Fall 2016

Faculty of Engineering
Computer Engineering Department
CMP 402

Machine Intelligence
Python – Part 2

Objectives:

By the end of this session, students should be able to:

- Use definite and indefinite loops for iterations (for, while).
- Use conditional statements (if, if/elif/else, try/except).
- Use change of flow statements (break, continue).
- Know when to "pass" statement is used.
- Use "input" function to take user input.

Sample code: control_flow.py

```
import string
import random

from numpy import square



score = random.randint(1,100)

#if/elif/else
if score >= 90:
    print (str(score) +" is more than
90: A")
elif score >= 80:
    print (str(score) +" is between 80
and 90: B")
elif score >= 70:
    print (str(score) +" is between 70
and 80: C")
elif score > 60:
    print (str(score) +" is between 60
and 70: D")
```

```
96 is more than 90: A
```

```
else:
    print (str(score) +" is less than
60: Failed")

#NO switch case

#for definite loops
print("Numbers in range of 5
[0,1,2,3,4]")
for i in range(5):
    print(i, end=" ")

print("\nSquare of numbers in range of 5
[0,1,2,3,4]")
for i in square(range(5)):
    print(i, end=" ")
names=['Ahmed','Amira','Mohamed','Mohsen
','Abeer']
names+=['Mona','Dalia','Menna','Yasmine'
,'Malak','Omar']
print("")
print(sorted(names))
print(sorted(names,reverse=True))
names.remove('Menna')
print(names)
```

```
Numbers in range of 5 [0,1,2,3,4]
0 1 2 3 4

Square of numbers in range of 5
[0,1,2,3,4]
0 1 4 9 16
```

```
['Abeer', 'Ahmed', 'Amira', 'Dalia',
'Malak', 'Menna', 'Mohamed', 'Mohsen',
'Mona', 'Omar', 'Yasmine']
['Yasmine', 'Omar', 'Mona', 'Mohsen',
'Mohamed', 'Menna', 'Malak', 'Dalia',
'Amira', 'Ahmed', 'Abeer']
['Ahmed', 'Amira', 'Mohamed', 'Mohsen',
'Abeer', 'Mona', 'Dalia', 'Yasmine',
'Malak', 'Omar']
```

```
[print(n) for n in names]
```

```
Ahmed
Amira
Mohamed
Mohsen
Abeer
Mona
Dalia
Yasmine
Malak
Omar
```

```
for x in string.ascii_lowercase:
    count = 0
    for name in names:

if(name.casefold().startswith(x)):
#if_statement
            count+=1
    if(count > 0):
            print(str(x).upper()+" :
"+str(count))
```

```
A : 3
D : 1
M : 4
O : 1
Y : 1
```

