| 1<br>( 20 ) | 2<br>( 20 ) | 3<br>( 20 ) | 4<br>( 20 ) | 5<br>( 20 ) | TOTAL<br>(100) |
|---|---|---|---|---|---|
| | | | | | |

# [ CS101 ] Introduction to Programming
# 2016 Fall - Final Examination

| SECTION | STUDENT ID | NAME |
|---|---|---|
| | | |

※ Please check if you have all 34 pages of the test material.
※ 시작하기 전에 반드시 페이지의 수를 확인 하십시오.(전체 : 34쪽)

※ Fill in your student identification number and name. Otherwise you will lose 1 point for each missing piece of information.
※ 위의 정보(학번,이름)를 정확히 기입하지 않을 경우, 각 실수 당 1점이 감점 됩니다.

※ **TAs will not answer your questions about the exam**. If you think there is anything ambiguous, unclear or wrong about a problem, please write down the reasons and make necessary assumptions to solve the problem. We will take your explanation into consideration while grading.
※ **시험시간동안 질문을 받지 않습니다**. 만일 문제에 오류나 문제가 있을 경우, 왜 문제가 이상이 있다고 생각하는지에 대해서 기술하시면 되겠습니다. 또한 문제가 애매하다고 생각되는 경우 문제를 푸실 때 본인이 생각하는 가정을 함께 작성하셔서 문제를 푸시면 되겠습니다. 채점 시 가정 및 설명을 고려하도록 하겠습니다.

※ **Use Python 3 only**. All your answers will be graded based on Python 3.
※ **파이썬 3만 사용하십시오**. 채점은 파이썬 3 기준으로만 합니다.

**1. (20 points) Please answer each of the following questions according to the instruction.**

**1-1. (8 Points)** Fill out the Answer field with "**T**" if the answer is **true** or **"F" otherwise.** You will get one point for each correct answer.

| Statement | Answer |
|---|---|
| *(Example) The first character of a name should not be a digit.* | **T** |
| Computational thinking is a way of human thinking for solving problems with a computer. | |
| The following code will produce a syntax error:<br>`x = 0`<br>`y = 34.5 / x` | |
| A floating point number object is mutable. | |
| A parameter of a function is a local variable. | |
| A value that is specified in a function call is a parameter. | |
| The method to move a graphical object to a relative coordinates on a canvas is 'moveTo()'. | |
| The result of the following comparison is 'True'.<br>`3 == 3.0` | |
| `__class__` is the constructor method that is called when an object is created. | |

**1-2. (4 points/1 point each)** Please consider the following statements, and answer whether each of the statements in the table will result "True" or "False". Write "True" or "False in the Answer field.

```
a = [ 'c', 's', '1', '0', '1' ]
b = [ 'c', 's', '1', '0', '1' ]
c = a
d = a[ : ]
```

| Statement | Answer |
|---|---|
| `a == b` | |
| `a is b` | |
| `a is c` | |
| `a is d` | |

**1-3 (2 points)** Which graphical object will be shown in foreground on the canvas?

```
rec = Rectangle(150, 75)
canvas.add(rec)
sqr = Square(100)
canvas.add(sqr)
rec.setDepth(50)
sqr.setDepth(50)

(1) Rectangle     (2) Square     (3) Canvas     (4) Circle
```

_____ (2 points)

**1-4 (2 points)** What happens if you don't call the close() method after writing data to a file?

```
(1) The file will not be created.
(2) The file will be deleted.
(3) The file contents may be incomplete.
(4) The file name will be changed.
```

_____ (2 points)

**1-5 (2 points)** Why is the merge sort algorithm is faster than the straight sort algorithm?

```
(1) Because the merge sort can be implemented as a recursive function.
(2) Because the merge sort requires a less number of comparisons
    between elements.
(3) Because the size of the merge sort program is smaller
(4) Because the complexity of the merge sort is (n-1)^2.
```

_____ (2 points)

**1-6. (2 points)** Please write the string value that will be assigned to the variable, c, as the result of running the following code.

```
a = "cs 101"
b = ["*", "#", "$"]
c = a.join(b)
```

_____ (2 points)

**2. (20 points)** In this problem, we use the Python class and object concepts to simulate the animal world. Read the problem description and code below and answer the questions.

