WeHelp

Assignment - Week 2

Complete tasks by Python and JavaScript without using third party libraries.

Note: Python code should run on Python 3 or above version.

Task 1:

We have example messages from 6 persons in JSON format. There are at least 3 persons who are older than 17. Using a loop to find out those who are most probably older than 17 years old based on example messages. Print their names in the console.

Note: never change existing code.

Python

```
def find_and_print(messages):
        # write down your judgment rules in comments
        # your code here, based on your own rules
find_and_print({
        "Bob":"My name is Bob. I'm 18 years old.",
        "Mary":"Hello, glad to meet you.",
        "Copper":"I'm a college student. Nice to meet you.",
        "Leslie":"I am of legal age in Taiwan.",
        "Vivian":"I will vote for Donald Trump next week",
        "Jenny":"Good morning."
})
```

JavaScript

```
function findAndPrint(messages){
        // write down your judgment rules in comments
        // your code here, based on your own rules
}
findAndPrint({
        "Bob":"My name is Bob. I'm 18 years old.",
        "Mary":"Hello, glad to meet you.",
        "Copper":"I'm a college student. Nice to meet you.",
        "Leslie":"I am of legal age in Taiwan.",
        "Vivian":"I will vote for Donald Trump next week",
        "Jenny":"Good morning."
});
```

Task 2:

Using a loop to complete functions below to calculate the sum of bonus of all employees in TWD and print it.

1. Bonus should depend on salary, performance and role fields. Define your own rules and calculate a bonus for each employee based on it.
2. The sum of bonus of all employees cannot exceed 10000 TWD based on your rules and example data.
3. You can assume the USD to TWD Exchange Rate is 30.
4. Salary is default to TWD if there is no specific mark.

Note: never change existing code.

Python

```
def calculate_sum_of_bonus(data):
        # write down your bonus rules in comments
        # your code here, based on your own rules
calculate_sum_of_bonus({
        "employees":[
                {
                        "name":"John",
                        "salary":"1000USD",
                        "performance":"above average",
                        "role":"Engineer"
                },
                {
                        "name":"Bob",
                        "salary":60000,
                        "performance":"average",
                        "role":"CEO"
                },
                {
                        "name":"Jenny",
                        "salary":"50,000",
                        "performance":"below average",
                        "role":"Sales"
                }
        ]
}) # call calculate_sum_of_bonus function
```

JavaScript

```
function calculateSumOfBonus(data){
    // write down your bonus rule in comments
    // your code here, based on your own rules
}
calculateSumOfBonus({
    "employees":[
        {
            "name":"John",
            "salary":"1000USD",
            "performance":"above average",
            "role":"Engineer"
        },
        {
            "name":"Bob",
            "salary":60000,
            "performance":"average",
            "role":"CEO"
        },
        {
            "name":"Jenny",
            "salary":"50,000",
            "performance":"below average",
            "role":"Sales"
        }
    ]
}); // call calculateSumOfBonus function
```

Task 3:

Find out whose middle name is unique among all the names, and print it. You can assume every input is a Chinese name with 2 ~ 3 words. If there are only 2 words in a name, the middle name is defined as the second word.

Note: never change existing code.

Python

```
def func(*data):
        # your code here
func("彭大牆", "王明雅", "吳明") # print 彭大牆
func("郭靜雅", "王立強", "林靜宜", "郭立恆", "林花花") # print 林花花
func("郭宣雅", "林靜宜", "郭宣恆", "林靜花") # print 沒有
```

JavaScript

```
function func(...data){
        // your code here
}
func("彭大牆", "王明雅", "吳明"); // print 彭大牆
func("郭靜雅", "王立強", "林靜宜", "郭立恆", "林花花"); // print 林花花
func("郭宣雅", "林靜宜", "郭宣恆", "林靜花"); // print 沒有
```

Task 4：

There is a number sequence: 0, 4, 3, 7, 6, 10, 9, 13, 12, 16, 15, …

Find out the nth term in this sequence.

Note: never change existing code.

Python

```
def get_number(index):
        # your code here
get_number(1) # print 4
get_number(5) # print 10
get_number(10) # print 15
```

JavaScript

```
function getNumber(index){
        // your code here
}
getNumber(1); // print 4
getNumber(5); // print 10
getNumber(10); // print 15
```

Task 5 (Optional)：

Given available seats for each car of a train, status bitmap, and number of incoming passengers, writing a procedure to find out the index of the most fitted car to serve passengers. Print -1 if there is no car which can serve incoming passengers.

- Available Seats: list/array containing number of available seats for each car.
- Status Bitmap: list/array containing only 0 or 1. 1 means the corresponding car can serve passengers for now.
- Passenger Number: number of incoming passengers.

We can assume all incoming passengers should be served in the same car.

Note: never change existing code.

Python

```
def find_index_of_car(seats, status, number):
    # your code here
find_index_of_car([3, 1, 5, 4, 2], [0, 1, 0, 1, 1], 2) # print 4
find_index_of_car([1, 0, 5, 1, 3], [0, 1, 0, 1, 1], 4) # print -1
find_index_of_car([4, 6, 5, 8], [0, 1, 1, 1], 4) # print 2
```

JavaScript

```
function findIndexOfCar(seats, status, number){
    // your code here
}
findIndexOfCar([3, 1, 5, 4, 2], [0, 1, 0, 1, 1], 2); // print 4
findIndexOfCar([1, 0, 5, 1, 3], [0, 1, 0, 1, 1], 4); // print -1
findIndexOfCar([4, 6, 5, 8], [0, 1, 1, 1], 4); // print 2
```