Session 2. Conditional Execution and Functions (with Solutions)

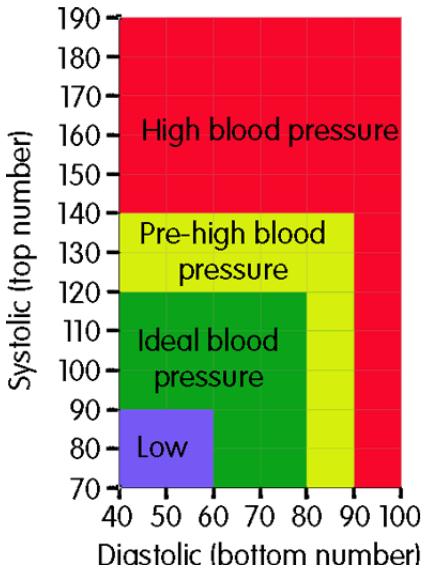
1. Conditional Execution

Example 1: The following program calculates a worker's weekly pay based on the following contract: for the first 40 hours, the hourly pay is 10. After that there is a 50 percent bonus per additional hour worked.

```
[1]: hours=float(input('Hours worked this week: '))
     if hours<=40:
         pay=hours*10
     else:
         pay=40*10+(hours-40)*15
     print('Total pay:',pay)
```

```
Hours worked this week: 42
Total pay: 430.0
```

Example 2: The following program asks for the user's systolic and diastolic blood pressure, and output one of `LOW`, `IDEAL`, `PRE-HIGH` or `HIGH` according to this chart:




```
[2]: high=float(input('Systolic blood pressure: '))
     low=float(input('Diastolic blood pressure: '))
     if low<=60 and high<=90:
         answer='LOW'
     elif low<=80 and high<=120:
         answer='IDEAL'
     elif low<=90 and high<=140:
         answer='PRE-HIGH'
     else:
         answer='HIGH'
     print('Your blood pressure is',answer)
```

```
Systolic blood pressure: 100
Diastolic blood pressure: 70
```

```
Your blood pressure is IDEAL
```

Example 3: The following program implements a primitive protocol for triaging asymptomatic individuals for risk of COVID-19.

```
[3]: Q1=input('Having been in one of the high risk countries in the past 14 days? (yes/no): '
     if Q1=='yes':
         print('You should be quarantined for 14 days.')
     else:
         Q2=input('Have you been in contact with any recent travelers to one of those countri
         if Q2=='yes':
             print('You should consider self-quarantine for 14 days.')
         else:
             print('You have low risks as long as you practice the regular precautions in a f
```

```
Having been in one of the high risk countries in the past 14 days? (yes/no): no
Have you been in contact with any recent travelers to one of those countries? (yes/no): no
You have low risks as long as you practice the regular precautions in a flu season.
```

Q1. Basestock Policy in Inventory Management

Write a program that asks the user for the current inventory level. If the inventory is at least equal to the target level of 100, then output "Sufficient inventory. No need to order."

Otherwise, output "Order x units", where x is the target minus the current inventory.

```
[4]: basestock=100
     inventory=input('Current inventory: ')
     inventory=int(inventory)
     if inventory>=basestock:
         print('Sufficient inventory. No need to order.')
     else:
         print(f'Order {basestock-inventory} units.')
```

```
Current inventory: 75
Order 25 units.
```

Q2. Blood Sugar Checker

Write a program that asks the user how many hours they have fasted and their current blood sugar level.

- If they have fasted less than 2 hours, then output "You need to fast at least 2 hours to perform this test."
- If they fasted at least 2 hours but less than 8 hours, then output "Your blood sugar level is high" if it is more than 140, and "Your blood sugar level is normal" otherwise.
- If they have fasted for at least 8 hours, then the threshold for normal sugar level changes from 140 to 100.

```
[5]: hours=float(input('How many hours have you fasted: '))
     level=float(input('What is your blood sugar level: '))
```

```
    high_msg='Your blood sugar level is high.'
    low_msg='Your blood suguar level is normal.'
    if hours<2:
        print('You need to fast at least 2 hours to perform this test.')
    elif hours<8:
        if level>140:
            print(high_msg)
        else:
            print(low_msg)
    else:
        if level>100:
            print(high_msg)
        else:
            print(low_msg)
```

```
How many hours have you fasted: 2
What is your blood sugar level: 110
Your blood suguar level is normal.
```