**2-1. (6 points)** The following is the content of a file that stores the sounds of animals as text. The storage format is as follows. 'Animal_name \t sound_1, sound_2 ...' There can be one or more sounds stored for each animal. The following is an example of storing the sounds of Lions :

Example)

| L | i | o | n | s | \t | r | o | a | r | , |  | g | r | o | w | l | \n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

animal_sounds.txt

```
Apes   gibber
Birds  chirrup, chirp, twitter, tweet, sing, whistle
Cats   mew, purr, meow, hiss, yowl
Dogs   bark
Ducks  quack
Giraffes  bleat
Lions  roar, growl
Mosquitoes    whine
Owls   hoot, scream, screech, shriek
Tigers growl, roar
Vultures  scream
Wolves howl, cry, yell
```

Please consider the `sound` function below, and answer the following questions.

q2-1.py

```
def sounds(name):
    f = open('animal_sounds.txt')
    for l in f.readlines():
        if name.split('#')[0] in l:
            word = l.strip().split('\t')[1].split(', ')
            break
    else:
        return ['niconiconi']
    return word

print(sounds('Tiger#12'))
print(sounds('Nico#12'))
print(type(sounds('Duck#3')))
```

※ **Hint** : A loop statement can have an ***else*** clause, which is executed when the loop terminates by reaching to the end of a list. However, the else clause will not be executed when the loop is terminated by running a break statement.

Please write the screen output of running the `q2-1.py` program.

**2-2. (5 points)** The following python code is for creating an object by using the user-defined class `Animal`, and printing the attributes of the object. The `sounds` function used in this code is the function that is defined in Question 2-1.

q2-2.py

```
import random
MSG_FORM =  '%-10s: %s'

class Animal(object):
    def __init__(self, name, level=1, life=20):
        self.name, self.level, self.life = name, level, life

    def hunting(self, food):
        print(MSG_FORM%(self.name, 'Start hunting '+food.name))
        if food.level < self.level:
            self.life = self.life - food.life
            food.life = 0
            print(MSG_FORM%(self.name,
                            random.choice(sounds(self.name))))
        else:
            print(MSG_FORM%(self.name, 'Hunting failure'))

    def died(self):
        if self.life == 0:
            print(MSG_FORM%(self.name, 'Rest in peace'))
            return True
        else:
            return False

a = Animal('Duck')
print(a.name, a.level, a.life)
```

※ **`random.choice`**`(list)` : Returns a random element from a non-empty list.

Please write the screen output of running the `q2-2.py` program.

What is the number of constructors and methods of the `Animal` class?

**2-3. (9 points)** The code below is for simulating an animal world that is composed of `Animal` objects. Please also check the screen output of the program. The `Animal` objects are created by using the `Animal` class that is defined in Question 2-2.

q2−3.py

```
# Initialize the animal world
animal_world = []

for i in range(50):
    animal_world.append(Animal('Duck#%-3d'%i, 1, 50))

for i in range(20):
    animal_world.append(Animal('Dog#%-3d'%i, 3, 80))

for i in range(5):
    animal_world.append(Animal('Tiger#%-3d'%i, 5, 100))

# Animal world simulation
for i in range(10):
    print('-'*40)
    random.shuffle(animal_world)
    animal1 = animal_world.pop()
    animal2 = animal_world.pop()
    animal1.hunting(animal2)

    if not animal1.died():
        animal_world.append(animal1)

    if not animal2.died():
        animal_world.append(animal2)
```

<Screen output of q2-3.py>

```
----------------------------------------
Tiger#0   : Start hunting Duck#32
```

**2-3-1**

```
Duck#32   : Rest in peace
----------------------------------------
Duck#23   : Start hunting Tiger#0
Duck#23   : Hunting failure
----------------------------------------
Duck#47   : Start hunting Duck#41
Duck#47   : Hunting failure
----------------------------------------
Dog#10    : Start hunting Duck#19
```

**2-3-2**

```
Duck#19   : Rest in peace
----------------------------------------
Duck#49   : Start hunting Duck#39
Duck#49   : Hunting failure
----------------------------------------
Duck#39   : Start hunting Duck#47
Duck#39   : Hunting failure
----------------------------------------
Dog#4     : Start hunting Duck#30
```

**2-3-3**

```
Duck#30   : Rest in peace
----------------------------------------
Tiger#1   : Start hunting Duck#16
```

**2-3-4**

```
Duck#16   : Rest in peace
----------------------------------------
Duck#18   : Start hunting Duck#38
Duck#18   : Hunting failure
----------------------------------------
Duck#33   : Start hunting Duck#42
Duck#33   : Hunting failure
```

Please fill in the following blanks with appropriate messages to be shown on the screen.

| | |
|---|---|
| **2-3-1** | |
| **2-3-2** | |
| **2-3-3** | |
| **2-3-4** | |

What is the number of remaining animals, that is, the number of `Animal` objects in the `animal_world` list?

