EECS 1210 Introduction to Programming in Python
Spring Semester 2024
Prof. Pai H. Chou

Self-Check 5

Answer the following questions to check your understanding of your material. Expect the same kind of questions to show up on your tests.

1. Definitions and Short Answers

1. What is the data type of `(1, 2, 3)`? Tuple
2. If `s = 'ABCDE'`, what is the value of
   - `s[0]` `'A'`
   - `s[1]` `'B'`
   - `s[-1]` `'E'`
   - `s[1:4]` `'BCD'`
   - `s[-5:-2]` `'ABC'`
   - `s[:2]` `'AB'`
   - `s[-3:]` `'CDE'`
   - `s[:]` `'ABCDE'`
   - `s[0:0]` `''`
   - `s[1:4:2]` `'BD'`
   - `s[-1:0:-1]` `'EDCB'`
3. If `S = ['h', 'e', 'l', 'l', 'o']`, what is the value of `S` after executing the statement `S[1:2] = ['a']`? `[ 'h', 'a', 'l', 'l' , 'o' ]`
4. If `T = ('h', 'e', 'l', 'l', 'o')`, which of the following is allowed or not allowed and why?
   - `T[3] = 'z'` `tuple' object does not support item assignment`
   - `T = ('w', 'o', 'r', 'l', 'd')` `allowed`
   - `T = T[2:-1]` `allowed`
5. What is the value of
   - `list('apple')` `['a', 'p', 'p', 'l', 'e']`
   - `tuple('apple')` `('a', 'p', 'p', 'l', 'e')`
   - `set('apple')` `{'a', 'p', 'e', 'l'}`
6. What is the value of
   - `str(['a', 'p', 'p', 'l', 'e'])` `['a', 'p', 'p', 'l', 'e']`
   - `str(('a', 'p', 'p', 'l', 'e'))` `('a', 'p', 'p', 'l', 'e')`
   - `str({'a', 'p', 'p', 'l', 'e'})` `{'a', 'p', 'e', 'l'}`
7. What is the value of
   - `list(('a', 'p', 'p', 'l', 'e'))` `['a', 'p', 'p', 'l', 'e']`
   - `tuple(['a', 'p', 'p', 'l', 'e'])``('a', 'p', 'p', 'l', 'e')`

- `set(['a', 'p', 'p', 'l', 'e']) {'a', 'p', 'e', 'l'}`

8. What is the result of
   - `'Apple' < 'apple' True`
   - `'Apple' <= 'apple' True`
   - `'Apple' == 'apple' False`
   - `'Apple' >= 'apple' False`
   - `'Apple' > 'apple' False`
   - `'Apple' != 'apple' True`
9. What is the result of
   - `'Apple' < 'adventure' True`
   - `'apple' < 'adventure' False`
   - `'apple' < 'Adventure' False`
   - `'apple' < 'bee' True`
   - `'apple' < 'Bee' False`
   - `'Apple' < 'bee' True`
   - `'Apple' < 'Bee' True`
10. What is the result of
    - `('apple', 0) < ('apple', 2) True`
    - `('apple', 0, 3) < ('apple', 1) True`
    - `['apple', 2, 2] < ['apple', 2, 1, 5] False`
    - `['apple', 3] < ['oranges', 0] True`
11. What is the result of
    - `'s' in 'school' True`
    - `'hoo' in 'school' True`
    - `'S' in 'school' False`
    - `'ol' in 'school' True`
    - `'k' not in 'school' True`
    - `'s' not in 'School' True`
12. What is the result of
    - `'s' in ['s', 'c', 'h', 'o', 'o', 'l'] True`
    - `['s'] in ['s', 'c', 'h', 'o', 'o', 'l'] False`
    - `['s'] in [['s'], ['c'], ['h'], ['o'], ['o'], ['l']] True`
    - `'hoo' in ['s', 'c', 'h', 'o', 'o', 'l'] False`
    - `['h', 'o', 'o'] in ['s', 'c', 'h', 'o', 'o', 'l'] False`
    - `('h', 'o', 'o') in ['s', 'c', ('h', 'o', 'o'), 'l'] True`
    - `('h', 'o', 'o') not in ('s', 'c', ('h', 'o', 'o'), 'l')` `False`
    - `'ol' in ['s', 'c', 'h', 'o', 'ol'] True`
    - `'s' in ['S', 'c', 'h', 'o', 'o', 'l'] False`
13. What is the result of
    - `'sch' + 'ool' 'school'`
    - `[1, 2, 3] + [4, 5, 6] [1, 2, 3, 4, 5, 6]`