2. Functions

Example 4: The following function makes the logic in Example 1 reusable.

```
[6]: def calculateWage(hours,base=10,bonus=.5):
         ''' Calculates weekly wage '''
         if hours<=40:
             pay=hours*base
         else:
             pay=40*base+(hours-40)*base*(1+bonus)
         return pay

    print('Pay for 42 hours under default base and bonus:',calculateWage(42))
    print('Pay for 42 hours under $12/hour base and default bonus:',calculateWage(42,12))
    print('Pay for 42 hours under $12/hour base and 60% bonus:', calculateWage(42,12,.6))
    print('Pay for 42 hours with default base and 60% bonus:',calculateWage(42,bonus=0.6))
```

```
Pay for 42 hours under default base and bonus: 430.0
Pay for 42 hours under $12/hour base and default bonus: 516.0
Pay for 42 hours under $12/hour base and 60% bonus: 518.4
Pay for 42 hours with default base and 60% bonus: 432.0
```

```
[7]: help(calculateWage)
```

```
Help on function calculateWage in module __main__:

calculateWage(hours, base=10, bonus=0.5)
    Calculates weekly wage
```

Q3. Making the Solution for Q1 and Q2 Reusable

a) Write a function called "orderQuantity" with two input arguments

- **inventory** (default value 0): number of items on hand.
- **basestock** (default value 100): the target inventory level.

If inventory is at least equal to basestock, then return 0. Otherwise, return how many items you should order to meet the target. Include an appropriate docstring to explain what the function does.

```
[8]: def orderQuantity(inventory=0,basestock=100):
         '''Calculates order quantity given current inventory and basestock level
                   - inventory: number of items on hand
                   - basetock: the target inventory level
         '''
         # notes for programmer only. (comment)
         if inventory>=basestock:
             return 0
         else:
             return basestock-inventory
```

```
[9]: help(orderQuantity)
```

```
Help on function orderQuantity in module __main__:

orderQuantity(inventory=0, basestock=100)
    Calculates order quantity given current inventory and basestock level
    - inventory: number of items on hand
    - basetock: the target inventory level
```

```
[23]: # Code to test your function
      orderQuantity()
```

```
100
```

```
[11]: orderQuantity()+10
```

```
110
```

```
[12]: orderQuantity(25)
```

```
75
```

```
[13]: orderQuantity(51,50)
```

```
0
```

```
[14]: orderQuantity(basestock=200)
```

```
200
```

```
[15]: orderQuantity(inventory=80)
```

```
20
```

```
[16]: help(orderQuantity)
```

```
Help on function orderQuantity in module __main__:

orderQuantity(inventory=0, basestock=100)
    Calculates order quantity given current inventory and basestock level
    - inventory: number of items on hand
    - basetock: the target inventory level
```

b) Write a function called "bloodSugarCheck" with two input arguments:

- **hours**: the number hours the patient has fasted.
- **level**: the patient's blood sugar level.

Following the logic of Q2, the function should return a message which is one of the following:

1. 'You need to fast at least 2 hours to perform this test.'
2. 'Your blood sugar level is high.'
3. 'Your blood sugar level is normal.'

Include a suitable docstring.

```
[17]: def bloodSugarCheck(hours,level):
          # this is a comment (Computer completely ignores all comments)
          'doc-string: documentation about the function' # Computer doesnt ignore doc-string
          high_msg='Your blood sugar level is high.'
          low_msg='Your blood suguar level is normal.'
          if hours<2:
              return 'You need to fast at least 2 hours to perform this test.'
          elif hours<8:
              if level>140:
                  return high_msg
              else:
                  return low_msg
          else:
              if level>100:
                  return high_msg
              else:
                  return low_msg
```

```
[18]: help(bloodSugarCheck)
```

```
Help on function bloodSugarCheck in module __main__:

bloodSugarCheck(hours, level)
    doc-string: documentation about the function
```

```
[19]: # Test code
      bloodSugarCheck(1,100)
```

```
'You need to fast at least 2 hours to perform this test.'
```

```
[20]: bloodSugarCheck(2,110)
```

```
'Your blood suguar level is normal.'
```

```
[21]: bloodSugarCheck(8,110)
```

```
'Your blood sugar level is high.'
```

```
[22]: help(bloodSugarCheck)
```

```
Help on function bloodSugarCheck in module __main__:

bloodSugarCheck(hours, level)
    doc-string: documentation about the function
```