Exam
Object Oriented Programming 1 - Python
BTH000

Time allowed: Three hours.
No books or notes are allowed by the students.
(International College, ZJUT, provides an English-Chinese dictionary in the exam room)

Total Points: 100

Important!

- All questions are related to Python Programming
- The questions are not ordered by difficulty, so if you get stuck on one question please go ahead with the next. You can always go back if there is time.

Good Luck!

1. Short answers

Σ: 22

(a) What is the value of the Python expression? (2p)

```
5 % 3
```

Answer: .....

(b) What is the value of the Python expression? (2p)

```
5 // 3
```

Answer: .....

(c) What is the value of the Python expression? (2p)

```
3 / 2
```

Answer: .....

(d) Give two important differences between ***list*** and ***tuple*** types in Python. (2p)

Answer: .....
. .....

(e) What statement(s) creates a dictionary? (2p)

Mark your choice, one or more alternatives.

A. `my_dict = {}`

B. `my_dict = {"Emma": 23, "Tobias": 26}`

C. `my_dict = {23: "Emma", 26: "Tobias"}`

D. `my_dict = ("Emma": 23, "Tobias": 26)`

(f) What type is the value put in variable `num`? (2p)

```
1 num_lst = [1, 2, 3.5, 4]
2 num = num_lst[3]
```

Answer: .....

(g) What is the difference between a *statement* and an *expression*? (2p)

.....
.....
.....

(h) What is printed when running the following statement? (2p)

```
print([1, 5, 16, 15, 5, 25, 30][2:4])
```

Answer: .....

(i) What does Python output when the commands are run interactively? (2p)

```
>>> a = 1
>>> b = a
>>> a += 1
>>> a == b
```

Answer: .....

(j) (2p)

```
class Pokemon:
    def __init__(self, hp):
        self._name = "Porygon"
        self._hp = hp

    def evolve(self):
        self._name += "2"
        self._hp *= 2
```

How do you call `evolve` for a `Pokemon` object `pokemon`?

Mark you choice.

A. `evolve(pokemon)`
B. `pokemon.evolve()`
C. `p = evolve()`
D. `evolve() = pokemon`
E. None of the other options is correct

(k) What is the the name of the first parameter in the parameter list of a method by convention? **Mark** you choice. (2p)

A. `me`
B. `__init__`
C. `this`
D. `self`
E. `append`
F. `None`

2. Explain

Σ: 16

(a) Explain the concept shared reference (*aliasing*) with help from the run example below. Explain everything that happens regarding the names, memory addresses and values of the variables. (4p)

```
>>> a = [1, 2, 3]
>>> b = a
>>> c = 100
>>> d = c
>>> c = 200
>>> d
100
>>> b[2]
3
>>> b[2] += 2
>>> b
[1, 2, 5]
>>> a
[1, 2, 5]
```

(b) What is the difference between *deep copy* and *shallow copy*. (4p)

(c) How do you normally call the constructor of a class named **MyClass**, and what happens when you do so? (4p)

(d) Explain what the concept *encapsulation* in Object Oriented Programming means. (4p)

3. Read Python Code

Σ: 10

(a) What will be printed by the following program? (3p)
Also explain what the function `foo` is doing.

```
1  def foo(lst):
2      lst2 = []
3      while len(lst) > 0:
4          index = 0
5          for i in range(0,len(lst)):
6              if lst[i] >= lst[index]:
7                  index = i
8          lst2.append(lst[index])
9          lst.remove(lst[index])
10     return lst2
11
12 print(foo([1,2,3]))
13 print(foo(["Bob","Alice","Alicia"]))
14
15 print(foo([1,2,3]))
16 print(foo(["Bob","Alice","Alicia"]))
```

.....

.....

.....

.....

(b) What will this program print? (3p)

```
1 alist = []
2 for i in range(1, 14, 2):
3     alist.append(i)
4 print(alist[7:2:-2])
```

.....

(c) What will the following code print? (4p)

```
1  def function(a_list):
2      result = True
3      for k in range(len(a_list)):
4          result = decr(a_list, k) and result
5      return result
6
7
8  def decr(nums, k):
9      nums[k] = nums[k] -1
10     return nums[k] >= 0
11
12
13 if __name__ == "__main__":
14     numbers = [1, 2, 3, 4]
15     print(function(numbers))
16     print(numbers)
17     print(function(numbers))
18     print(numbers)
```

.....

.....

.....

.....

