

2. What character is returned by **my_string[10]** ?

3. What character is returned by **my_string[-1]** ?

4. What character is returned by **my_string[19]** ?

5. What string is returned by **my_string[-3] * 3** ?

6. What string is returned by **my_string[4:7]** ?

7. What string is returned by **my_string[-4:]** ?

8. What string is returned by **my_string[:-4]** ?

9. What string is returned by **my_string[::3]** ?

10. How can you extract the “**90:00**” substring using a slice and positive indexes?

11. How can you extract the “**90:00**” substring using a slice and negative indexes?

12. How can you get the number of colons (“:”) in **my_string** using a string method?

13. How can you remove the colons in **my_string** using a string method?

14. How can you get a list where each element is an octet of the MAC address in **my_string** using a string method?

15. How can you concatenate the elements in the list obtained at question 14 using a dot (“.”) as a separator (**a0.12.90.00.80.43**) using a string method?

II. Numbers

1. How would you raise 5 to the power of 3 in the Python interpreter? **Hint:** You can do it in 2 ways.

2. What would be the result of the following operation in the Python interpreter? **30 % 8**

3. What will the Python interpreter return when entering the following expression? **25 != 52**

4. What will the Python 2.x interpreter return when entering the following expression? **50 / 15**

5. What will the Python 2.x interpreter return when entering the following expression? **50 / 15.0**

6. What will the Python 2.x interpreter return when entering the following expression? **abs(-11)**

7. What will the Python 2.x interpreter return when entering the following expression? **max(5,’y’)**

III. Booleans

1. What would be the result of: **"nortel" == "nOrtel"**?

2. What would be the result of: **(10 == 10) and (20 == 30)**?

3. What would be the result of: **(211 == 210) or (7 == 7)**?

4. What would be the result of: **bool(0.0)**?

5. What would be the result of: **bool('y')**?

6. What would be the result of: **bool(0j) or bool(2015)**?

IV. Lists

Given the following list, answer the questions below.

my_list = [10, 'x', 20.02, 'y', 30j, 'z', 10L, False]

1. What would be the result of: **my_list[-1]**?

2. What would be the result of: **my_list[0]**?

3. What would be the result of: **my_list[:]**?

4. What would be the result of: **my_list[5]**?

5. What would be the result of: **my_list[3:6]**?

6. What would be the result of: **my_list[-4:-2]**?

7. What would be the result of: **my_list[::3]**?

8. What would be the result of: **my_list[:-5] * 5**?

9. What would be the result of: **type(my_list[6])**?

10. What would be the result of: **type(my_list[7])**?

11. How would you add the following element to **my_list**? **'new element'**

12. How would you delete element **'x'** from **my_list**? Do it in 3 ways.

13. How can you find the index of element **20.02** in **my_list**?

14. Remove the **30j** element and sort the **my_list** list in ascending order using 2 methods.

15. Remove the **30j** element and sort the **my_list** list in descending order using 2 methods.

V. Sets

Given the following sets, answer the questions below.

set1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

set2 = {2, 4, 6, 9}

1. How would you add the following element to **set1**? 500

2. How would you remove the following element from **set1**? 7

3. How would you remove a random element from **set1**?

4. How would you get the common elements of **set1** and **set2**?

5. How would you get the elements in **set1** which are not in **set2**?

6. How would you unify **set1** and **set2**?

7. How would you clear **set2**?

VI. Tuples

Given the following tuple, answer the questions below.

my_tuple = (1, 2, 3, 'a', 'b', 'c', [4, 5, 6])

1. What would be the result of: **my_tuple[-3]**?

2. What would be the result of: **my_tuple[0]**?

3. What would be the result of: **my_tuple[:2]**?

4. What would be the result of: **my_tuple[3]**?

5. What would be the result of: **my_tuple[3:5]**?

6. What would be the result of: **my_tuple[-5:-3]**?

7. What would be the result of: **my_tuple[::2]**?

8. What would be the result of: **my_tuple[-1] * 5**?

9. If **my_tup1 = (1, 2, 3)** and **(a, b, c) = my_tup1** then who is **b**?

10. If **(x, y, z) = (15, 25, 35)** then what is the result of **y % x**?

VII. Dictionaries

Given the following dictionary, answer the questions below.

my_dict = {1: "Cisco", 2: "HP", 3: "Juniper", 4: "Arista", 5: "Avaya"}

1. How would you add a 6th element to **my_dict**, having the value of **"Nortel"**?

2. How would you delete the previously added element, by specifying its value?

3. How can you check if the **4** key exists in my_dict?

4. What would be the result of **len(my_dict) == 4** in the Python interpreter?

5. What would be the result of **max(my_dict.keys())** ?

6. What would be the result of **sorted(my_dict.values())[2]** ?

7. What would be the result of **my_dict.items()[-1][1]** ?

VIII. If/For/While/Nesting

Given the following code, answer the questions below.

```
if x < 10:
    for i in range(1, 5):
        print x * i
elif x > 10:
    j = 1
    while j < 5:
        print x * j
        j = j + 1
else:
    print x ** 10
```

1. What will this code block return if **x** is equal to 2?

2. What will this code block return if **x** is equal to 11?

3. What will this code block return if **x** is equal to 10?

IX. Regular Expressions

Given the following string (an Avaya 3510 routing table entry), answer the questions below.

```
my_regex_str = '200.10.2.0      255.255.255.0   200.20.5.2      1     205  T#1     S  IB     5'
```

1. What will this code block return in the Python interpreter?

```
import re
a = re.match(r"255", my_regex_str)
type(a)
```

2. What will this code block return in the Python interpreter?

Hint 1: For a better view on a single line copy the code in Notepad.

