CSI 31 Review and Practice

Topics:

1. Conditionals
2. Classes
3. Class diagrams

Example 1

Assume that *x*, *y* and *z* are real numbers.
How would you write the following conditions in Python?

(a) the product of *x* and *y* is not more than 10 and *z* is less than 7

(b) *x* is not a sum nor a difference of *y* and *z*

(c) negation of “*x* is not greater than *y* and *x* is not greater than *z*”

Example 2

Draw a class diagram for the following class:

```
class It:

  def __init__(self,a,b,c):

    self._f = a
    self._d = b
    self._g = c

  def operation(self,x):

    return self._f + x

  def getSum(self):

    return self._f + self._d + self._g

  def operation2(self,x):

    return self._f - y
```

Example 3

What does the following code output?

```
from copy import copy

class Apple:
  def __init__(self,a,b):
    self._n = a + a
    self._s = copy(b)
    self._s.append(a)

  def getInfo(self):
    return self._n,self._s
```

```
class Pear:
  def __init__(self,a,b):
    self._n = 3*a
    self._s = b
    self._s.append(15)

  def getInfo(self):
    return self._n, self._s
```

```
    def main():
      x, y = 10, [1,9,2]
      o = Apple(x,y)
      print("Apple object's info:",o.getInfo())
      print("x={0}, y={1}".format(x,y))
      m = Pear(x,y)
      print("Pear object's info:",m.getInfo())
      print("x={0}, y={1}".format(x,y))
    main()
```

Example 4

Given the definition of the class Me, which statements are correct with respect to “it is a bad style to directly access an instance variable outside a class definition” and which ones are not?

```
class Me:
    def __init__(self,a,b):
        self._name = a
        self._age = b
    def getAge(self):
        return self._age
    def getName(self):
        return self._name
    def setAge(self,value):
        self._age = value
    def setName(self,name):
        self._name = name
```

(a) `p1 = Me("Alan",59)`

(b) `p1._age = 60`

(c) `p1.setAge(60)`

(d) `print(p1._name, " is ", p1._age)`

Example 5

Find syntax errors and correct them (the program is 3 slides long).

```
class Thing:
  def __init__(a,b):

    self._n = a
    self._d = b

  def asString():

    return str(self._n) + ' / ' + str(self._d)

  def getNum():

    return self._n

  def getDen()

    return self._d
```

Example 5

Find syntax errors and correct them.

```
def add(f1,f2):
  if type(f1) = type(f2) = Thing:
    num = f1.getNum * f2.getDen() +
                 f2.getNum() * f1.getDen()
    den = f1.getDen() * f2.getDen()
    return Thing(num,den)

  else:
    return False
```

Example 5

Find syntax errors and correct them.

```
def main():
  f1 = Thing(1,2)
  f2 = Thing(2,3)

  print("let's create two fractions:)
  print(f1.asString(), end = "\t and \t")
  print(f2.asString())

  print("Their sum is {0:s}".
          format(add(f1,f2).asString()))

main()
```

Example 6

Create and test a Set class to represent a classical set. The sets should support the following methods:

Set(elements)

creates a set (elements are initial elements in the set);
Also recall that sets don’t have duplicates

addElement(x) adds element to the set (if it doesn’t belong to it)

deleteElement(x) removes x from the set, if present
If x is not element of the set, the set is left unchanged

member(x) returns true if x is in the set and false otherwise

intersection(set2) returns a new set containing just those elements that are common to this set and set2 (set ∩ set2).

union(set2) returns a new set containing all the elements that is in either of the sets (set ∪ set2)

subtract(set2) returns set – set2, i.e. a new set containing all the elements of this set that are not in set2.

Example 6

Create and test a Set class to represent a classical set. The sets should support the following methods:

Set(elements)

creates a set (elements are initial elements in the set);
Also recall that sets don’t have duplicates

addElement(x) adds ele it)

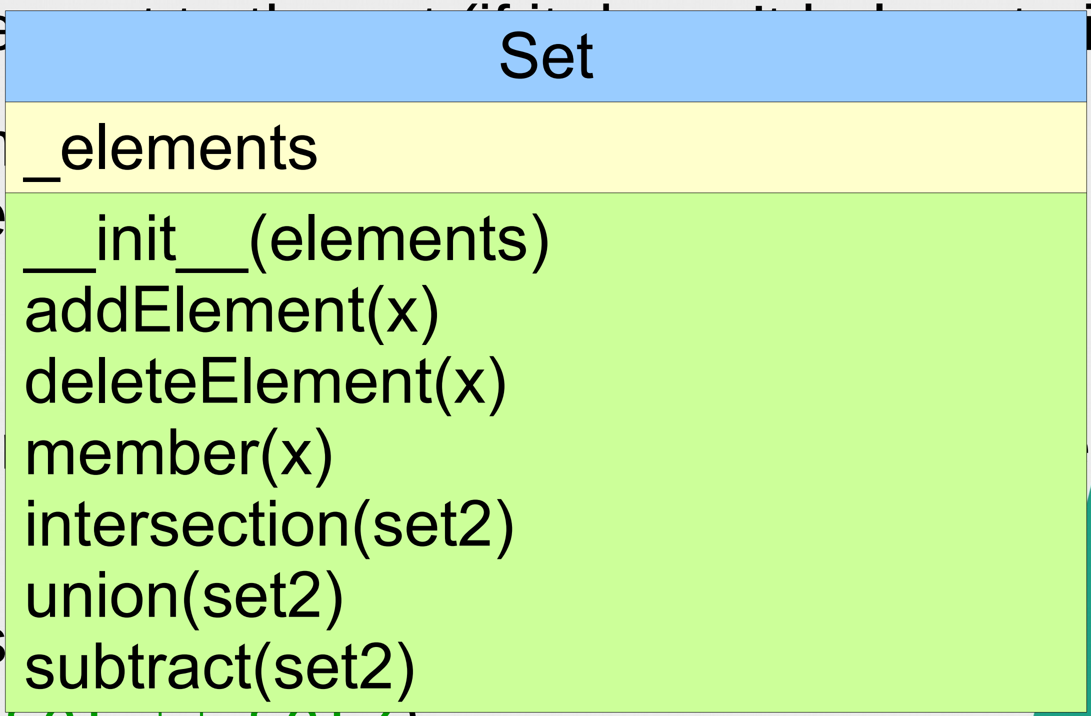
deleteElement(x) rem
If x is not element of the

member(x) returns tru

intersection(set2)
elements that are common

union(set2) returns
that is in either of the sets (set ∪ set2)

subtract(set2) returns set – set2, i.e. a new set containing all the elements of this set that are not in set2.



Example 6

Create and test a `Set` class to represent a classical set. The sets should support the following methods:

Write the definition of the `Set` class, then use the program to test it: testingSet.py

Example 7

Be ready to use a definition of a class to do something.