| |
|---|
| |

**3. (20 points)** Please answer each of the following questions according to the instruction.

**3-1. (7 points)** What are the results of running the following program?

```
def Avg(data, start = 0, end = None):
   if not end:
      end = len(data)
      return sum(data[start:end]) / float(end-start)

def Angel(data):
   data2 = [1, 0, 0]
   data2 = data
   data.append(4)
   return data

#main function1
list = [1, 3, 8, 11]
list_slice = list[-2: ]
print (list_slice)                #(3-1-1)
print ("%d" % Avg(list))          #(3-1-2)
print ("%.2f" % Avg(list, 2))     #(3-1-3)
list = Angel(list)
print (list)                      #(3-1-4)
```

| **Answer 3-1-1 (1 points)** |
| --- |
| |

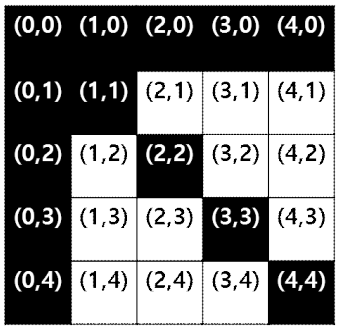
| **Answer 3-1-2 (2 points)** |
| --- |
| |

| **Answer 3-1-3 (2 points)** |
| --- |
| |

| **Answer 3-1-4 (2 points)** |
| --- |
| |

**3-2. (5 points)** An image object named `'img1'` is composed of black and white pixels as follows.

img1




**3-2-1.** The following program is to create a new image based on the above image. Please show the resulting image `'img2'` by shading the black pixels of the image.

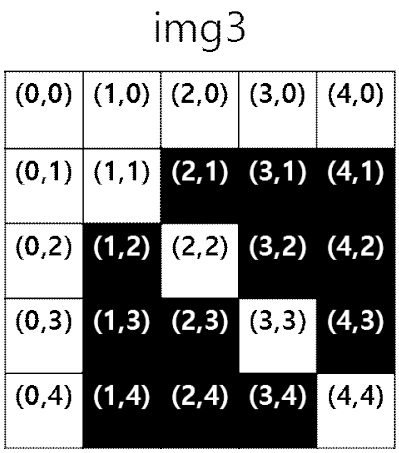
```
black = (0,0,0)
white = (255,255,255)

width, height = img1.size()
img2 = create_picture(width,height)

for y in range(height):
   for x in range(width/2):
      temp = img1.get(width-x-1, y)
      img2.set(width-x-1, y, img1.get(x,y))
      img2.set(x, y, temp)
img2.show()
```

**Answer 3-2-1 (2 points)**

img2

| | | | | |
|---|---|---|---|---|
| (0,0) | (1,0) | (2,0) | (3,0) | (4,0) |
| (0,1) | (1,1) | (2,1) | (3,1) | (4,1) |
| (0,2) | (1,2) | (2,2) | (3,2) | (4,2) |
| (0,3) | (1,3) | (2,3) | (3,3) | (4,3) |
| (0,4) | (1,4) | (2,4) | (3,4) | (4,4) |

**3-2-2.** Please complete the code below to draw the following image `'img3'`.
※ Hint: `img3` is a negative image of `'img1'`




```
black = (0,0,0)
white = (255,255,255)

width, height = img1.size()
img3= create_picture(width,height)

for y in range(height):
   for x in range(width/2):
```

**3-2-2**

```
img3.show()
```

**Answer 3-2-2 (3 points)**

**3-3. (8 points)** What is the result of the following program?

```
def func1(var2, var3):
   var2 += 2
   var3 = 0

def func2(var):
   global var2, var4
   var = var2
   var4 *= 2

def func3(var1, var2):
   sequence = []
   sequence.append(var1)
   sequence.append(var2)
   i = len(sequence)
   while sequence[i-1] < 10 :
      sequence.append(sequence[i-2] + sequence[i-1])
      i = len(sequence)
   sequence.pop()
   return sequence

var1, var2, var3, var4 = 0, 1, 1, 3
func1(var2, var3)
print (var1, var2, var3, var4)          #(3-3-1)

var1, var2, var3, var4 = 0, 1, 1, 3
var1 = func2(var3)
print (var1, var2, var3, var4)          #(3-3-2)

var1, var2, var3, var4 = 0, 1, 1, 3
var1 = func3(var1, var2)
print (var1[3])                         #(3-3-3)

var7 = str(7)
print (var7 is '7')                     #(3-3-4)
```

| Answer 3-3-1 (2 points) |
| --- |
| |

| Answer 3-3-2 (2 points) |
| --- |
| |

| Answer 3-3-3 (2 points) |
| --- |
| |

| Answer 3-3-4 (2 points) |
| --- |
| |

**4. (20 points)** Now that you have taken CS101 and feel comfortable with programming in Python, you decide to take your passion in programming further. While you're looking through Amazon.com for a good python programming book, you realize the books have inconsistent, inflated reviews! Therefore, you decide to make a Python program to analyze the book reviews by using the text processing skills that you have learned in this semester, and generate your own review score on the books.

