General notes:

- When writing programs, comment your code if you think it may be unclear
- You will lose a small # of points for syntax errors so double-check your work to ensure that it is syntactically correct.
- No calculators or any other devices. Just a pencil and your mind
- If you need more space, use the back of each page
1. Given the following Strings:

   good = "You are the best!"
   bad = "You are the rest?"

Trace the output (10 points):

   print (good[0])
   Y

   print (good[3])
   —

   print (bad[:3])
   You

   print (good.isalpha())
   False

   print (good[len(bad)-3])
   s

   print (bad[-5:].upper())
   REST?

   print (bad[-1:].isalpha())
   False

   print (bad[0].isalpha())
   True

   print (good.find(bad[-4:-2]))
   13

   print (bad[4:7] in good)
   True
2. Trace the output of the following code samples (8 points)

```python
for i in range(3, 8):
    print("num =", i, end=" ")
num = 3 num = 4 num = 5 num = 6 num = 7
```

```python
for i in [3, 8]:
    print("num =", i, end=" ")
num = 3 num = 8
```

```python
for i in range(3, 8, 3):
    print("num =", i, end=" ")
num = 3 num = 6
```

```python
for i in range(3, 8, 3):
    for j in range(2, 4):
        print("num 1 =", i, "- num 2 =", j)
num 1 = 3 - num 2 = 2
num 1 = 3 - num 2 = 3
num 1 = 6 - num 2 = 2
num 1 = 6 - num 2 = 3
```
3. Trace the output of the following programs (6 points):

```python
def signJam(x):
    print ("-+" * x)

password = "4avai1"

for i in password:
    if i.isdigit():
        signJam(int(i))

-+-+++--
-+-


def red_leather(x):
    print ("red:", x)
    x -= 3
    sticks(x)
    print ("red:", x)

def yellow_leather(x):
    print ("yellow:", x)
    x -= 1
    print ("yellow:", x)
    return x

x = 3
red_leather(x + 2)
print ("end:", x)

red: 5
yellow: 2
yellow: 1
red: 2
end: 3
```
Questions #4 through 8 are programming questions. Select 4 of these questions to solve (6 points each)

4. Business is booming at 'House of Cards’, a local business card shop, and they want you to make them a function to automate their business card online ordering system. There are basic rates for cards with extras that users can purchase.

- Business cards cost $8.00 for the first 100 cards.
- Every additional 100 cards are $4.00.
- Black ink is considered ‘1-color print’ and that’s included
- Every color thereafter is $3.00 per 100 cards
- If they want glossy paper, it’s $5.00 per 100 cards

Write a FUNCTION called card_calc that accepts 3 ARGUMENTS – the number of cards, the amount of colors, and glossy paper or not. Compute the total cost of an order using these three inputs and RETURN it. Note that if any input is invalid (i.e. # of cards isn’t a multiple of 100, less than one color of ink) then your function should return the string "INVALID ORDER". You only need to worry about the function, nothing else. Here’s a sample program to show your function in action.

```
print ( card_calc(300, 3, "y") ) >> 49.0
print ( card_calc(300, 1, "bazooka") ) >> INVALID ORDER
print ( card_calc(600, 5, "n") ) >> 100.0
```

def card_calc(amount, colors, gloss):
    if amount % 100 == 0:
        amountCost = amount/100 * 4 + 4.00
    else:
        return "INVALID ORDER"
    if colors == 1:
        colorsCost = 0
    elif colors > 1:
        colorsCost = amount/100 * ((colors - 1) * 3)
    else:
        return "INVALID ORDER"
    if gloss == "y":
        glossCost = amount/100 * 5.00
    elif gloss == "n":
        glossCost = 0
    else:
        return "INVALID ORDER"

    return amountCost + colorsCost + glossCost
5. Write a program that continually prompts the user to enter in specifications for a business card order (amount, number of colors, and glossy or not). Then use the “card_calc” function written for the previous question to compute the cost of the desired order. If the order is valid you should add the cost of the order to the user’s bill and provide them with a running total of their bill. If the order isn’t valid you should report this. Allow the user to continue to enter in orders until they choose to end the program. Here’s a sample running of your program:

Your current bill is $0
How many business cards would you like? 300
How many colors should the card be? 3
Do you want glossy cards? Type 'y' or 'n': y
This order will cost $49.0
Type 'yes' to place another order: yes

Your current bill is $49.0
How many business cards would you like? 300
How many colors should the cards be? 3
Do you want glossy cards? Type 'y' or 'n': bazooka
This is not a valid order
Type 'yes' to place another order: no

Your total bill is $49.0

order = "yes"
bill = 0

while order == "yes":
    print("Your current bill is $", bill, sep="")
    cards = int(input("How many business cards would you like? "))
    colors = int(input("How many colors should the cards be? "))
    gloss = input("Do you want glossy cards? Type 'y' or 'n': ")
    totalCost = card_calc(cards, colors, gloss)

    if totalCost == "INVALID ORDER":
        print("That's not a valid order.")
    else:
        print("This order will cost $", totalCost, sep="")
        bill += totalCost

    order = input("Type 'yes' to place another order: ")
    print()
print("Your total bill is $", bill, sep="")
Enter the text to run through the Jnoeser: \textit{I once met Sarah, plain and tall, last name Jones.}

Sarah appears at index: 11
Jones appears at index: 44
New text: I once met Sraah, plain and tall, last name Jnoes.
Type ‘yes’ to test more text: yes
Enter the text to run through the Jnoeser: Sea shells by the sea shore.

There is no Sarah or Jones in this text.
Type ‘yes’ to test more text: yes
Enter the text to run through the Jnoeser: I once knew a lady named Sarah.

Sarah appears at index: 25
New text: I once knew a lady named Sraah.
Type ‘yes’ to test more text: no

GoAgain = ‘yes’

while GoAgain == ‘yes’:
    Text = input("Enter the text to run through the Jnoeser: ")
    print()

    if "Sarah" not in Text and "Jones" not in Text:
        print("There is no Sarah or Jones in this text.")
        GoAgain = input("Type ‘yes’ to test more text: ")
        continue

    if "Sarah" in Text:
        SarahPos = Text.find("Sarah")
        print("John appears at: ", SarahPos)
        Text = Text.replace("Sarah", "Sraah")

    if "Jones" in Text:
        JonesPos = Text.find("Jones")
        print("Jones appears at: ", JonesPos)
        Text = Text.replace("Jones", "Jnoes")

    print("New text: ", Text)
    GoAgain = input("Type ‘yes’ to test more text: ")
def cipher(phrase):
    newPhrase ="

    for i in phrase:
        newPhrase += str(ord(i)*2) + " 

    return newPhrase

phrase = input("What do you want to encode: ")
newPhrase = cipher(phrase)
print(newPhrase)
8. Aki Arkbard is a great mathematician. But he hates everything that isn’t glorious numbers and punctuation (as it’s often used in math). He’s getting older, a little crazier, and his family has asked you to write a program that strips all pieces of text of everything that isn’t numbers and punctuation. Therefore, he should never see any letters or spaces. The program should consist of a **FUNCTION** that takes a single input **ARGUMENT**: a string. It should **RETURN** two values: the characters that are numbers/punctuation for him to see, and, for his family, the letters and spaces that was subtracted from it.

Besides the function itself, prompt the user for input string, execute the function, and print the return value like below. Don’t worry about making sure the inputted data is accurate or making it prompt continuously.

Sample input/output:

Enter the text to run through the Arkbarder: **it. Is. !gr8. 2. Be, near, this; crazy…dude!**

Aki’s text is: **..!8.2.,..;…!**
But don’t show him: **It Is gr  Be near this crazydude**

```python
def wordSplice(phrase):
    newPhrase = ""
    secret = ""
    for i in phrase:
        if i.isspace() or i.isalpha():
            secret += i
        else:
            newPhrase += i

    return newPhrase, secret

phrase = input("Enter the text to run through the Arkbarder: ")
print()
new, hidden = wordSplice(phrase)
print("Aki’s text is: ", new)
print("But don’t show him: ", hidden)
```