

Assignment -2

Python Programming

Assignment Date	29 September 2022
Student Name	Mr. Hadhi Sulaiman
Team ID	PNT2022TMID44152
Student Roll Number	724019104005
Maximum Marks	2 Marks

Question-1:

Build a python code, Assume you get temperature and humidity values (generated with random functions to a variable) and write a condition to continuously detect alarm in case of high temperature

Solution:

```
import random
Temperature=random.randint(1,100)
Humidity=random.randint(1,100)
print(Temperature)
print(Humidity)
if((Temperature>30)&(Humidity>40)):
    print("Temperature and Humidity are HIGH!!! ")
    print("***ALARM ON**")
else:
    print("Temperature and Humidity are NORMAL!!! ")
    print("***ALARM OFF**")
```



```
main.py  Run Shell Clear
1 import random
2 Temperature=random.randint(1,100)
3 Humidity=random.randint(1,100)
4 print(Temperature)
5 print(Humidity)
6 if((Temperature>30)&(Humidity>40)):
7     print("Temperature and Humidity are HIGH!!! ")
8     print("***ALARM ON**")
9 else:
10    print("Temperature and Humidity are NORMAL!!! ")
11    print("***ALARM OFF**")
12
10
8
Temperature and Humidity are NORMAL!!!
**ALARM OFF**
```



```
main.py  Run Shell Clear
1 import random
2 Temperature=random.randint(1,100)
3 Humidity=random.randint(1,100)
4 print(Temperature)
5 print(Humidity)
6 if((Temperature>30)&(Humidity>40)):
7     print("Temperature and Humidity are HIGH!!! ")
8     print("***ALARM ON**")
9 else:
10    print("Temperature and Humidity are NORMAL!!! ")
11    print("***ALARM OFF**")
12
44
54
Temperature and Humidity are HIGH!!!
**ALARM ON**
```

Question-2:

Write a Python program to calculate number of days between two dates.

Solution:

```
from datetime import date
f_date = date(2014, 7, 2)
l_date = date(2014, 7, 11)
delta = l_date - f_date
print(delta.days)
from datetime import date
f_date = date(2014, 7, 2)
```



Question-3:

Write a Python program to test whether a number is within 100 of 1000 or 2000

Solution:

```
def near_thousand(n):  
    return ((abs(1000 - n) <= 100) or (abs(2000 - n) <= 100))  
print(near_thousand(1000))  
print(near_thousand(900))  
print(near_thousand(800))  
print(near_thousand(2200))
```



The screenshot shows a Trinket Python IDE interface. The code editor on the left contains the following Python code:

```
1 def near_thousand(n):  
2     return ((abs(1000 - n) <= 100) or (abs(2000 - n) <= 100))  
3  
4 print(near_thousand(1000))  
5 print(near_thousand(900))  
6 print(near_thousand(800))  
7 print(near_thousand(2200))
```

The output console on the right shows the results of the program execution:

```
True  
True  
False  
False
```

A tooltip in the center of the IDE says "Press Esc to exit full screen".

Question-4:

Write a NumPy program to partition a given array in a specified position and move all the smaller elements values to the left of the partition, and the remaining values to the right, in arbitrary order (based on random choice).

Solution:

```
import numpy as np
nums = np.array([70, 50, 20, 30, -11, 60, 50, 40])
print("Original array:")
print(nums)
print("\nAfter partitioning on 4 the position:")
print(np.partition(nums, 4))
```



The screenshot shows a Trinket Python IDE interface. On the left, a code editor displays the following Python code:

```
1 import numpy as np
2 nums = np.array([70, 50, 20, 30, -11, 60, 50, 40])
3 print("Original array:")
4 print(nums)
5 print("\nAfter partitioning on 4 the position:")
6 print(np.partition(nums, 4))
7
8
```

On the right, the output console shows the results of the program execution:

```
Original array:
[ 70  50  20  30 -11  60  50  40]
After partitioning on 4 the position:
[-11  20  30  40  50  50  60  70]
```

A tooltip提示 "Press Esc to exit full screen" is visible over the code editor.

Question-5:

Write a Python program to reverse the digits of a given number and add it to the original, If the sum is not a palindrome repeat this procedure.

Solution:

```
def rev_number(n):
```

```
    s = 0
```

```
    while True:
```

```
        k = str(n)
```

```
        if k == k[::-1]:
```

```
            break
```

```
        else:
```

```
            m = int(k[::-1])
```

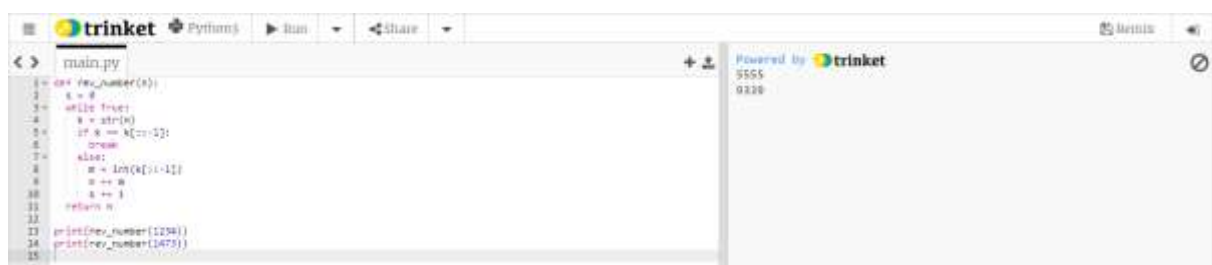
```
            n += m
```

```
            s += 1
```

```
    return n
```

```
print(rev_number(1234))
```

```
print(rev_number(1473))
```



The screenshot shows a Trinket Python IDE interface. On the left, the code editor displays the Python program for reversing a number and checking for a palindrome. The code includes a function `rev_number(n)` and two print statements: `print(rev_number(1234))` and `print(rev_number(1473))`. On the right, the output console shows the results of the program execution: `5555` and `0220`. The interface also includes a toolbar with buttons for running, sharing, and other IDE functions.