The new scoring mechanism is very simple. The file "reviews.txt" contains all 15 reviews of the book "Practical Programming: An Introduction to Computer Science Using Python 3." (Some snippets of the reviews are shown on the next page) Each review is composed of six lines, each of which is about rating, summary, date of purchase, book type, review content, end of review respectively and a blank line follows to divide one review from another. **You will count the number of positive words and negative words in each the review content section of each review, subtract number of negative words from number of positive words to get the re-scaled new score.**

Positive words are stored in "poswords.txt" file and the negative words are stored in "negwords.txt" file. They are both shown below.

**poswords.txt**

```
good\n
well\n
tremendous\n
great\n
awesome\n
superbly\n
amazing\n
excellent\n
recommended\n
helpful\n
```

**negwords.txt**

```
bad\n
horrible\n
difficult\n
hard\n
```

**reviews.txt**

```
5.0 out of 5 stars\n
Used this book with online course in Python 3.0\n
November 21, 2013\n
Paperback\n
Lives up to what is promised. Want to learn Python 3.0 and this book teaches the basics superbly. You can use the book with the online Coursera class thru the University of Toronto. The course is rapid pace and the book is a great help.\n
End of review\n

4.0 out of 5 stars \n
A very good book that helps to be a good programmer using Python\n
March 27, 2014\n
Paperback\n
This is a very good book well suited to support a programming course that uses Python as a language base. This book is not limited to teaching a programming language like Python, but provides guidance on how to address and solve problems, which is the most important task you need to know to be a good programmer.\n
End of review\n

5.0 out of 5 stars\n
The book is real great for people who really don't know anything\n
January 30, 2015\n
Paperback\n
The book is well written and the concepts are well explaine'd. I just hope it went a little deeper. I am not beginner but not intermediate either. The book is real great for people who really don't know anything. But if you have some exposure or experience of programming then it lacks a little bit of content. However, it is one of the best CS books that I have read ^^ Also, it is a good book to supplement with other python courses or materials.\n
End of review\n

5.0 out of 5 stars\n
Five Stars\n
January 29, 2015\n
Format: Paperback\n
no problems\n
End of review\n

5.0 out of 5 stars\n
Five Stars\n
March 31, 2015\n
Format: Paperback\n
Very helpful\n
End of review\n

3.0 out of 5 stars\n
required textbook for college course\n
October 29, 2013\n
Format: Paperback\n
It is sort of difficult to read, but computer programming is not simple. The program itself is wonderful, go python :)\n
End of review\n

...
```

**Note** "..." in the last line means more contents are in the file, but are not shown. Ignore them when processing files in your code.

**4-1. (4 points)** First, let's implement the following function read_words. Its main purpose is to read positive words and negative words from the files. It takes two arguments, the name of the file to read the words from, and the name of the list to store the words in without the new line character. Please complete the function by considering the demonstrated usage example shown below.

[Program Code 4 – 1]

```
def read_words(file_name, list_name):
    #read the file
    f = 4-1-1
    for line in f:
        #add word to the list without the new line character
        4-1-2
    f.close()
```

[Usage Example 4 – 1: Python Terminal]

```
>> negword_list = []
>> read_words("badwords.txt", negword_list)
>> print negword_list
['bad', 'horrible', 'difficult', 'hard']
```

**Your Answers:**

**4-1-1 (2pts)**

**4-1-2 (2pts)**

**4-2. (4 points)** Please complete the function how_many_reviews, which counts the number of reviews in the given file. Remember there is "End of review" line at the end of each review, we will exploit this fact and simply count the number of occurrences of "End of review" in the given file.

[Program Code 4 - 2]

```
def how_many_reviews(file_name):
    cnt = 0
    f = 4-2-1
    for line in f:
        if line.lower().4-2-2 == "end of review":
            4-2-3
    f.close()
    return cnt
```

[Usage Example 4 - 2: Python Terminal]

```
>> read_words("reviews.txt")
15
```

**Your Answers:**

**4-2-1. (2 points)**

**4-2-2. (1 point)**

**4-2-3. (1 point)**

**4-3. (8 points)** Please implement the new_scores function, which prints the new score. New scores are calculated by subtracting the number negative words from the number of positive words for each review. The function takes four arguments, the name of the file containing the reviews, total number of reviews in the file, a list that contains positive word and a list that contains negative words. The function prints only non-zero scores after the calculation. (Hint. All reviews have the same structure! Examine "reviews.txt" carefully.)

