SC6

1. Definitions and Short Answers

1. Given the following if-statement

```
1  x = int(input('enter one number: '))
2  y = int(input('another number: '))
3  if y == 0:
4      print('sorry, cannot divide by 0!')
5  else:
6      print(f'{x} / {y} = {x/y}')
```

- What is the purpose of checking if y == 0?
  - Divide by 0 is undefined behavior.
- What is the purpose of calling the int() function on the first two lines? Will the code still work if you replace lines 1-2 with x = input('enter one number: ') y = input('another number: ') instead? Why or why not?
  - 把input的type從string轉成int
  - 不能，因為line3 string不能跟int比較

2. The following if-statements are not correct in Python syntax. What is wrong with them and how can they be fixed? Do not change the functionality.

```
if y == 0:
    # do nothing
else:
    print(f'{x} / {y} = {x/y}')

if y != 0:
    print(f'{x} / {y} = {x/y}')
else:
    # do nothing

if x == 0:
    print('zero, without checking y')
else if y == 0:
    print('sorry, cannot divide by 0!')
else:
    print(f'{x} / {y} = {x/y}')
```

```
if y == 0: print('sorry, cannot divide by zero') else:
    print(f'{x}/{y}={x/y}')
```

```
if y == 0:
    pass
else:
    print(f'{x} / {y} = {x/y}')

if y != 0:
    print(f'{x} / {y} = {x/y}')
else:
    pass

if x == 0:
    print('zero, without checking y')
elif y == 0:
    print('sorry, cannot divide by 0!')
else:
    print(f'{x} / {y} = {x/y}')

if y == 0:
    print('sorry, cannot divide by zero')
else:
    print(f'{x}/{y}={x/y}')
```

3. Given the dictionary

   eng = {'one': 1, 'two': 2, 'three': 3, 'four': 4}

   what is the **value** of the following expressions, and which ones cause **errors**?

- 4 in eng
  - False
- 'three' in eng
  - True
- ('two', 2) in eng
  - False
- {'one': 1} in eng
  - TypeError: unhashable type: 'dict’
- eng['three']

    - 3
- eng[0]
    - KeyError: 0
- eng[1]
    - KeyError: 1
- eng['two':2]
    - TypeError: unhashable type: 'slice’

4. Consider the code:

```
 1 english={'one':1,'pan':'鍋子','cabbage':'高麗菜','pie':'派餅'}
 2 spanish = {'uno':1, 'pan':'麵包','col':'高麗菜',  'pie':'腳'}
 3 french = {'une':1, 'toi':'你',  'col':'衣領',    'pie':'鵲'}
 4 word = input('enter word to look up: ')
 5 if word in english:
 6     print(f'in English: {english[word]}')
 7 elif word in spanish:
 8     print(f'in Spanish: {spanish[word]}')
 9 elif word in french:
10     print(f'in French: {french[word]}')
11 else:
12     print(f'{word} not found in dictionary')
```

When running the program, what happens when you type the following text when prompted?

- uno
    - 1
- pan
    - 鍋子
- toi
    - 你
- pie
    - 派餅

- col
  - 高麗菜
- cabbage
  - 高麗菜
- 1
  - 1 not found in dictionary
- ten
  - ten not found in dictionary

5. In Question #4, if you swap lines 7-8 with lines 9-10, will the code output exactly the same results for all inputs or different for some inputs? Which?

> 💡 pie & col will be different, the others remain the same.

6. If the code in Question#4 is modified so all elif clauses get replaced by if, as shown in the following code:

```
 1 english={'one':1,'pan':'鍋子','cabbage':'高麗菜','pie':'派餅'}
 2 spanish = {'uno':1, 'pan':'麵包','col':'高麗菜',  'pie':'腳'}
 3 french = {'une':1, 'toi':'你',  'col':'衣領',    'pie':'鵲'}
 4 word = input('enter word to look up: ')
 5 if word in english:
 6     print(f'in English: {english[word]}')
 7 if word in spanish:
 8     print(f'in Spanish: {spanish[word]}')
 9 if word in french:
10     print(f'in French: {french[word]}')
11 else:
12     print(f'{word} not found in dictionary')
```

Suppose the user enters a word that is in english but not in french, does the print statement on the last line still get executed? Why or why not?

> 💡 Yes, 因為9-12行應該看做一個if-else判斷。

7. Given the following code

```
passing = False
if int(input('enter grade: ')) >= 60:
    passing = True
```

- what is the equivalent statement that uses both if and else?

```
if int(input('enter grade: ')) >= 60:
    passing = True
else:
    passing = False
```

- what is the equivalent (assignment) statement that does not use if at all?

```
passing = int(input('enter grade: ')) >= 60
```

8. Consider the following version of the code

```
 1 word = input('enter word to look up: ')
 2 outList = []
 3 if word in english:
 4     outList.append(f'in English: {english[word]}')
 5 if word in spanish:
 6     outList.append(f'in Spanish: {spanish[word]}')
 7 if word in french:
 8     outList.append(f'in French: {french[word]}')
 9 if not outList:
10     print(f'{word} not found in dictionary')
11 else:
12     print('\n'.join(outList))
```

