QUESTION 1

What data type is each of the following?

In [15]:
```
type (5)
```

Out[15]: int

In [2]:
```
type (5.0)
```

Out[2]: float

In [3]:
```
type (5>1)
```

Out[3]: bool

In [4]:
```
type ('5')
```

Out[4]: str

In [5]:
```
type (5*2)
```

Out[5]: int

In [6]:
```
type ('5'*2)
```

Out[6]: str

In [7]:
```
type ('5'+'2')
```

Out[7]: str

In [8]:
```
type (5/2)
```

Out[8]:
```
float
```

In [9]:
```
type (5//2)
```

Out[9]:
```
int
```

In [10]:
```
type ([5,2,1])
```

Out[10]:
```
list
```

In [11]:
```
type (5 in[1,4,6])
```

Out[11]:
```
bool
```

In [12]:
```
import math
type (math.pi)
```

Out[12]:
```
float
```

QUESTION 2

Write (and evaluate) Python expressions that answer these questions:

a. How many letters are there in 'Supercalifragilisticexpialidocious'?

In [17]:
```
Super = ('Supercalifragilisticexpialidocious')
len (Super)
```

Out[17]:
```
34
```

b. Does 'Supercalifragilisticexpialidocious' contain 'ice' as a substring?

In [18]:
```
Super = 'Supercalifragilisticexpialidocious'
if 'ice' in Super:
 print ('Yes')
else:
 print ('No')
```

```
Yes
```

c. Which of the following words is the longest:

Supercalifragilisticexpialidocious, Honorificabilitudinitatibus, or Bababadalgharaghtakamminarronnkonn?

In [19]:
```
# Putting all the words into variables

first_word = 'Supercalifragilisticexpialidocious'
second_word = 'Honorificabilitudinitatibus'
third_word = 'Honorificabilitudinitatibus'
```

In [20]:
```
# Compare the lenght of the words

compare_list = [len(first_word), len(second_word), len(third_word)]
```

In [21]:
```
# Using the 'if-elif-else' clause to determine the longest word

if max(compare_list) == compare_list[0]:
    print(first_word + " is the longest of the three words")
elif max(compare_list) == compare_list[1]:
    print(second_word + " is the longest of the three words")
else:
    print(third_word + " is the longest of the three words")
```

```
Supercalifragilisticexpialidocious is the longest of the three words
```

d. Which composer comes first in the dictionary: 'Berlioz', 'Borodin', 'Brian',

'Bartok', 'Bellini', 'Buxtehude', 'Bernstein'. Which one comes last?

In [22]:
```
# I create a list with the names of the composers then sort the list alphabetically
composers = ['Berlioz', 'Borodin', 'Brian', 'Bartok', 'Bellini', 'Buxtehude', 'Bernstein']
sorted_composers = sorted(composers)
```

In [23]:
```
# I check for the first composer and the last composer

print("The composer name that comes first alphabetically is: " + sorted_composers[0])
print("The composer name that comes last alphabetically is: " + sorted_composers[-1])
```

```
The composer name that comes first alphabetically is: Bartok
The composer name that comes last alphabetically is: Buxtehude
```

Question 3

a. Write a function inside (x,y,x1,y1,x2,y2) that returns True or False depending on whether the point (x,y) lies in the rectangle with lower left corner (x1,y1) and upper right corner (x2,y2).

inside(1,1,0,0,2,3) True inside(-1,-1,0,0,2,3) False

In [24]:
```
# Answer:

def inside(x,y,x1,y1,x2,y2):
    print(x in range(x1,x2) and y in range(y1,y2))


inside(1,1,0,0,2,3)

inside(-1,-1,0,0,2,3)
```

```
True
False
```

b. Use function inside() from part a. to write an expression that tests whether the point (1,1) lies in both of the following rectangles: one with lower left corner (0.3, 0.5) and upper right corner (1.1, 0.7) and the other with lower left corner (0.5, 0.2) and upper right corner (1.1, 2).

In [27]:
```
# Answer:Testing that x lies between x1 and x2 i.e. x is greater than x1 but less than x2 and y lies between
 y1 and y2 i.e. y is greater than y1 but less than y2

def inside(x,y,x1,y1,x2,y2):

    x1=int(10*x1)

    y1=int(10*y1)

    x2=int(10*x2)

    y2=int(10*y2)

    print(10*x in range(x1,x2) and 10*y in range(y1,y2))



inside(1,1,0.3,0.5,1.1,0.7)

inside(1,1,0.5,0.2,1.1,2)
```

```
False
True
```

Question 4

You can turn a word into pig-Latin using the following two rules (simplified):

• If the word starts with a consonant, move that letter to the end and append 'ay'. For example, 'happy' becomes 'appyhay' and 'pencil' becomes 'encilpay'. • If the word starts with a vowel, simply append 'way' to the end of the word. For example, 'enter' becomes 'enterway' and 'other' becomes 'otherway' . For our purposes, there are 5 vowels: a, e, i, o, u (so we count y as a consonant). Write a function pig() that takes a word (i.e., a string) as input and returns its pigLatin form. Your function should still work if the input word contains upper case characters. Your output should always be lower case however.

In [39]:
```
#I initialize a variable for the vowels


def pig(str):

    vowels=['a','e','u','i','o','A','E','I','U','O']

    if str[0] in vowels:

        return str.lower()+'hay'

    else:

        return str[1:].lower()+str[0].lower()+'ay'


pig('happy')
```

Out[39]: 'appyhay'

In [40]:
```
pig('Enter')
```

Out[40]: 'enterhay'

Question 5

File bloodtype1.txt records blood-types of patients (A, B, AB, O or OO) at a clinic.

