no

|  |
| --- |
| **Instructions**  Download and extract the files from the **Lab3\_starting\_code.zip**. Save the files into a folder of your choice – you should obtain several \*.py files. Enter your code into the respective \*.py file to complete the lab questions. |

### Q1 [ \* ]: Code Tracing

Given the following codes, could you **predict** what is going to be the **output** of the code without running the code?

a) Run the code in q1a.py provided to **check** whether your guess is correct.

|  |
| --- |
| def perform\_magic1(condition, output):  if (condition):  print(output)  def perform\_magic2(condition, output):  if (condition):  print(output)  else:  print("The condition is false!")  def perform\_magic3(a, b):  if (a >= b):  print("a >= b")    if (a <= b):  print("a <= b")  a = 12  b = 15  perform\_magic1(a == b, "a == b")  perform\_magic2(a > b, "a > b")  perform\_magic3(a + 3, b) |

b) Run the code in q1b.py provided to **check** whether your guess is correct.

|  |
| --- |
| def compare\_values\_1(a, b):  if (a != b):  return True  else:  return False    def compare\_values\_2(a, b, c):  if (a == b + c):  print("a == b + c")  if (b == a + c):  print("b == a + c")  if (c == a + b):  print("c == a + b")  print(compare\_values\_1("1 + 2", "3"))  compare\_values\_2('1', '', '1') |

**Q2 [ \*\* ] : Online Shopping**

**Part 1**

Suppose you are purchasing some Christmas gifts on an online retail website. At the website, you get 10% discount if you are a member. If an item is on sale (regardless of whether you are a member or not), you get 5% discount. (So if you are a member and you are buying an item on sale, you get 15% discount.)

Write a function called calculate\_price in q2.py that takes in three parameters:

* orig\_price: the cost of the item that you are purchasing (a float)
* is\_member: a Boolean variable indicating whether you are a member or not
* is\_on\_sale: a Boolean variable indicating whether the item is on sale or not.

The function is to apply the discount appropriately and **return** the final discounted price of the item.

**Part 2**

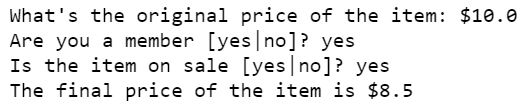
Write Python code by adding on to q2.py that prompts the user for the following information:

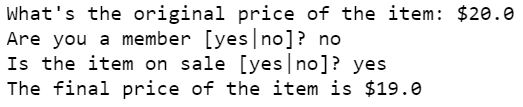
* The original price of the item
* Whether the user is a member or not
* Whether the item is on sale or not

The code then displays the final price of the item.

Call the calculate\_price function written in Part 1 to compute the final price of the item.

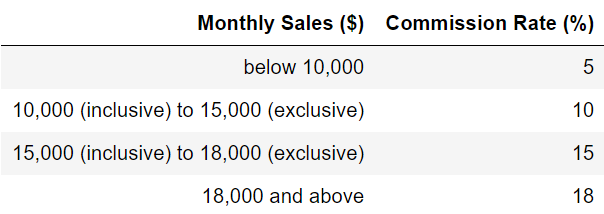
Two sample runs of the code can be found below. You can assume that for the second and the third inputs, the user is always going to enter either 'yes' or 'no' in lowercase.





### Q3 [ \*\* ] : Monthly Salary

A shop pays its sales staff based on each salesperson’s monthly sales. Each salesperson is paid a base monthly salary of $2000 plus commission based on the following table:



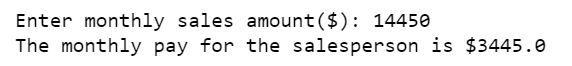
**Part 1**

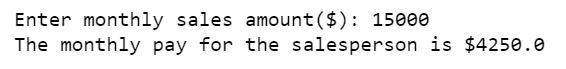
In q3.py, implement a function in Python called calculate\_salary that takes in a salesperson's monthly sales amount ($) and calculate this salesperson's total salary (base salary plus commission) for that month.

**Part 2**

Now prompt the user for a monthly sales amount. Call the function calculate\_salary written in Part 1 to calculate the total monthly salary and display the amount.

Two sample runs of the code are shown as follows:





### Q4 [ \*\* ]: Zoo Ticket

The Singapore Zoo offers the following prices to its visitors based on the visitor's age:

* Adult: $33 (full rate)
* Child (3 - 12, both inclusive): $22 (discounted rate)
* Senior Citizen (60 and above): $15 (discounted rate)

It is free for children below 3.

**Part 1**

Write a function in q4.py called get\_ticket\_info that takes in the age of a visitor. The function returns a tuple that contains two values:

* The price this person has to pay for his/her ticket
* A Boolean value indicating whether or not the person has got a discounted price. Note that children below 3 pay $0 and are also considered to receive a discounted price.

Use the test code provided in q4.py to test your implementation.

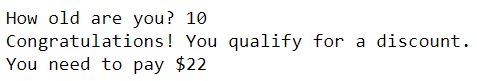
**Part 2**

Use the get\_ticket\_info function you have implemented in Part 1 to write a piece of code that carries out the following:

* Prompt the user for his/her age.
* If the user qualifies for a discounted ticket, display the message "Congratulations! You qualify for a discount." Then display the price of the ticket for this user no matter whether he/she qualifies for a discount.