- If line 9 is changed to if outList:, then how should the rest of the code be updated so it behaves exactly the same?

```
 9 if outList:
10     print('\n'.join(outList))
11 else:
12     print(f'{word} not found in dictionary')
```

- What does line 12 do?
  - 把outList的每個東西，依序一次印一個，然後換行，再印下一個以此類推。
- Is line 2 really necessary?
  - Line 2 is not strictly necessary for the functionality of the code, as the `outList` list could be initialized at the first append operation.
- What if line 2 is removed and line 4 is replaced by outList = [f'in English: {english[word]}'] will the code still work the same way as before? if not, how can it be fixed by adding more code?
  - 如果單字不在english內，會出錯，因為就沒有outList了
  - 改法如下:

```
word = input('enter word to look up: ')
if word in english:
  outList = [f'in English: {english[word]}']
elif word in spanish:
  outList = [f'in Spanish: {spanish[word]}']
elif word in french:
  outList = [f'in French: {french[word]}']
if not outList:
  print(f'{word} not found in dictionary')
else:
  print('\n'.join(outList))
```

9. Convert the if-else statement in the previous question (lines 9-12) into a single print() with a **conditional expression** as its argument.

```
print( f'{word} not found in dictionary' if not outList \
  else '\n'.join(outList))
```

10. Convert the if-elif-elif-else statement with four suites in Question#4 (lines 5-12) into a single print() statement whose argument is a **conditional expression**.

```
 5 if word in english:
 6     print(f'in English: {english[word]}')
 7 elif word in spanish:
 8     print(f'in Spanish: {spanish[word]}')
 9 elif word in french:
```

```
10     print(f'in French: {french[word]}')
11 else:
12     print(f'{word} not found in dictionary')

print( f'in English: {english[word]}' if word in english else \
  f'in Spanish: {spanish[word]}' if word in spanish else \
  f'in French: {french[word]}' if word in french else \
  f'{word} not found in dictionary')
```

11. Is there a form of conditional expression that uses the elif keyword? If so, provide an example.

No, there is no form of conditional expression that uses the `elif` keyword.

12. Given the following nested if-statement

```
1  if x < 0:
2      if y > 3:
3          print('correct')
```

rewrite it as a single if statement by combining the two conditions.

```
if x < 0 && y > 3:
    print('correct')
```

13. Given the following function that returns a boolean value:

```
1  def test(y):
2      if y % 10:
3          return False
4      else:
5          return True
```

rewrite it as a single return statement by eliminating the if-else construct completely.

```
def test(y):
  return False if y%10 else True
```

14. Given the following function:

```
1  def fn(z):
2      if y % 10:
3           return True
4      if y > 400:
5          return False
6      return True
```

rewrite it as a single return statement by eliminating all if-constructs completely.

```
def fn(z):
  return y%10 or y>400
```

15. Write a Python function to concatenate a list of strings **using a while loop**.

```
1  def concat_str(L): # L is a list of strings
2      # initialize local variables:
3      idx = 0 # - an index into L to select one item at a time
4      s = '' # - a result variable initialized to empty string
5      while idx < len(s):  # fill in your condition
6          s += s[idx]# concatenate result with the next string in L
7          idx += 1 # increment the index variable
8      return s # return the result string
```

16. Write a Python function to concatenate a list of strings **using a for loop**.

```
1  def concat_str(L):  # L is a list of strings
2      # initialize local variables:
3      s = '' # - the result variable initialized to empty string
4      for ss in L :   # get one string at a time from L
5          s += ss # concatenate result with string from for loop
6      return s # return the result string
```

17. Given the source code

```
1  def index(L, val):
2      i = 0
3      while i < len(L):
4          if L[i] == val:
5              break
```

```
6           i += 1
7       return i if i < len(L) else -1
```

Rewrite the code so that it does not use a break to exit the loop but still uses the same code as line 7 to return the i value. Hint: you need a separate "flag" variable that can take a True or False value; it should be initialized before the loop and is tested as part of the while-condition. The flag should be set to cause a break out of the loop. Be sure the i value is correct after exiting the loop.

```
def index(L, val):
  i = 0
  flag = False
  while not flag:
    if L[i] == val:
      flag = True
    i += 1
  return i-1 if i < len(L) else -1
```

18. Convert the code from Problem #17 into a while-else construct (where the else clause is associated with the while construct rather than with an if clause)

```
def index(L, val):
    i = 0
    while i < len(L):
        if L[i] == val:
            break
        i += 1
    else:
        i = -1
    return i
```

19. Convert the code from Problem #17 into a while True: on line 3 and convert the while condition into an if condition inside the loop.

```
def index(L, val):
    i = 0
    while True:
        if i >= len(L):
            break;
        if L[i] == val:
            break
```

```
        i += 1
    return i if i < len(L) else -1
```

20. Referring to the same code as in Problem #17, what happens if you replace break on line 5 with continue? For example,

- >>> L = "abaca" >>> index(L, 'a') Does the function return a value or go into an infinite loop? Why?
    - 無窮迴圈，因為一開始碰到a之後，就無限的continue，i的值改不到。
- >>> L = "abcde" >>> index(L, 'z') Does the function return a value or go into an infinite loop and why? Try to predict the behavior of the code before you try running it to verify your answer.
    - return 5

21. In the code

```
1  def index(L, val):
2      i = 0
3      while i < len(L):
4          if L[i] == val:
5              break
6          i = i + 1
7      else:
8          return -1
9      return i
```

What causes the else suite (lines 7-8) to be executed? If line 5 is executed, will line 8 be executed?

