ENVIRONMENT DIAGRAMS, RECURSION, DATA ABSTRACTION

COMPUTER SCIENCE 61A

February 7 and February 9, 2017

1 Recursion

1.1 Questions

1. Implement the function nearest two, which takes a positive number x as input and returns the power of two $(\dots, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8, \dots)$

```
def nearest_two(x):
    """ Return the power of two that is nearest to x.
    >>> nearest_two(8)
    8.0
    >>> nearest_two(11.5) # closer to 8 than 16
    8.0
    >>> nearest_two(0.75) # tie between 0.5 and 1
    1.0
    """
```

   1. Write an iterative solution.

Solution:

```
    power_of_two = 1.0
    if x < 1:
        factor = 0.5
    else:
        factor = 2.0
    while abs(power_of_two * factor - x) < abs (
       power_of_two - x):
        power_of_two = power_of_two * factor
    if abs(power_of_two * 2 - x) == abs (power_of_two - x
       ):
        power_of_two = power_of_two * 2
    return power_of_two
```

2. Write a recursive solution.

Solution:

```
if x < 1:
        factor = 0.5
    else:
        factor = 2.0

    def helper(x, nearest_power, factor):
        if abs(nearest_power * factor - x) >= abs(
           nearest_power - x):
            return nearest_power
        return helper(x, nearest_power * factor, factor)

    result = helper(x, 1.0, factor)
    if abs(result * 2 - x) == abs(result - x):
        result = result * 2
    return result
```

2. Write a function that computes the *digital root* of a number n. The *digital root* is defined as a recursive summation of the digits of n until only one digit is left. Hint: you may find a separate function to sum the digits of a number useful.

```
def digital_root(n):
    """
    >>>digital_root(5789)
    2 #5+7+8+9 = 29; 2+9 = 11; 1+1 = 2
    >>>digital_root(37)
    1 #3+7 = 10; 1+0 = 1
    >>>digital_root(999888774)
    6 #9+9+9+8+8+8+7+7+4 = 69; 6+9 = 15; 1+5 = 6
    """
```

Solution:

```
    if summer(n) // 10 == 0:
        return summer(n)
    else:
        return digital_root(summer(n))

def summer(n):
    counter = 0
    while n > 0:
        counter += n % 10
        n = n // 10
    return counter
```

3. Alice has a crush on one of Bob's friends. However, she will only tell him who if he is able to correctly guess her secret number (an integer in the range from 0 to 100) in less than 8 guesses. Bob's genius friend Euler tells Bob that he can easily figure out Alice's secret number in less than 8 guesses as long as she is willing to tell him if his guess is equal to, lower than, or higher than her secret number. Bob asks if Alice is willing to give him this information and she agrees to tell him 0 if his guess is equal to the secret number, -1 if his guess is less than the secret number, and 1 if his guess is greater than the secret number. Help Bob figure out Alice's crush.

```
def binarysearch(low, high):
    """
    Alice gives Bob the function direction where direction(
       guess) returns either 0, -1, or 1 as specified in the
       problem statement.
    """
```

Solution:

```
        guess = (low + high) // 2
        sign = direction(guess)
        if sign == 0:
                return guess
        if sign < 0:
                return binarysearch(low, guess)
        if sign > 0:
                return binarysearch(guess, high)
```

2 Environment Diagrams

2.1 Questions

1. Draw an environment diagram for the following code.

```
hil = "hello "

def fin(ger):
    hil = "bye "
    return (lambda fin: lambda fin: ger(hil) + "world")(hil)

def ger(ger):
    return hil

hil = fin((lambda: ger)())(hil)
```

Solution: Python Tutor

3 Data Abstraction

Let’s practice data abstraction by implementing a cell phone! Here are a few functions weve written for you:
Constructor (creates an abstraction):

```
def phone(name, model, contacts):
        """Returns an abstraction for a cell phone. """
        return [name, model, contacts]
```

Selectors (gets data from the abstraction):

```
def get_name(phone):
        """Returns the name of the owner of the phone. """
        return phone[0]

def get_model(phone):
        """Returns the model of the phone. """
        return phone[1]

def get_contacts(phone):
        """Returns a list of contacts contained in the phone.
           """
        return phone[2]

def add_contact(phone, contact):
        """Adds a contact to the phone (must also be a phone).
            """
        get_contacts(phone).append(contact)
```

1. What Would Python Print?

```
>>> google = phone( Garfield , Google , [])
>>> lg = phone( Charles , LG , [google])
>>> get_name(google)
```

Solution: Garfield

```
>>> get_model(google)
```

Solution: Google

```
>>> get_contacts(google)
```

Solution: []

```
>>> google[0]  # Data abstraction barrier  Does it work?
```

Solution: Garfield

```
>>> add_contact(google, lg)
>>> get_name(get_contacts(google)[0])
```

Solution: Charles

```
>>> get_model(get_contacts(google)[0])
```

Solution: LG

2. Bob has mysteriously forgotten how to use his phone, and he has forgotten who is in his list of contacts! Write a function that returns true if and only if a contact is in his phone.

```
def contains_contact(phone, name):
    """Returns true if phone contains this person.
    >>> bob = phone("Bob", "Samsung", [])
    >>> contact1 = phone("Fate", "Sony", [])
    >>> add_contact(bob, contact1)
    >>> contains_contact(bob, "Fate")
    True
    >>> contains_contact(bob, "Hope")
    False
    """
```

Solution:

```
        for contact in get_contacts(phone):
                if get_name(contact) == name:
                                return True
        return False
```

3. Finally, lets write a function that gives us a description of our phone.

```
def print_info(phone):
    """Prints out information about the phone.
    >>> bob = phone("Bob", "Samsung", [])
    >>> print_info(bob)
    This is Bob's Samsung phone
    This phone contains contact information for nobody
    >>> contact1 = phone("Tutorial", "Sony", [])
    >>> add_contact(bob, contact1)
    >>> print_info(bob)
    This is Bob's Samsung phone
    This phone contains contact information for 1
    """
```

Solution:

```
print("This is " + get_name(phone) + "\'s " + get_make(
   phone) + " phone")
                if not get_contacts(phone):
                        print("This phone contains the
                           contact information for nobody")
                else:
                        print("This phone contains the
                           contact information for", len(
                           get_contacts(phone)))
```