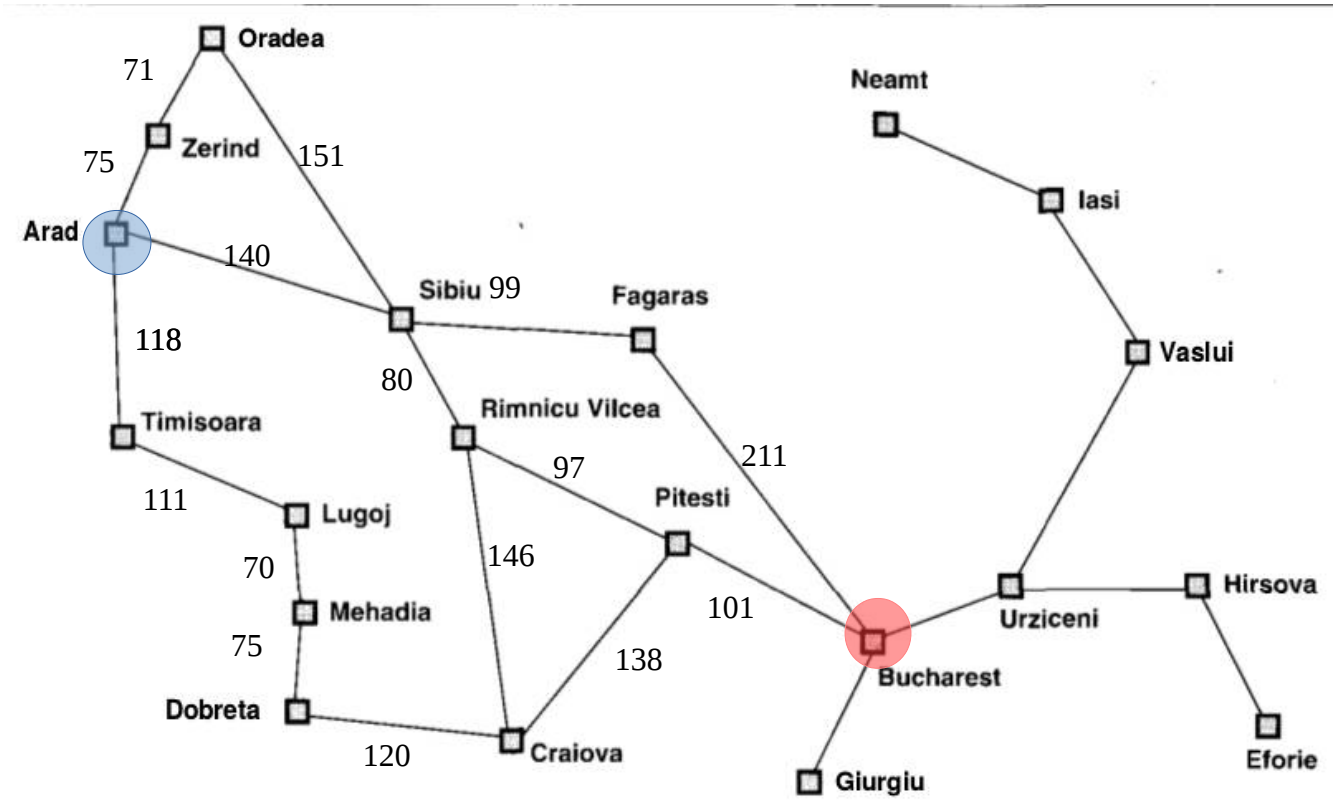
```python
#while indefinite loop
temp = 24
low_threshold = 20
high_threshold = 30
while temp>low_threshold: #indefinite
loop
    temp -= 1
    print("Current temperature is
"+str(temp))


#take input from user
temp = input("Enter the current
temperature:")
while True: #indefinite loop
    if (int(temp)<low_threshold or
int(temp) >high_threshold):
        break   #break statement
    print("While loop: Current
temperature is "+str(temp))
    temp = input("Enter the current
temperature:")

#continue
for letter in 'Python':
    if letter == 'h':
        continue #continue statement
    print ('Current Letter :'+ letter)

#pass
for letter in 'Python':
    if (letter == 'h'):
        pass
        print ('This is pass block')
    print ('Current Letter :'+letter)

#try/except
try: #to avoid ValueError: invalid
literal for int() with base 10
    int_value = input("Enter an integer
please :")
    print(int(int_value))
except:
    print("The value entered is not an
integer!")
finally:
    print("Executed in both cases.")
```

```
Current temperature is 23
Current temperature is 22
Current temperature is 21
Current temperature is 20

Enter the current temperature:27
While loop: Current temperature is 27
Enter the current temperature:35




Current Letter :P
Current Letter :y
Current Letter :t
Current Letter :o
Current Letter :n


Current Letter :P
Current Letter :y
Current Letter :t
This is pass block
Current Letter :h
Current Letter :o
Current Letter :n


Enter an integer please :4
4
Executed in both cases.


Enter an integer please :k
The value entered is not an integer!
Executed in both cases.
```

Listing 1.1: control_flow.py

MI AIMA code:

For Romania map example in book, we will use test_romania_problem.py and search.py (which imports utils.py and grid.py):




```
print(breadth_first_search(romania_problem).solution())
print(uniform_cost_search(romania_problem).solution())
print(depth_first_graph_search(romania_problem).solution())
```

```
['Sibiu', 'Fagaras', 'Bucharest']
['Sibiu', 'Rimnicu', 'Pitesti', 'Bucharest']
['Timisoara', 'Lugoj', 'Mehadia', 'Drobeta', 'Craiova', 'Pitesti', 'Bucharest']
```

```
['Sibiu', 'Fagaras', 'Bucharest']
['Sibiu', 'Rimnicu', 'Pitesti', 'Bucharest']
['Sibiu', 'Fagaras', 'Bucharest']
```

```
['Sibiu', 'Fagaras', 'Bucharest']
['Sibiu', 'Rimnicu', 'Pitesti', 'Bucharest']
['Sibiu', 'Rimnicu', 'Pitesti', 'Bucharest']
```

Listing 1.2: Running test_romania_problem.py many times may produce different results

Practice

Please solve the following exercises :

1. Write python code to print even, odd numbers and prime numbers in a range (0,limit). Take limit as input from user.

2. When executing test_romania_problem.py many times, you will find that depth_first_graph_search outputs different solutions each time as highlighted in Listing 1.2. Debug the code in search.py to find the reason behind this behavior. Modify the code in to have the same output each time you run the program.