- `(1, 2, 3) + (4, 5, 6)` `(1, 2, 3, 4, 5, 6)`

14. What is the result of
    - `'sch' + 'o' * 10 + 'l'` `'schoooooooooool'`
    - `'do' * 5` `'dododododo'`
    - `['s'] + ['o'] * 5 + ['l']` `['s', 'o', 'o', 'o', 'o', 'o', 'l']`
15. How do you express a tuple literal of a single element? For example, how do you write a tuple literal that has the same value as `tuple([1])`?(1,)
16. Suppose you have `x = 1, 2, 3`
    What is the value of `type(x)`? tuple
17. Suppose you have `L = ['f', 'r', 'o', 'g']`
    What is the new value of L after executing each of the following statements in order?
    - `L.append('s')` `['f', 'r', 'o', 'g', 's']`
    - `L.extend(['p', 'o', 'n', 'd'])` `['f', 'r', 'o', 'g', 's', 'p', 'o', 'n', 'd']`
    - `L.insert(4, ' ')` `['f', 'r', 'o', 'g', '', 's', 'p', 'o', 'n', 'd']`
    - `L.reverse()` `['d', 'n', 'o', 'p', 's', '', 'g', 'o', 'r', 'f']`
    - `L.sort()` `['', 'd', 'f', 'g', 'n', 'o', 'o', 'p', 'r', 's']`
    - `L.remove('o')` `['', 'd', 'f', 'g', 'n', 'o', 'p', 'r', 's']`
    - `L.pop()` `['', 'd', 'f', 'g', 'n', 'o', 'p', 'r']`
    - `L.pop(0)` `['d', 'f', 'g', 'n', 'o', 'p', 'r']`
    - `L.clear()` `[]`
    - `L.append('z')` `['z']`
18. If `T = (1, 3, 5, 7, 9, 11)`, Can you call `del(T[1])`? why or why not? Can you call `del(T)`? What is the effect?
    'tuple' object doesn't support item deletion
    del(T) deletes tuple T
19. Suppose `L = list('hello')` and separately `M = list('hello')`. After executing
    `L.reverse()`
    `M = M[::-1]`
    - is `L == M` evaluate to `True` or `False`? True
    - What is the difference between these two ways of reversing elements in a list?
      L 在原資料結構上直接被修改了！M 是先創一個新的資料結構再從舊的資料結構指到新的資料結構 若沒有人指向舊的資料結構, 則系統會定期回收該資料結構
20. if `T = tuple('hello')`, are the following statements allowed in Python? Why or why not?

- `T.reverse()` `'tuple' object has no attribute 'reverse'`
- `T = T[::-1]`
  `It's allowed because tuple is a sequencial data structure`

21. What is a **stack** as a data structure? What is another name (4-letter initialism) for a stack? How can a stack be implemented using a `list`? Show how **push** and **pop** can be accomplished by calling list methods.
    像一個只有一端開口的罐子,又稱 LIFO.用 List 實作 stack
    Push 可使用 L.append('item') pop 可使用 L.pop()
22. What is a **queue** as a data structure? What is another name (4-letter initialism) for a queue? How can a queue be implemented using a list? Show how enqueue and dequeue can be accomplished by calling list methods.
    像一個有兩端開口的管子,又稱 FIFO. 用 List 實作 queue
    Enqueue 可使用 L.append('item') Dequeue 可使用 L.pop(0)
23. Show how a **tuple** can be used to implement
    - a stack's push and pop functionality
      push : T = T+tuple('item')
      pop : T = T[:-1]
    - a queue's enqueue and dequeue functionality
      enqueue : T = tuple('item') + T
      dequeue : T = T[:-1]
    - Is a tuple more or less efficient than a list for implementing the stack and queue data structures? Why?
      tuple 很沒效率且不直觀,建議用 list
24. What do these built-in functions do?
    - `max(['h', 'e', 'l', 'l', 'o'])` `'o'`
    - `min('hello')` `'e'`
    - `sum([2, 3, 4, 5, 6])` `20`
    - `sum(range(10))` `45`
    - `any(['', 'apples', 'oranges', 'banana'])` `True`
    - `any([0, '', 0.0, [], ()])` `False`
    - `any(['0', '', 0.0, [], ()])` `True`
    - `any([0, ' ', 0.0, [], ()])` `True`
    - `all(['', 'apples', 'oranges', 'banana'])` `False`
    - `all([' ', 'apples', 'oranges', 'bananas'])` `True`
    - `all([0, '', 0.0, [], ()])` `False`
25. What is the **non-mutation** version of the following statements? Assume `L` is a list
    - `L.sort()` `S = sorted(S)`
    - `L.reverse()` `S = list(reversed(S))` or `S =[::-1]`
    - `L.extend([1, 2, 3])` `S = S + [1,2,3]`
    - `del(L[1])` `S = S[:1] + S[2:]`
    - `L.pop()` `S = S[:-1]`
26. How do you use **list comprehension** to create a list with values
    - `['*', '**', '***', '****', '*****']`
      `['*' * (i + 1) for i in range(5)]`

