

# 100 intro to computer science. final exam. 2010.

Total XP: 250

**HOW TO SUBMIT:** Because I am old and have a hard time reading the script of the young, please submit your answers typed (PDF preferred) to [submit.o.bot@gmail.com](mailto:submit.o.bot@gmail.com). (subject line: 110 exam submission)

## 1. average (25 XP)

I asked students to create a function that asked a user for a series of numbers and return the sum. One student submitted the following code:

```
def sum():
    num = input("enter a number: ")
    while num != '':
        total += num
        num = input("enter a number: ")
    return(total)
```

There are 2 errors that prevent the function from working. What are the errors and how would I fix the program?

## 2. formatDate (25 XP)

I asked a student to write a function that would format a date. That is, it would take a date like 05-06-51 and print out *May 6, 1951* or a date like 10-28-10 and print out *October 28, 2010*. Here is the code the student wrote:

```
def formatDate(aDate):
    """Takes a date in the form of 05-10-95 and prints May 10, 1995
    I need this to work for dates between 1950 and the present"""
    months = ['January', 'February', 'March', 'April', 'May', 'June',
              'July', 'August', 'September', 'October', 'November',
              'December']
    # take a date like 05-10-95 and split it to ['05', '10', '95']
    theDate = aDate.split('-')
    mo = int(theDate[0])
    month = months[mo]
    day = int(theDate[1])
    year = 1900 + int(theDate[2])
    print("%s %i, %i" % (month, day, year))
```

Unfortunately, this doesn't seem to work as it gives the wrong output for the two dates listed above. Describe the two errors and fix the function.

### 3. Factorial (25)

Recall that factorial is the result of multiplying a given number of consecutive integers from 1 to the given number. So  $3!$  is  $1 * 2 * 3 = 6$  and  $5!$  is  $1 * 2 * 3 * 4 * 5 = 120$ . I tried to write a function that computes factorial but it doesn't give me the correct answer.

```
def factorial(n):
    total = 0
    while n > 0:
        total = total + 1
        n -= 1
    return total
```

Please fix the errors.

### Dog (35XP)

Dogs can understand only certain words. For this example, consider they know the words:

*squirrel, walk, hungry, dog biscuit, ball*

If someone says one of these words, they hear that word fine. If someone says a different word, all they hear is *blah*.

So, for example, if someone says *Want to go for a walk and play ball?* a dog hears:

*Blah blah blah blah blah walk blah blah ball*

Finish the following dog function which translates English into dog:

```
def dog(sentence):
    words = ['squirrel', 'walk', 'hungry', 'dog biscuit', 'ball']
```

## Relations (50XP)

Consider the following family relations.

I, Ron, am married to Cheryl and we have a son, Adam.

I, Ron, have a sister named Lois who is married to Eric and they have a son Peter, and a daughter Paola.

My father's name is Syl and my mother's name is Evelyn.

Cheryl has a sister, Linda. Their father is Joseph and their mother is Madeline.

I have represented the above using Python dictionaries:

```
sex = {'Adam': 'm', 'Peter': 'm', 'Paola': 'f', 'Eric': 'm', 'Lois': 'f',
       'Ron': 'm', 'Cheryl': 'f', 'Linda': 'f', 'Evelyn': 'f', 'Syl': 'm',
       'Joseph': 'm', 'Madeline': 'f'}

fatherOf = {'Peter': 'Eric', 'Paola': 'Eric', 'Adam': 'Ron', 'Lois': 'Syl',
            'Ron': 'Syl', 'Cheryl': 'Joseph', 'Linda': 'Joseph'}

motherOf = {'Peter': 'Lois', 'Paola': 'Lois', 'Adam': 'Cheryl', 'Lois':
            'Evelyn', 'Ron': 'Evelyn', 'Cheryl': 'Madeline',
            'Linda': 'Madeline'}
```

I can find out who is Linda's father by

```
>>> fatherOf['Linda']
'Joseph'
```

Part 1: Implement grandfatherOf, which will return a person's grandfathers (20XP)

```
>>> grandfatherOf('Adam')
['Syl', 'Joseph']
```

Part 2: Implement sisterOf, which will return a person's sisters (30XP)

Part 3: Implement nephewOf, which will return a person's nephews (30XP)

## Acronym revisited (25XP)

I asked a student to create an acronym function that ignored the common words *the*, *of*, *and*, and *at*.

Here is an example of what I intended:

```
>>> acronym('University of Mary Washington')
'UMW'
>>> acronym('University of Wisconsin at Milwaukee')
'UWM'
>>> acronym('Massachusetts Institute of Technology')
'MIT'
```

A student submitted the following code. Please fix the errors:

```
def acronym(phrase):
    result = ''
    for word in phrase.split():
        if word.lower() in ['the', 'of', 'and', 'at']:
            result += word
    return(result)
```

## Destinations (30xp)

I've recorded the current costs of flights from Dulles to various spots using a dictionary:

```
destinations = {'Gudalajara': 660, 'Honolulu': 470, 'Brazil': 770,
                'Dublin': 530, 'Paris': 680, 'Norway': 750, 'Edinburgh': 660}
```

I would like you to write a function, `whereCanIfly` that can tell me what destinations I can fly to with a given dollar amount:

```
>>> whereCanIfly(700, destinations)
Edinburgh
Gudalajara
Paris
Dublin
Honolulu
>>> whereCanIfly(500, destinations)
Honolulu
```