Lab 12

We encourage you to work together on the Pre Lab. The Pre Lab is not graded but will help you prepare for your lab session. If you have any questions on the material of the Prelab, first check the book and recitation slides. If you continue to have any doubts or questions about it, please email your recitation TA or the course instructor.

This prelab introduces Classes. It provides you with a Dice class and asks you to write a main function which uses the Dice class to get the appropriate output. It also introduces you to the concept of Stacks.

Here is the PreLab12

Environment Setup

Go to your working directory in “data.cs.purdue.edu” and create a directory “cs177/lab12”. Refer to the first lab material if you need to remember the steps to do so. Then start the IDLE Python Interpreter.

Exercise 1

Write a class to represent the geometric solid cuboid. A cuboid is a closed box composed of three pairs of rectangular faces placed opposite each other and joined at right angles to each other. Given the length, width and height of a cuboid, we can calculate the following:

\[
\text{Lateral Surface Area} = 2 \times (\text{length} \times \text{height} + \text{width} \times \text{height}) \\
\text{Surface Area} = 2 \times (\text{length} \times \text{width} + \text{width} \times \text{height} + \text{height} \times \text{length}) \\
\text{Volume} = \text{length} \times \text{width} \times \text{height}
\]

Your class should implement the following methods:

```python
__init__(self, length, width, height)  # Creates a cuboid with the given length, width and height.
getDimensions(self)  # Returns the length, width and height of this
```

Computer Science Courses - http://courses.cs.purdue.edu/
cuboid.

LateralSurfaceArea(self)  #Returns the Lateral Surface Area of the cuboid
SurfaceArea(self)         #Returns the Surface Area of the cuboid.
Volume(self)              #Returns the volume of the cuboid.

Name the class as Cuboid and save the program as cuboid.py in cs177/lab12/ directory.

Then write the following program and save it as cuboidMain.py in cs177/lab12/ directory.

```
#cuboidMain.py
from cuboid import *

def main():
    cb = Cuboid(3, 4, 5)
    l, w, h = cb.getDimensions()
    print("The length of the cuboid is", l)
    print("The width of the cuboid is", w)
    print("The height of the cuboid is", h)
    print("lateral surface area=", cb.LateralSurfaceArea())
    print("surface area=", cb.SurfaceArea())
    print("volume=", cb.Volume())

main()
```

Then when you run the cuboidMain.py, you should get the following output:

**Output:**

```
The length of the cuboid is 3
The width of the cuboid is 4
The height of the cuboid is 5
lateral surface area= 70
surface area= 94
volume= 60
```

**Exercise 2**

Now we will write a class called Stack. This class will represent some of the operations that can be performed on a list, like the following:

1) checking if the list is empty or not
2) adding elements at the end of the list
3) if the list is not empty, then removing the last element from the list. If the list is empty, you should print that the list is Empty.
4) printing the contents of the list

Your class should look like this:

```python
class Stack:
    # takes in the object as input argument
    # will not return anything
    def __init__(self):
        # initialise an instance variable to an empty list.
        self.list = []

    # takes in the object as input argument
    # return value Type: True or False
    def isEmpty(self):
        # check if the list is empty or not. If empty, return True else return False
        if not self.list:
            return True
        else:
            return False

    # takes in the object as the first argument
    # takes the element to be inserted into the list as the second argument
    # should not return anything
    def push(self, x):
        # add the element to be inserted at the end of the list
        self.list.append(x)

    # takes in the object as the input argument
    # if the list is not empty then returns the last element deleted from the list. If the list is empty, don't return anything
    def pop(self):
        # check if the list is Empty
        # if Empty: print the list is empty
        # if the list is not empty, then remove the last element from the list and return it
        if self.list:
            return self.list.pop()
        else:
            return None

    # takes in the object as the input argument
    # should not return anything
    def printContents(self):
        # if the list is not empty, then print each element of the list
        if self.list:
            for x in self.list:
                print(x)
```

Name the class as Stack and save the program as `stack.py` in `cs177/lab12/` directory.

Then write the following program and save it as `StackMain.py` in `cs177/lab12/` directory.

```python
#StackMain.py

from stack import *

def main():
    s = Stack()
    s.push(1)
    s.pop()
```
s.pop()
s.push(2)
s.push(3)
s.push(4)
s.printContents()

main()

Then when you run the `StackMain.py`, you should get the following output:

**Output:**

The list is empty
The contents of the list is: 2 3 4

**Exercise 3**

In this Exercise, we will explore how to use a Class to create people and conduct various steps.

Your task is to complete the `sayHi()` and `howMany()` methods. Follow the instructions given to you under the methods.

Write lines of code as instructed under the main function and the output should be produced exactly how shown below.

```python
class Person:
    #Represents a person.
    population = 0

    def __init__(self, name):
        #Initializes the person's data.
        self.name = name
        print ('Initializing',self.name)

        # When this person is created, he/she
        # adds to the population
        Person.population += 1

    def __del__(self):
        #I am dying.
        print (self.name,"says bye.")

        Person.population -= 1

        if Person.population == 0:
            print ('I was the last one.')
        else:
            print ('There is/are still',Person.population , 'person/persons')
```
```python
left.)

def sayHi(self):
    # Greeting by the person.
    # Should print "Hi, my name is Ben" for example if the name
    is Ben


def howMany(self):
    # Prints the current population.
    # If there is only 1 person then it should print "I am the only one"
    # If there are 2 people for example, then print "We have 2
    people here"


def main():
    # Step 1: Ask for names of 2 people
    # Step 2: Initialize Person 1
    # Step 3: Use function sayHi() for Person 1
    # Step 4: Use howMany() for Person 1
    # Step 5: Initialize Person 2
    # Step 6: Use function sayHi() for Person 2
    # Step 7: Use howMany() for Person 2
    # Step 8: Say Hi to Person 1
    # Step 9: Use howMany() for Person 1
    # Step 10: Terminate Person 1
    # Step 11: Terminate Person 2

main()

Output:

Name of Person 1? Alec
Name of Person 2? Sarah

Initializing Alec
Hi, my name is Alec
I am the only person here.

Initializing Sarah
Hi, my name is Sarah
We have 2 people here.

Hi, my name is Alec
We have 2 people here.

Alec says bye.
There is/are still 1 person/persons left.
Sarah says bye.
I was the last one.

Save this program in cs177/lab12/people.py

**Turnin Instructions**

Run putty and login to data.cs.purdue.edu. Turn in your lab by typing:

```bash
$ cd cs177
$ turnin -v -c cs177=COMMON -p lab12 lab12
```

To verify that you did not submit a wrong file or an empty one, run the following command:

```bash
$ turnin -v -p lab12
```

**Grading Criteria**

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
</tr>
</tbody>
</table>

From: [http://courses.cs.purdue.edu/](http://courses.cs.purdue.edu/) - Computer Science Courses


Last update: 2015/11/12 11:36