[Program Code 4 – 3]

```
def new_scores(file_name, num_of_reviews, goodwords, badwords):
    f = open(file_name, 'r') #1pts

    for i in range(num_of_reviews):
        pos_count = 0
        neg_count = 0
        rating = summary = dop = booktype = review = endofreview = ""

        [          4-3-1          ]

        for word in review. [ 4-3-2 ] :
            if [          4-3-3          ] :
                pos_count += 1
            if [          4-3-4          ] :
                neg_count += 1

        sum = pos_count – neg_count
        if sum != 0:
            print sum
```

**Your Answers:**

**4-3-1. (3 points)**

**4-3-2. (1 point)**

**4-3-3. (2 points)**

**4-3-4. (2 points)**

**4-4. (4 points)** Now, let's put everything together in the main function, which first asks the user for the name of the file that contains the positive words, then asks the name of the file that contains the negative words. Then the function asks the user for the name of the file that contains the reviews and will print the (non-zero) new scores. Use the functions you completed in the questions above.

[Program Code 4 – 4]

```
def main():
    poswords=[]
    negwords=[]

   # ask users for the file names and
   # store them in the lists declared above
```

**4-4-1**

```
    num_of_reviews = 0
    reviews_filename = ""
    # ask user for the review file name
    # calculate number of reviews and store it in num_of_reviews
    # store the file name in reviews_filename
```

**4-4-2**

```
    new_scores(reviews_filename, num_of_reviews, poswords, negwords)
```

[Usage Example 4 – 4: Python Terminal]

```
>> What's the name of the good words file? poswords.txt
>> What's the name of the bad words file? negwords.txt
>> What's the name of the reviews file? reviews.txt
1
3
4
1
3
-1
1
1
-1
```

**Your Answers:**

**4-4-1. (2 points)**

**4-4-2. (2 points)**

**5. (20 points)** Please read the following instructions carefully and answer the questions according to the instructions.

**5-1. (12 points)** Fill in the blanks to overcome the limitations of the `one_digit_ocr` function of Homework 3-1, and compute the accuracy of matching images. Answers can be one or multiple lines.

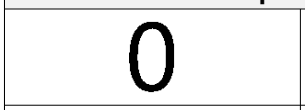
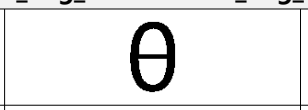
**Python Code 5-1**

```
from cs1media import *

error_threshold = 200
reference_img_list = ['0.bmp', '1.bmp', '2.bmp', '3.bmp', '4.bmp',
'5.bmp', '6.bmp', '7.bmp', '8.bmp', '9.bmp']
reference_img_path = "./reference_img/"
test_img_list = [('test_0.bmp',0),('test_1.bmp',1),('test_2.bmp',2),
        ('test_3.bmp', 3), ('test_4.bmp', 4), ('test_5.bmp', 5),
        ('test_6.bmp', 6), ('test_7.bmp', 7), ('test_8.bmp', 8),
        ('test_9.bmp', 9), ('test_10.bmp', 0), ('test_11.bmp', 7)]
test_img_path = "./test_img/"

def one_digit_ocr(img_path):
  img = load_picture(img_path)
  ret = -1
  w, h = img.size()
  for ref_img_filename in reference_img_list:
    ref_img = load_picture(reference_img_path + ref_img_filename)
    is_matched = True
    for y in range(h):
      for x in range(w):
        if img.get(x, y) != ref_img.get(x, y):
          is_matched = False
    if is_matched:
      ret = int(ref_img_filename[0])
  return ret

def one_digit_ocr_error(img_path):
  ret = -1
  img = load_picture(img_path)
  w, h = img.size()
  for ref_img_filename in reference_img_list:
    ref_img = load_picture(reference_img_path + ref_img_filename)
    errors = 0
```

```
    for y in range(h):
      for x in range(w):
          5-1-1
      5-1-2
  return ret

def one_digit_ocr_min_error(img_path):
  ret = -1
  img = load_picture(img_path)
  w, h = img.size()
  min_error = w * h + 1
  for ref_img_filename in reference_img_list:
    ref_img = load_picture(reference_img_path + ref_img_filename)
    errors = 0
    for y in range(h):
      for x in range(w):
          5-1-1
      5-1-3
  return ret

def accuracy(func_num):
    5-1-4

ocrr = one_digit_ocr(test_img_path + test_img_list[1][0])
print("The test_img_list[1] by one_digit_ocr is %d" % ocrr)
ocrr = one_digit_ocr(test_img_path + test_img_list[11][0])
print("The test_img_list[11] by one_digit_ocr is %d" % ocrr)
ocrr = one_digit_ocr_error(test_img_path + test_img_list[11][0])
print("The test_img_list[11] by one_digit_ocr_error is %d" % ocrr)
ocrr = one_digit_ocr_min_error(test_img_path + test_img_list[11][0])
print("The test_img_list[11] by one_digit_ocr_min_error is %d" % ocrr)
print("Accuracy of one_digit_ocr is %f" % accuracy(1))
print("Accuracy of one_digit_ocr_error is %f" % accuracy(2))
print("Accuracy of one_digit_ocr_min_error is %f" % accuracy(3))
```