> 💡 i 等於 len(L) 時，才執行line 7-8。
> 不會，line5跟line8只會有一個執行，因為是跳出迴圈的條件判斷，只會因為其中一種條件而跳出迴圈。

22. Given the code

```
1  ESdict = { 'one': 'uno', 'dos': 'two', 'three': 'tres'}
2  word = input('enter an English word:')
3  while word != '':
```

```
 4      if word in ESdict:
 5          print(f'{word} in Spanish is {ESdict[word]}')
 6      else:
 7          print(f'{word} not in dictionary')
 8      word = input('enter an English word:')
```

However, there is redundancy because line 2 and line 8 are exactly the same (except for indentation). How can this code be rewritten to eliminate this redundancy?

```
while True:
    word = input('enter an English word:')
    if word == '':
        break
    if word in ESdict:
          print(f'{word} in Spanish is {ESdict[word]}')
    else:
          print(f'{word} not in dictionary')
    word = input('enter an English word:')
```

23. Given the following code

```
 1  def StackInterpreter():
 2      L = []
 3      while True:
 4          line = input('command? ')
 5          words = line.split()
 6          if len(words) == 0:
 7              pass
 8          elif words[0] == 'show':
 9              print(L)
10          elif words[0] == 'push':
11              L.extend(words[1:])
12          elif words[0] == 'pop':
13              print(L.pop())
14          elif words[0] == 'quit':
15              break
16          else:
17              print('unknown command')
```

If the elif keywords on lines 8, 10, 12, 14 are replaced by if, will the code still behave the same way as far as the user is concerned? Why?

💡 No, the code will not behave the same way.
例如: 使用者輸入 push 1 2，除了push 1 2外，還會執行line17。

24. Using the same code from the previous problem, line 7 can be replaced by a single-keyword statement and the code still behaves the same way to the user. What is the keyword? Explain.

💡 The `continue` statement is used inside a loop to skip the rest of the code in the current iteration and move on to the next iteration. In the original code, if the `words` list is empty, the `pass` statement is used to do nothing and move on to the next iteration. This behavior can be achieved more cleanly using the `continue` statement.

25. Using the same code from the previous problem, line 15 can be replaced by a single-keyword statement and the code still behave the same way. What is it?

💡 return

26. Given the code

```
1  def StackInterpreter():
2      L = []
3      while True:
4          line = input('command? ')
5          words = line.split()
6          if len(words) == 0:
7              continue
8          if words[0] == 'show':
9              print(L)
10             continue
11         if words[0] == 'push':
12             L.extend(words[1:])
13             continue
14         if words[0] == 'pop':
15             print(L.pop())
16             continue
```

```
17          if words[0] == 'quit':
18              break
19          print('unknown command')
```

Why doesn't line 19 need to be inside an else-clause but can be an unconditional statement at the same level as all preceding if-statements?

> 💡 因為前面的if都有加continue 或 break，所以只要前面的if有執行，line19就一定不會執行，換句話講，只要前面的if都沒執行，就直接執行line19就好了。

27. Given the Python code

```
1  def product(L):
2      p = 1
3      for i in range(len(L)):
4          p = p * L[i]
5      return p
```

- Rewrite it as a while-loop

```
def product(L):
    p = 1
    i = 0
    while i < len(L):
        p = p * L[i]
        i += 1
    return p
```

- Rewrite it as a Pythonic for-loop

```
def product(L):
    p = 1
    i = 0
    for i in L:
        p = p * i
    return p
```

28. What is the value of the expression

- list(enumerate("abcde"))
  - `[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd'), (4, 'e')]`
- list(enumerate(range(5, 10)))
  - `[(0, 5), (1, 6), (2, 7), (3, 8), (4, 9)]`
- list(enumerate(['Sun', 'Mon', 'Tue', 'Wed']))
  - `[(0, 'Sun'), (1, 'Mon'), (2, 'Tue'), (3, 'Wed')]`
- list(enumerate(['Jan', 'Feb', 'Mar'], start=1))
  - `[(1, 'Jan'), (2, 'Feb'), (3, 'Mar')]`

29. Rewrite the following code using a Pythonic for-loop:

```
1  def find(L, val):
2      i = 0
3      while i < len(L):
4          if L[i] == val:
5              return i
6          i = i + 1
7      return -1
```

```
def find(L, val):
    i = 0
    for i in L:
        if i == val:
            return i
    return -1
```