Hint 2: Be careful at the number of spaces in the string. Count them using your mouse.

Hint 3: The definition of “a” is the same for questions 2-13. Only the argument of group() differs.

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(0)
```

3. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(1)
```

4. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(2)
```

5. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(3)
```

6. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(4)
```

7. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(5)
```

8. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(6)
```

9. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(7)
```

10. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(8)
```

11. What will this code block return in the Python interpreter?

```
import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3})
+(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(9)
```

12. What will this code block return in the Python interpreter?

import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3}) +(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(10)

13. What will this code block return in the Python interpreter?

import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3}) +(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(11)

14. What will this code block return in the Python interpreter?

import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3}) +(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group()

15. What will this code block return in the Python interpreter?

import re

a = re.search(r"(.+?) +\d\d(\d)\.([0-9]{2,})\.([0-9]{1,3})\.(\d) +(.+)1 +(\d{3}) +(\w{1})#.+S(\s+)(\w)\w +(.*)", my_regex_str)

a.group(12)

16. What will this code block return in the Python interpreter?

import re

my_regex_str = "Ethernet Routing Switch 3549GTS-PWR+"

a = re.findall(r"(.{5}).+ (.+?)\s(\d{2,4}).+-(.{4})", my_regex_str)

a[0][0]

17. What will this code block return in the Python interpreter?

```
import re

my_regex_str = "Ethernet Routing Switch 3549GTS-PWR+"

a = re.findall(r"(.{5}).+ (.+?)\s(\d{2,4}).+-(.{4})", my_regex_str)

a[0][1]
```

18. What will this code block return in the Python interpreter?

```
import re

my_regex_str = "Ethernet Routing Switch 3549GTS-PWR+"

a = re.findall(r"(.{5}).+ (.+?)\s(\d{2,4}).+-(.{4})", my_regex_str)

a[0][2]
```

19. What will this code block return in the Python interpreter?

```
import re

my_regex_str = "Ethernet Routing Switch 3549GTS-PWR+"

a = re.findall(r"(.{5}).+ (.+?)\s(\d{2,4}).+-(.{4})", my_regex_str)

a[0][3]
```

20. What will this code block return in the Python interpreter?

```
import re

my_regex_str = "Ethernet Routing Switch 3549GTS-PWR+"

a = re.sub(r"[0-9]", "5xy", my_regex_str)

a
```

X. Advanced Python Tools

1. Write a list comprehension that takes every integer from **[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]** and multiplies it with 10.

The result should be: **[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]**

2. Write a list comprehension that interates over **[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]** and multiplies with 10 only the elements that are less than or equal to 5.

The result should be: **[10, 20, 30, 40, 50]**

3. Write a list comprehension that interates over both **[1, 2, 3, 4, 5]** and **[10, 11, 12]** and multiplies each element of the first list with each element of the second list.

The result should be: **[10, 11, 12, 20, 22, 24, 30, 33, 36, 40, 44, 48, 50, 55, 60]**

4. Write a list comprehension that interates over both **[1, 2, 3, 4, 5]** and **[10, 11, 12]** and multiplies each element of the first list with each element of the second list that is less than or equal to 11.

The result should be: **[10, 11, 20, 22, 30, 33, 40, 44, 50, 55]**

5. Write a lambda function that takes two parameters and multiplies them.

The result should be similar to: **my_lam(10, 5) -> 50**

6. Write a lambda function that takes three strings as parameters and concatenates them, inserting a space character between each word.

The result should be similar to: **my_lam("Python", "Network", "Programming") -> 'Python Network Programming'**

7. Write a lambda function that takes a dictionary as a parameter and returns the keys of that dictionary in a list, sorted in reverse order (descending).

The result should be similar to: **my_lam({1:'a', 2:'b', 3:'c'}) -> [3, 2, 1]**

8. Having the **lambda x: x / 100** function, use **map()** to apply it to the list generated by **range(0, 1000, 100)**.

The result should be: **[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**

9. Having the **lambda x: x % 100 == 1** function, use **filter()** to apply it to the list generated by **range(0, 1000, 100)**.

The result should be: **[]** (because the remainder of dividing each element in the list by 100 is 0)

10. Calculate the result of multiplying all the integers starting from 1 up to and including 10 using the **reduce()** function.

The result should be: **3628800**