| Output of the Python Code 5-1 |
|---|
| The test_img_list[1] by one_digit_ocr is 1 |
| The test_img_list[11] by one_digit_ocr is -1 |
| The test_img_list[11] by one_digit_ocr_error is 7 |
| The test_img_list[11] by one_digit_ocr_min_error is 7 |
| Accuracy of one_digit_ocr is 0.833333 |
| Accuracy of one_digit_ocr_error is 1.000000 |
| Accuracy of one_digit_ocr_min_error is 1.000000 |

The `Python Code 5-1` implements a simple solution for the `one_digit_ocr` function. It compares all pixels of the `reference_img_list` images against the `img_path` image file. If one of the reference images is the same as the `img_path` file, it returns only the file name of the reference image. This function passes all test images that are used for Homework 3-1. The test images are maintained in `test_img_list`, and nine images from `test_0.bmp` to `test_9.bmp` in the list are same files in Homework 3.

| Sample images in the reference_img_list and test_img_list lists | | | |
|---|---|---|---|
|  |  |  |  |
| 0.bmp | 7.bmp | test_10.bmp | test_11.bmp |

However, if the shape of a digit is changed, the function cannot recognize the digit correctly. The above table shows two reference images in `reference_img_list`, and two test images in `test_img_list`. A human can recognize the images in `test_10.bmp` and `test_11.bmp` as the numbers zero and seven respectively. However, the `one_digit_ocr` function returns –1 for these images.

To solve this problem, we can relax the pixel comparison test with using a threshold. First, we set the threshold of pixel errors, and when we compare two reference and test images, we count the number of pixel errors, which mean unmatched pixels between the two images. If the number of pixel errors is less than the threshold, we regard the test image represents the same number as the reference image. For example, the numbers represented in the `test_11.bmp` and `7.bmp` images can be recognized as the same number if the number of pixel differences between the two images is less than a pre-defined threshold.

Please implement the above solution by completing the `one_digit_ocr_error` function. The function accepts a test-image path, and returns a recognized integer number. The error threshold value is set as 200, and defined as a global variable. You should use this variable.

| Answer 5-1-1 (2 points) |
| --- |
| |

| Answer 5-1-2 (2 points) |
| --- |
| |

One of the problems of the `one_digit_ocr_error` function is that you should set the error threshold ahead. It is hard to pre-define the threshold value when there are various test images. To overcome this, we can find reference images that cause the minimum number of errors. The error-checking process is similar to the `one_digit_ocr_error` function. We count the number of pixel differences between a test image and all reference images, and find a reference image that causes the least number of errors. For example, for the `test_11.bmp` image above, the reference image, `7.bmp`, causes the smallest pixel errors than other reference images.

Please implement this solution by completing the `one_digit_ocr_min_error` function. The function accepts a test-image path, and returns a recognized integer number.

| Answer 5-1-3 (2 points) |
| --- |
| |

Now, you need to implement the `accuracy` function to compute the accuracy of the algorithms. The accuracy of an algorithm is calculated by using the following formula:

$$Accuracy\ of\ an\ algorithm = \frac{the\ number\ of\ correct\ answers\ by\ the\ algorithm}{the\ number\ of\ test\ data}$$

Each element of `test_img_list` is a tuple that is composed of the file name of a test image and the integer number that corresponds to the image. For instance, the last element of the list indicates that the digit in the `test_11.bmp` is the number 7. The size of `test_img_list` is not fixed.

We now have three functions that implements the algorithms. To identify the algorithm used for an error-checking function within the `accuracy` function, we use the parameter, `func_num` as follows:

- If `func_num` is 1: Compute the accuracy of the `one_digit_ocr` function
- If `func_num` is 2: Compute the accuracy of the `one_digit_ocr_error` function
- If `func_num` is 3: Compute the accuracy of the `one_digit_ocr_min_error` function
- Others: Return zero

`Output of the Python Code 5-1` shows the results of the `accuracy` function. The return value of `accuracy(1)` is 0.833333, because it is about checking errors in the `one_digit_ocr` function, and only 10 out of 12 test images are correctly detected and returned. Other two functions show 100% accuracy.

**Answer 5-1-4 (6 points)**

```
def accuracy(func_num):
```

**5-2. (8 points)** Homework 4 is about making a monopoly game. In the homework, the number of players is fixed as two. Here, we want to add more players. You should revise the code in `Python Code 5-2`, which is a solution of the homework. More descriptions about the problem and the answer sheet are located below the `Python Code 5-2`.

**Python Code 5-2**

```
import random


def main():
  courses = read_course()
  p1 = Player(_name="KSW", _cash=10 ** 4)
  p2 = Player(_name="LJS", _cash=10 ** 5)
  board = Board(_courses=courses, _players=[p1, p2])
  for p in board.player_list:
    print("[%s] has %d Won." % (p, p.cash))
  separator = "-" * 5
```

```
  while not board.finished:
    print(separator)
    board.next_turn()

def read_course():
  (...)  # Hidden code

def random_integer(low, high):
  (...)  # Hidden code

def random_choice(lst):
  (...)  # Hidden code

def ask_yes_or_no(prompt):
  (...)  # Hidden code

class Dice:
  def __init__(self, _face):
    self.face = _face

  def throw(self):
    first = random_integer(1, self.face)
    second = random_integer(1, self.face)
    return first == second, first+second



class Space:
  def __init__(self, _name):
    self.name = _name
    self.owner = None
    self.count = 0

  def __str__(self):
    return self.get_name()
  def get_name(self):
    return str(self.name)

  def set_owner(self, _owner):
    self.owner = _owner
    self.count += 1
```

```
    print("Now [%s] owns %s." % (self.get_owner(), self))

  def get_owner(self):
    return self.owner


class Course(Space):
  def __init__(self, _name, _code, _price):
    super().__init__(_name)
    self.code = _code
    self.price = _price

  def __str__(self):
    return "[%s] %s" % (self.get_code(), self.get_name())

  def get_code(self):
    return str(self.code)

  def get_price(self):
    return int(self.price)

class SpecialActivity(Space):
  def __init__(self, _name):
    super().__init__(_name)
    self.owner = Player(_name="NPC", _cash=0)

class Player:
  def __init__(self, _name, _cash):
    self.position = 0
    self.property = []
    self.name = _name
    self.cash = _cash
    self.is_bankrupt = False
    self.is_active = True

  def __str__(self):
    return self.name
  def bankruptcy(self):
    self.is_bankrupt = True
    print("[%s] goes bankrupt." % self.name)
```

```
  def take(self, _course):
    if self.pay(_course.get_price()):
      self.property.append(_course)
      _course.set_owner(self)

  def retake(self, _course):
    print("[%s] tries to retake %s." % (self.name, _course))
    if not self.pay(_course.get_price()):
      self.bankruptcy()

  def give(self, _course, other):
    print("[%s] must take %s(2*price)." % (other, _course))
    if other.pay(_course.get_price() * 2):
      self.earn(_course.get_price() // 2)
      _course.set_owner(other)
      self.property.remove(_course)
      other.property.append(_course)
    else:
      other.bankruptcy()

  def pay(self, money):
    if self.cash < money:
      print("Not enough money.")
      return False
    self.cash -= money
    return True

  def earn(self, money):
    self.cash += money
    print("[%s] earns %d Won." % (self.name, money))

  def activate(self):
    print("[%s] returns to school." % self.name)
    self.is_active = True


  def deactivate(self):
    print("[%s] takes leave of absence." % self.name)
    self.is_active = False
```

```
class Board:
  def __init__(self, _courses, _players):
    self.special_list = [SpecialActivity("Start"),
              SpecialActivity("Leave of Absence"),
              SpecialActivity("Scholarship"),
              SpecialActivity("Course Retaking")]
    self.course_list = _courses
    length = len(_courses) // 4
    self.road = []

    for idx, item in enumerate(self.special_list):
      assert isinstance(item, Space)
      self.road.append(item)
      for i in range(idx * 4, length+idx * 4):
        assert isinstance(self.course_list[i], Space)
        self.road.append(self.course_list[i])
    self.iter_idx = -1

    for p in _players:
      p.position = 0
    self.player_list = _players[:]
    self.dice = Dice(6)
    self.finished = False

  def next_turn(self):
    self.iter_idx += 1
    if self.iter_idx >= len(self.player_list):
      self.iter_idx = 0
    p = self.player_list[self.iter_idx]
    print("[%s]'s turn!" % p)
    if not p.is_active:
      p.activate()
      return
    is_double = True

    while is_double and not self.finished:
      is_double, move = self.dice.throw()
      if is_double:
```

```
        print("Double!")
      print("[%s] moves %d." % (p, move))
      p.position += move
      while p.position >= len(self.road):
        p.position -= len(self.road)
      space = self.road[p.position]

      if space.owner is None:
        print("[%s] has %d Won." % (p, p.cash))
        print("The price of %s is %d." % (space, space.price))
        if ask_yes_or_no("Wanna take %s? (y/n) " % space):
          p.take(space)
        else:
          print("[%s] chooses 'Drop this course'" % p)

      elif not space.get_owner().name == "NPC":
        print("This space is owned by [%s]."
              % space.get_owner().name)
        if space.get_owner().name != p.name:
          print("The price of %s is %d." % (space, space.price))
          space.get_owner().give(space, p)

      else:
        if space.get_name() == "Start":
          p.earn(97)
        elif space.get_name() == "Leave of Absence":
          p.deactivate()
          break
        elif space.get_name() == "Scholarship":
          p.earn(4019)
        elif space.get_name() == "Course Retaking":
          if p.property:
            p.retake(random_choice(p.property))
          else:
            print("[%s] has no course." % p)

      self.check_finish()
      if self.finished:
        return
```

```
    def rank(self):
      def key_course_price(course):
        return course.price
      def key_course_count(course):
        return course.count

      self.course_list.sort(key=key_course_price, reverse=True)
      self.course_list.sort(key=key_course_count, reverse=True)
      print("The most popular course was %s." % self.course_list[0])

    def check_finish(self):
      for player in self.player_list:
        if player.is_bankrupt:
          self.rank()
          self.finished = True
          print("[%s] loses the game." % player)
          return

  main()
```

The goal here is to add more players to the monopoly game implemented in `Python Code 5-2`. For simplicity, we considered only two players (KSW and LJS) in Homework 4. But now, we have four players for the monopoly game, and their name and initial amount of cash that they have are as follows:

- 1st player name: KSW, Initial cash = 10000
- 2nd player name: LJS, Initial cash = 100000
- 3rd player name: SWL, Initial cash = 200000
- 4th player name: JYB, Initial cash = 10849

The following table shows a sample output of the monopoly game that is revised for the four players.

**Output of revised Python Code 5-2**

```
[KSW] has 10000 Won.
[LJS] has 100000 Won.
[SWL] has 200000 Won.
[JYB] has 10849 Won.
-----
[KSW]'s turn!
[KSW] moves 5.
Wanna take [EE209] Programming Structure for Electrical Engineering? (y/n) y
Now [SWL] owns [EE209] Programming Structure for Electrical Engineering.
-----
```

```
[KSW] takes leave of absence.
-----
[LJS]'s turn!
[LJS] moves 9.
[LJS] has 100000 Won.
The price of [HSS378] Violin Family Instrument Making & Experimentation is 911.
Wanna take [HSS378] Violin Family Instrument Making & Experimentation? (y/n) y
Now [LJS] owns [HSS378] Violin Family Instrument Making & Experimentation.
-----
[SWL]'s turn!
Double!
[SWL] moves 2.
[SWL] has 200000 Won.
The price of [EE105] Present and Future of Electronics is 23.
Wanna take [EE105] Present and Future of Electronics? (y/n) y
Now [SWL] owns [EE105] Present and Future of Electronics.
[SWL] moves 9.
[SWL] has 199977 Won.
The price of [EE209] Programming Structure for Electrical Engineering is 1009.
```

**Page 1**

```
[JYB]'s turn!
[JYB] moves 11.
This space is owned by [SWL].
The price of [EE209] Programming Structure for Electrical Engineering is 1009.
[JYB] must take [EE209] Programming Structure for Electrical Engineering(2*price).
[SWL] earns 504 Won.
Now [JYB] owns [EE209] Programming Structure for Electrical Engineering.
(...)  # Hide the output
-----
[JYB]'s turn!
[JYB] moves 6.
This space is owned by [LJS].
The price of [CS408] Computer Science Project is 32413.
[JYB] must take [CS408] Computer Science Project(2*price).
Not enough money.
[JYB] goes bankrupt.
The most popular course was [HSS378] Violin Family Instrument Making & Experimentation.
[JYB] loses the game.
```

**Page 2**

Please modify `Python Code 5-2` to turn it into a monopoly game for the four players. Like the sample answer shown below, please indicate the location of each change that you made, and show how you changed the code by showing the code 'before and after' using '==>'. The following sample answer describes addition, deletion and change of the code.

**Sample Answer 5-2**

```
- Add a line
import random (at the first line)
==>
import random
import cs1robots

- Delete a line
import random (at the first line)
==>

- Change a line
self.position = 0 (at Player class __init__ function)
==>
self.position = 1
```

**Hints**

- Read the main function carefully because it is the entry point of the program.
- Some parts of `Python Code 5-2` are omitted with showing the (...) mark to reduce the space. You can solve this problem without changing these omitted parts.
- Please carefully check other parts of the program that may be affected by a change that you make. The monopoly game should play well on the revised code.
- Indicate the exact location of the changes that you made in the program. Ambiguous answers cannot get any partial points.

**Answer 5-2 (8 points)**