- `[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]`
  `[2**i for i in range(0,13)]`

27. How do you use **two-level list comprehension** to create a multiplication table in the following format: [(1, 1, 1), (1, 2, 2), (1, 3, 3), … (1, 9, 9), (2, 1, 2), (2, 2, 4), (2, 3, 6), (2, 4, 8), … (2, 9, 18), (3, 1, 3), (3, 2, 6), (3, 3, 9), … (3, 9, 27), (4, 1, 4) … (4, 9, 36), (5, 1, 5), … (9, 9, 81)]
    [(i,j,i*j) for i in range(1,10) for j in range(1,10)]

28. How do you use list comprehension with filter to generate the list of upper-case letters except 'A', 'E', 'I', 'O', 'U'?
    [chr(i) for i in range(65,91) if chr(i) not in ['A', 'E', 'I', 'O', 'U']]
29. After executing the following sequence of statements:
    ```
    x = 3
    y = x
    x = 4
    ```
    what is the value of y? 3
30. After executing the following sequence of statements
    ```
    x = [1, 2, 3]
    y = x
    x = [4, 5, 6]
    ```
    what is the value of y? [1, 2, 3]
31. After executing the following sequence of statements
    ```
    x = [1, 2, 3]
    y = x
    x[1] = 4
    ```
    what is the value of y? [1, 4, 3]
32. After executing the following sequence of statements
    ```
    x = [1, 2, 3]
    y = x[:]
    x[1] = 4
    ```
    what is the value of y? [1, 2, 3]
33. After executing the following sequence of statements
    ```
    x = [1, 2, 3]
    y = x
    y[:] = [4, 5, 6]
    ```
    what is the value of x? [4, 5, 6]
34. After executing the following sequence of statements
    ```
    z = ['a', 'b']
    x = [1, z, 3]
    z.append('c')
    ```
    what is the value of x? [1, ['a', 'b', 'c'], 3]
35. After executing the following sequence of statements
    ```
    z = ['a', 'b']
    ```

```
x = [1, z, 3]
y = x
z.append('c')
```

what is the value of `y`? [1, ['a', 'b', 'c'], 3]

36. After executing the following sequence of statements

```
z = ['a', 'b']
x = [1, z, 3]
y = x[:]
z.append('c')
```

what is the value of `y`? [1, ['a', 'b', 'c'], 3]

37. After executing the following sequence of statements

```
z = ['a', 'b']
x = [1, z, 3]
y = x[:]
x[0] = 4
z.append('c')
```

what is the value of `y`? [1, ['a', 'b', 'c'], 3]

38. After executing the following sequence of statements

```
import copy
z = ['a', 'b']
x = [1, z, 3]
y = copy.copy(x)
x[0] = 4
z.append('c')
```

what is the value of `y`? [1, ['a', 'b', 'c'], 3]

39. After executing the following sequence of statements

```
import copy
z = ['a', 'b']
x = [1, z, 3]
y = copy.deepcopy(x)
x[0] = 4
z.append('c')
```

what is the value of `y`? [1, ['a', 'b'], 3]