Write a function bldcount() that reads the file with name name and reports (i.e., prints) how many patients there are in each bloodtype.

Question 6

Write a function curconv() that takes as input:

1. a currency represented using a string (e.g., 'JPY' for the Japanese Yen or 'EUR' for the Euro)
2. an amount and then converts and returns the amount in US dollars. curconv('EUR', 100)

In [57]:
```
# Creating a dictionary with the currencies and their rates

conv_dictionary = {
'AUD': 1.0345157,
'CHF': 1.0237414,
'CNY': 0.1550176,
'DKK': 0.1651442,
'EUR': 1.2296544,
'GBP': 1.5550989,
'HKD': 0.1270207,
'INR': 0.0177643,
'JPY': 0.01241401,
'MXN': 0.0751848,
'MYR': 0.3145411,
'NOK': 0.1677063,
'NZD': 0.8003591,
'PHP': 0.0233234,
'SEK': 0.148269,
'SGD': 0.788871,
'THB': 0.0313789
}

def curconv(cur, amount):
    usd_amt = conv_dictionary[cur] * amount
    curconv('EUR', 100)
    print(usd_amt)
    curconv('JPY', 100)
    print (usd_amt)
```

In [ ]:
```
```

Question 7

In [59]:
```
# Trying to add incompatible variables, as in adding 6 + ‘a’

6 + 'a'

#Type Error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-59-97f6d3731181> in <module>
      1 # Trying to add incompatible variables, as in adding 6 + ‘a’
      2
----> 3 6 + 'a'
      4
      5 #Type Error

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [61]:
```
# Referring to the 12th item of a list that has only 10 items

numbers = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
print (numbers[12])

#Index Error
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-61-4778dd6f9611> in <module>
      2
      3 numbers = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
----> 4 print (numbers[12])

IndexError: list index out of range
```

In [63]:
```
# Using a value that is out of range for a function's
input, such as calling math.sqrt(-1.0)

math.sqrt(-1.0)

# Syntax Error
```

```
  File "<ipython-input-63-c97f52311a80>", line 2
    input, such as calling math.sqrt(-1.0)
                ^
SyntaxError: invalid syntax
```

In [66]:
```
#Using an undeclared variable, such as print(x) when x has not been defined

print (g)

# Name Error, g not defined
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-66-28f4e20ddd86> in <module>
      1 #Using an undeclared variable, such as print(x) when x has not been defined
      2
----> 3 print (g)
      4
      5 # Syntax Error, g not defined

NameError: name 'g' is not defined
```

Question 8

Encryption is the process of hiding the meaning of a text by substituting letters in the message with other letters, according to some system. If the process is successful, no one but the intended recipient can understand the encrypted message. Cryptanalysis refers to attempts to undo the encryption, even if some details of the encryption are unknown (for example, if an encrypted message has been intercepted). The first step of cryptanalysis is often to build up a table of letter frequencies in the encrypted text. Assume that the string letters is already defined as 'abcdefghijklmnopqrstuvwxyz'. Write a function called frequencies() that takes a string as its only parameter, and returns a list of integers, showing the number of times each character appears in the text. Your function may ignore any characters that are not in letters

In [70]:
```
encryption_string = 'abcdefghijklmnopqrstuvwxyz'
my_phrase = ('The quick red fox got bored and went home')
def frequencies(my_phrase):
    encrypted_text = []
for letter in encryption_string:
    freq = my_phrase.count(letter)
encrypted_text.append(freq)
print('Encrypted message: ' + str(encrypted_text))

frequencies('The quick red fox got bored and went home.')
frequencies('apple')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-70-cbb87ede6626> in <module>
      5 for letter in encryption_string:
      6     freq = my_phrase.count(letter)
----> 7 encrypted_text.append(freq)
      8 print('Encrypted message: ' + str(encrypted_text))
      9

NameError: name 'encrypted_text' is not defined
```

Question 9

The Sieve of Erastophenes is an algorithm -- known to ancient Greeks -- that finds all prime numbers up to a given number n. It does this by first creating a list L from 2 to n and an (initially empty) list primeL. The algorithm then takes the first number in list L (2) and appends it to list primeL, and then removes 2 and all its multiples (4,6,8,10,12, ...) from L. The algorithm then takes the new first number in L (3) and appends it to list primeL, and then removes from L 3 and all its remaining multiples (9,15,21,...). So, in every iteration, the first number of list L is appended to list primeL and then it and its multiples are removed from list L. The iterations stop when list L becomes empty. Write a function sieve() that takes as input a positive integer n, implements the above algorithm, and returns a list of all prime numbers up to n.

def sieve(number_limit):

In [73]:
```
def sieve(L):

    primeCheck = []

    primeL=[]

    for i in range(2, L+1):

        if i not in primeCheck:

            primeL.append(i)

            for j in range(i*i, L+1, i):

                primeCheck.append(j)

    print(primeL)



sieve(56)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53]
```

In [74]:
```
sieve(368)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 10
7, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 2
29, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353,
359, 367]
```

In [75]:
```
sieve(32)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]
```

Question 10

Implement function triangleArea(a,b,c) that takes as input the lengths of the 3 sides of a triangle and returns the area of the triangle. By Heron's formula, the area of a triangle with side lengths a, b, and c is s(s - a)(s -b)(s -c), where s = (a+b+c)/2.

In [76]:
```
import math

def triangleArea(a,b,c):

 s= ((a+b+c)/2)

 area = math.sqrt(s*(s-a)*(s-b)*(s-c))

 return area

print(triangleArea(2,2,2))
```

```
1.7320508075688772
```

In [ ]:
```

```