Two sample runs of the code can be found below:

https://drive.google.com/uc?export=view&id=0B3N0YMArBdSBakxqZTBUb0hxSEU



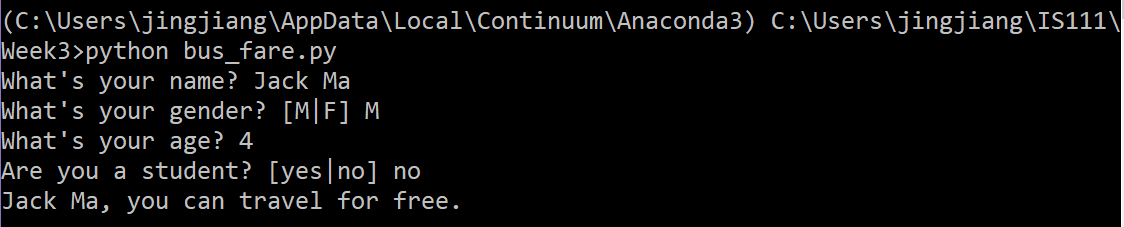
**Q5 [ \*\* ]: Bus Fares**

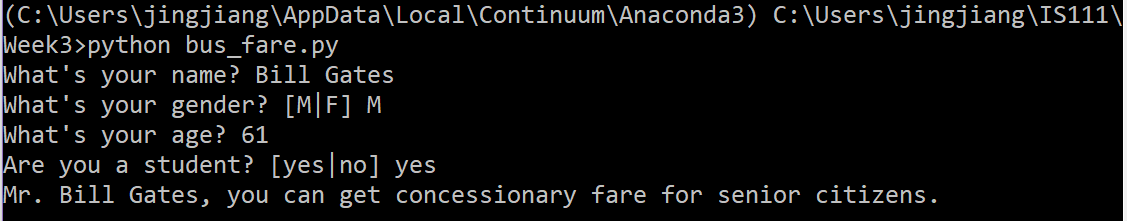
In q5.py, a function called get\_user\_info is implemented. **Do not modify this function.**

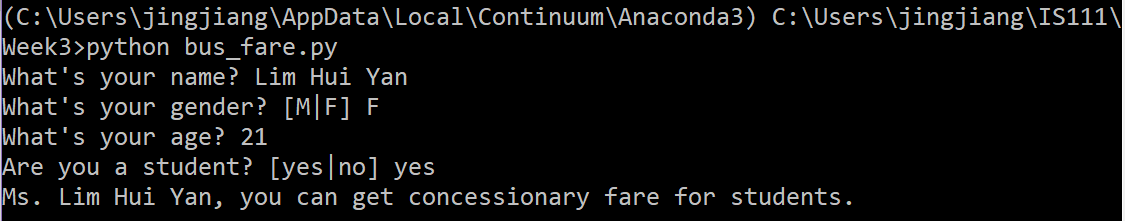
Write Python code in the same file to carry out the following:

* Call the function get\_user\_info to obtain some information from the user.
* Based on the users’ information, display the following messages accordingly:
  + If the user is 6 years old or below, display "<name>, you can travel for free."
  + If the user is 60 years old or above, display "<Mr.|Ms.> <name>, you can get concessionary fare for senior citizens."
  + If the user is above 6 and below 60, and the user is a student, display "<Mr.|Ms.> <name>, you can get concessionary fare for students."
  + For all the other users, display "<Mr.|Ms.> <name>, you need to pay full fare."

Three sample runs of the program can be found below:







**Q6 [ \*\*\* ]: Taxi Fares**

You are going to implement a taxi fare calculator.

The real taxi fare structure is fairly complex. We will instead use the following simplified fare structure for this question.

* The total taxi fare of a ride consists of two parts: (1) meter fare, and (2) surcharges.
* The meter fare of a ride consists of two parts: (1) flag-down fare, and (2) fare based on distance rate.
  + The flag-down fare covers the first **1km** of a ride. If the ride is shorter than 1km, the total meter fare is the flag-down fare.
  + If the ride is between **1km** and **9.8km**, the additional distance beyond 1km incurs a fare based on a distance rate per 400 meters. (Note that we use 9.8km instead of 10km here to simplify the calculation later.)
  + For any additional distance beyond **9.8km**, there is another distance rate per 350 meters.
* We consider two kinds of surcharges: (1) time-based surcharges, and (2) location surcharges.
  + We assume that if the ride is during a peak period, the time-based surcharge is 25% of the meter fare. If a ride is between midnight and 6am, the time-based surcharge is 50% of the meter fare. Other rides don’t have any time-based surcharge.

For example, suppose we have the following rates for a taxi ride:

* Flag-down: $3.50
* Beyond 1km and up to 9.8km: 22 cents per 400 meters
* After 9.8km: 22 cents per 350 meters

Suppose a taxi ride has a distance of 11400 meters. Also suppose that the ride is between midnight and 6am, and there is a location surcharge of $3.00.

Then the total taxi fare is calculated as follows:

* Meter fare:
  + Flag-down for the first 1km: $3.50
  + Next 8800 meters: $0.22 \* 22 = $4.84
  + Next 1600 meters: $0.22 \* 5 = $1.10
  + Total meter fare: $9.44
* Time-based surcharge:
  + $9.44 x 50% = $4.72
* Location surcharge:
  + $3.00
* Total fare: ($9.44 + $4.72 + $3.00) = $17.16

Create a new python script called taxi\_fare.py that helps a user calculate the taxi fare of a journey.

Two sample runs of the program can be found below.

**Note:**

* For the distance travelled, you can assume that the user is always going to enter a whole number.
* If the ride is during a peak period, the program should not ask the user whether the ride is between midnight and 6am.

Design function(s) to help you in your implementation.