4. Debugging Code

Σ: 6

The function defined below is buggy and does not work. There are (at least) two bugs in it. In order to find the bugs, we have added several `print()` calls throughout the code. (6p)

```
1  def   time_to_minutes(string):
2      """Returns:  minutes since midnight
3      Examples:       '2:45 PM'   =>  14*60+45 = 885
4                      '9:05 AM'   =>  9*60+5 = 545
5                      '12:00 AM'   =>  0
6      Prerequisite: string is  in  12-hour format;
7      <hours>:<min> AM/PM"""
8
9      # Find the separators
10     pos1 = string.index(':'  )
11     print(f'pos1 is {pos1}')
12     pos2 = string.index(' ')
13     print(f'pos2 is {pos2}')
14
15     # Get hour and convert to int
16     hour = string[:pos1]
17     print(f'hour is {hour}')
18     hour = int (hour)
19     print(f'hour is  {hour}')
20
21     # Adjust hour to be correct.
22     suffix = string[pos2 + 1:]
23     print(f'suffix is  {suffix}')
24     if suffix ==  'PM':
25         hoar = hour + 12
26     elif hour ==  12:
27         hour = 0
28     print(f'hour is  {hour}')
29
30     # Get min and convert to  int
31     mins = string[pos1:pos2]
32     print(f'mins is {mins}')
33     mins = int(mins)
34     print(f'mins is {mins}')
35     return hour * 60 + mins
```

The problem continues on next page ...

The result of running the code with these prints is shown below. Using this information as a guide, **identify and fix the bugs**.

```
>>> time_to_minutes('2:45 PM')
pos1 is 1
pos2 is 4
hour is 2
hour is 2
suffix is PM
hour is 2
mins is :45

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 33, in time_to_minutes
ValueError: invalid literal for int() with base 10: ':45'
```

Your answer:

5. Write Some Code

Σ: 28

(a) The given code shows a nested complicated **if** statement. (4p)
Rewrite the **if** statement in a simpler way, that is logically equal. It should be completely *without nestings*, but *may* contain **one or more elif** and **an else**.

```
1  if choice > 120:
2      print("Green!")
3  else:
4      if choice > 80:
5          print("Blue!")
6      else:
7          if choice < 32:
8              print("Red!")
9          else:
10             print("Orange!")
```

(b) Write a function named `is_even(param)` that, given a single integer parameter, returns `True` if the parameter is an even number, `False` otherwise. (3p)

(c) Write a function `find_smallest(lst)` that takes a list of numbers as a parameter and returns the smallest of the number value. If the list is empty the function should return `None` (6p)

(d) Write a function `find_side(area, side)`, that given the parameters area (area of a rectangle) och `side` (one side of the rectangle) calculates and returns a floating point number that gives the size of the other side of the rectangle. (5p)

Also write some code to test doing the following:

1. Read a float number (rectangle area) from terminal input.
2. Read a float number (rectangle side) from terminal input.
3. Call the function and use the two inputted numbers as arguments to the function.
4. Print the returned result from the function call to display it in the terminal.

(e) Below follows the body of a function definition. Suggest a suitable head for the function (with name and parameters). Also provide a suitable doc-string for the function. (4p)

```
# given function body
    result = 1
    if y < 0:
        x = 1/x
        y = -y
    for i in range(y):
        result *= x
    return result
```

(f) A palindrome sentence is a sentence having all letters mirrored and can be read as the same forwards or backwards if you ignore letter case and omit blanks and other special characters that are not letters (that is punctuation, white space, figures etc.). (6p)

Examples of palindrome sentences:

Madam I'm Adam.

Was it a car or a cat I saw?

To do:

Write a function `is_palindrome(string)` that takes in a string (string is possibly empty, and the function should not crash if it is). The string contains the sentence to examine.

If the string is a palindrome sentence the function should return `True`, `False` otherwise.

Hint: You can use some built-in string methods in Python. To see if a character is a letter, you can use the string method `isalpha()`. If you like to convert letters to uppercase, you can use the string method `upper()`.

6. Classes and Objects

Σ: 18

(a) Define a new class `Matrix2x2` representing a two-dimensional matrix. The constructor should take the four matrix elements as parameters `x00`, `x01`, `x10` and, `x11`. The internal representation of the matrix is up to you. (4p)

Also give example code that creates an object of class `Matrix2x2`.

(b) Write the magic method `__str__` that returns a nice string representation of the matrix, making an object printable in the terminal. (5p)

(c) Write a getter method for an element by giving index for row and column in the matrix. (3p)

(d) Write another method for `Matrix2x2`, called `determinant`, that calculates and returns the determinant of of the object. The determinant of a 2D matrix (3p)
$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

is calculated like this

$det(A) = ad - bc$

(e) Write a method for `Matrix2x2`, called `transpose`, that transposes the matrix.That is to rearrange the values in the matrix so the rows and columns are interchanged, and the matrix gets flipped over its diagonal. If matrix $\boldsymbol{A}$ is arranged as above the transposed version looks like this: $A^T = \begin{pmatrix} d & b \\ c & a \end{pmatrix}$ (3p)

Extra writing space: