# Sample exam (not much was done to make this look nice)

# Solutions are in white, or you can change the solution style.

1. Ask the user for an integer. Then write a for loop in python that asks for that many doubles. Lastly, output the arrays values divided by 2, using integer division, on the same line separated with a space. (pulled form actual exam)

Example usages:

Cnt: 4  
Num: 1.5  
Num: 2  
Num: 4  
Num: 4.5  
0.0 1.0 2.0 2.0

Sample solution:

x = [0] \*5  
n = int(input("Cnt: "))  
for i in range(0,n):  
 y = float(input("Num: "))  
 x[i] = y // 2  
  
for i in range(0, n):  
 print(x[i], end = " ")

1. Write a python class that called Pumpkin. It must have the first letter capitalized. Pumpkin has the following:

* A constructor that takes a optional radius. The default radius is 15.
* A method called grow() that will increase the radius by 2%.
* A override str() function that returns “Pumpkin: radius X” where x is the current radius

2pt per line, 1.5 for syntax, 0.5 for purpose

|  |  |
| --- | --- |
|  |  |
| example solution  class Pumpkin:     def \_\_init\_\_(self, r = 15):         self.radius = r      def grow( self ):         self.radius \*= 1.02      def \_\_str\_\_(self):         return "Pumkin: radius " + str(self.radius) | |

1. How does a generator compare to an iterator? (1-2 sentences)

A generator can run forever while an iterator will stop at the end of its collection

1. Consider the following pseudocode, assuming nested functions and static scope. What does this program print? Separate the numbers with one space. (nearly pulled from exam)

g : integer

func B(a : integer)

x : integer

func A(n : integer)

g = n

func R(m : integer)

print x

x = x / 2 // integer division

if x > 1

R(m + 1)

else

A(m)

//back in B()

x = a \* a

R(1)

//back in main()

B(3)

print g

\_\_\_\_\_\_\_\_(9 4 2 3)

1. Consider the following pseudocode (I’ve used very similar problems in exams)

x = 1

y = 2

func add()

x = x + y

func second( m : method() )

local x = 2

m()

func first()

local y = 3

second(add)

first()

print x

* 1. What does this program print if the language uses static scoping?

\_\_\_\_\_\_\_\_(3)

* 1. What does it print if the language uses dynamic scoping with deep binding (closure-based dynamic scoping)?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(4) deep binding uses the current state (closure) when the method is made an argument is made. At the second (add), this means the local y and outer x. This is halfway better static and dynamic with shallow binding

* 1. What does it print if the language uses dynamic scoping with shallow binding (standard dynamic scoping)?\_\_\_\_\_\_\_\_

(1) shallow binding evaluates at its location of use. This is the more standard dynamic scoping. This means at x = x + y, it sees the local x and local y first, and uses them. This also means it set the local x! by the time we back out, the outer x remains the same: 1

1. Python allow easy implementation of the iterator pattern since functions are first class objects. What are the two main parts of the pattern if we do not have first class functions?

The interface to place the functions, and setting up the main class to accept and call and concrete instance of the interface.

1. What does **yield** do in Python?

It pauses the function and optionally returns a value. When the function is called again, it restarts right after the yield.

1. How does an iterator aid in encapsulation? (1-2 sentences)

It protects the collection from direct access

|  |  |
| --- | --- |
|  |  |
| 1. Name one solid principle the iterator pattern aids in keeping, and why? (pulled from exam)   I is the most obvious. It is the primary reason for the iterator pattern  O is also easy claim as it stops people from having to open up the collection  L is another large one and it forces to treat sub element the same way | |
|  | |

1. **Select all** of the following that should use a strategy pattern rather than inheritance or the visitor pattern: (pulled from a past exam)

|  |  |
| --- | --- |
|  | A plant blooms only in a given season which can change with the hemisphere. It blooming method is dependent on the specific flower object. |
|  | All plants grow. |
|  | A car engine processes fuel a certain way, and this can depend on the vehicle and can change depending on how the vehicle is driven. |
|  | You have several dogs all with their own size |

The first and third options. Plants growing all have the same function and the function remains the same for all instances of the class, so inheritance works. Dog size should be an instance variable.

Blooming changes with the individual plant, so inheritance doesn’t work. Fuel process also changes with the individual vehicle, so inheritances doesn’t work.

1. What is Type Compatibility? (essentially pulled form exam)

A type rule that deals with which types a construct will accept or not

1. Explain how data types would be used/handled in a language that is both weakly typed and dynamically typed (2-3 sentences). Give an example variable declaration. (pulled from exam)

This is close to LISP which I’ve referenced in class. You would not need to explicitly declare the datatype since it would be determined by the value assigned to it. However, since it is weakly typed, this also means we can cast without explicit notation since datatypes are fluid. Since it is dynamic, it is almost guaranteed to have garbage collection as well.

Example usage:

x = 2  
x = 'c' +2  
will result in 'e' or the ACSII value of 'e'

1. Why do many modern languages include properties rather than only traditional member variables?

Ease of use. Adding a setter/getter is assumed in everything that has to be accessed outside the class, so this simplifies use.

1. Write the regular expression for 0 or more a’s followed by 1 or 2 bs.

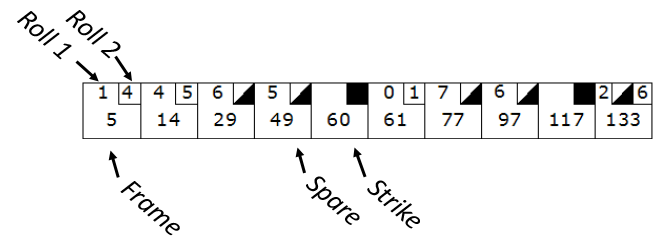
a\*b{1,2}

## [15pt] Draw the structure of the strategy pattern if you do not have first class functions like in Python (AKA class based). Include the core function calls.

1. Diagram the following problem

**Description**

Suppose you are writing a system to keep score for a game of bowling. Here is an example of a bowling score card:



There are 10 frames. In each frame the bowler rolls two balls. Each box represents one frame. The number of pins knocked down for a rolls is put on the top of the box, as shown above. The scores accumulate.

A strike is when the bowler knows down all of the pins with one roll. In that case there is no second roll. A spare is when the bowler knocks down all of the pins with the second roll.

*Here is how a frame is scored:*

If neither a strike or spare is rolled, the frame score is the previous frame score plus the number of pins knocked down in this frame. In frame 2 above, the score is the previous frame's 5 plus 4 plus 5 = 14.

If a spare is thrown, the number of pins in the next roll is added as a bonus to the frame score. In frame 3 a spare has been rolled. The score is 14 (previous frame) plus 10 (this frame) plus 5, the next roll = 29.

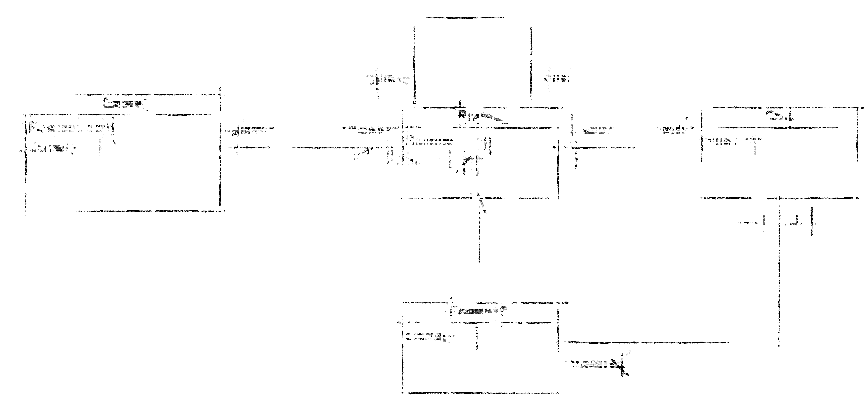
If a strike is thrown, the number of pins in the next two rolls is added as a bonus to the frame score. In frame 5 a strike this thrown. The score is 49 (previous frame) plus 10 (this frame) plus 0 (next roll) plus 1 (next roll after that) = 60.

The 10th frame is special in that you can roll one extra ball if you get a strike in the 10th frame.

*Required Functionality:*

You must be able to roll and retrieve the score.

Reset the picture to make it show

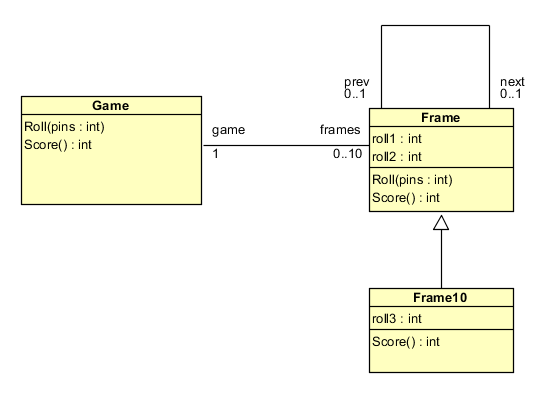


Explanation:

Note the prev/next links on the frame. If we do that, it is easy to compute the score.

Alternative

The 10th frame is special. That needs to be accomodated in some way. I took a "roll" as a thing and made a class for it. That works well since a frame may have 0 to 3 rolls. But, another reasonable approach would be to make the roll an attribute:



This is a reasonable solution, but it will be harder to tell a partially rolled frame. Do we set roll1 and roll2 to zero initially? Or maybe -1 to mean an invalid value? That's why I prefer the first version.

**Give an example of a good time to add *multiple* iterators to a problem, other than a serial *for-each* loop over a collection. In other words, something similar to “print all” level of complexity for more than one iterator is forbiddenas an answer.**