40. What is the **type** of `{}`? dict
41. What is the expression for an **empty set**? set()
42. Which of the following can or cannot be a **member of a set**? Why?
    - `'hello'` ok
    - `23` ok
    - `44.27` ok
    - `5e-3` ok
    - `2+4j` ok
    - `['Mary', 'had', 'a', 'little', 'lamb']` not legal "list"

- `('Mary', 'had', 'a', 'little', 'lamb')` ok
- `{'Mary', 'had', 'a', 'little', 'lamb'}` not legal "set"
- `{'Sun': 0, 'Mon': 1, 'Tue': 2, 'Wed': 3}` not legal "dict"
- `True` ok
- `False` ok
- `()` ok
- `[]` not legal "list"
- `{}` not legal "dict"

43. What is the value of `len(set('hello'))`? 4
44. What is the value of each of the following expressions?
    - `{1, 2} - {2, 3}` {1}
    - `{1, 2} | {2, 3}` {1, 2, 3}
    - `{1, 2} & {2, 3}` {2}
    - `{1, 2} ^ {2, 3}` {1,3}
45. What is the result of the following comparisons?
    - `{1, 2, 3} > {2, 3}` True
    - `{1, 2, 3} < {1, 2, 4}` False
    - `{1, 2, 2, 3} == {1, 2, 3}` True
    - `{1, 2, 4} != {4, 2, 1}` False
46. Assume `S = {1, 2, 3}`, what is the difference between
    `S = S | {3, 4}` and `S |= {3, 4}`?
    前者會先建立一個新的集合為{1,2,3,4}然後將 S 指向新集合
    後者直接將{1,2,3}做結構性修改為{1.2.3.4}
47. Assume `D = {'Sun': 0, 'Mon': 1, 'Tue': 2, 'Wed': 3}`
    - What is the value of `D['Mon']`? 1
    - What is the value of `D` after `D['Thu'] = 4`?
      {'Sun': 0, 'Mon': 1, 'Tue': 2, 'Wed': 3, 'Thu': 4}
    - Continuing with the previous statement, what is the value of `D` after `D['Sun'] = 7`?
      {'Sun': 7, 'Mon': 1, 'Tue': 2, 'Wed': 3, 'Thu': 4}
    - What happens if you attempt `print(D['Fri'])`?
      Traceback (most recent call last):
      File "<stdin>", line 1, in <module>
      KeyError: 'Fri'
48. Assume `D = {'Sun': 0, 'Mon': 1, 'Tue': 2, 'Wed': 3}`
    - What is the value of `D.keys()`
      `dict_keys(['Sun', 'Mon', 'Tue', 'Wed'])`
    - What is the value of `D.values()`
      `dict_values([0, 1, 2, 3])`
    - What is the value of `D.items()`
      `dict_items([('Sun', 0), ('Mon', 1), ('Tue', 2), ('Wed', 3)])`

49. Assuming `D = {}`, which of the following is legal or not legal in Python? If not legal, why not?
    - `D[()] = 10` legal
    - `D['']  = {}` legal
    - `D[0] = ''` legal
    - `D[{}] = ()` not legal "dict cannot be key."
    - `D[[]] = set()` not legal "list cannot be key."
    - `D[:] = range(10)` not legal "slice cannot be key."
    - `D[-1] = [-1]` legal
    - `D[(())]  = [{}]` legal
50. How do you use dictionary comprehension to create a reverse mapping? For example, suppose `D = {'Sun': 0, 'Mon': 1, 'Tue': 2, 'Wed': 3, 'Thu': 4, 'Fri': 5, 'Sat': 6}`, create its reverse mapping whose value should be `{0: 'Sun', 1: 'Mon', 2: 'Tue', 3: 'Wed', 4: 'Thu', 5: 'Fri', 6: 'Sat'}`?
    {value:key for key,value in D.items()}

2. Programming Exercises

1. Write a program that prompts the user to input a sentence and a prohibited list. the program needs to report a list which include all alphabets in the sentence except who appear in the prohibited list. For example, (blue text = typed input, green highlight = program printout)

```
$ python3 Week5_q1.py
Enter a sentence: HelloPython
Enter prohibited list: ['l','P','h']
['H', 'e', 'o', 'y', 't', 'o', 'n']
```

```
$ python Week5_q1.py
Enter a sentence: HelloPython
Enter prohibited list: ['l','
['H', 'e', 'o', 'y', 't', 'o'
```

Note : Sentence is case sensitive.

2.