Problem Set 1

Question 1

What data type is each of the following (evaluate where necessary)?

```
5
5.0
5 > 1
'5'
5 * 2
'5' * 2
'5' + '2'
5 / 2
5 % 2
{5, 2, 1}
5 == 3
Pi (the number)
```

In [1]:
```
print(f"datatype of 5 is",type(5).__name__)
print(f"datatype of 5.0 is",type(5.0).__name__)
print(f"datatype of 5 > 1 is",type(5 > 1).__name__)
print(f"datatype of '5' is",type('5').__name__)
print(f"datatype of 5 * 2 is",type(5 * 2).__name__)
print(f"datatype of '5' * 2 is",type('5' * 2).__name__)
print(f"datatype of '5' + '2' is",type('5' + '2').__name__)
print(f"datatype of 5 / 2 is",type(5 / 2).__name__)
print(f"datatype of 5 % 2 is",type(5 % 2).__name__)
print(f"datatype of {{5, 2, 1}} is",type({5, 2, 1}).__name__)
print(f"datatype of 5 == 3 is",type(5 == 3).__name__)

import math
print(f"datatype of Pi (the number) is",type(math.pi).__name__)
```

```
datatype of 5 is int
datatype of 5.0 is float
datatype of 5 > 1 is bool
datatype of '5' is str
datatype of 5 * 2 is int
datatype of '5' * 2 is str
datatype of '5' + '2' is str
datatype of 5 / 2 is float
datatype of 5 % 2 is int
datatype of {5, 2, 1} is set
datatype of 5 == 3 is bool
datatype of Pi (the number) is float
```

Question 2

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

```
a. How many letters are there in 'Supercalifragilisticexpialidocious'?
b. Does 'Supercalifragilisticexpialidocious' contain 'ice' as a substring?
c. Which of the following words is the longest:
    Supercalifragilisticexpialidocious, Honorificabilitudinitatibus, or
```

Bababadalgharaghtakamminarronnkonn?

d. Which composer comes first in the dictionary: 'Berlioz', 'Borodin', 'Brian', 'Bartok', 'Bellini', 'Buxtehude', 'Bernstein'. Which one comes last?

In [8]:
```
# Solution 2a.
givenWord = 'Supercalifragilisticexpialidocious'
numberOfLetters = len(givenWord) # To get the length of the word
print(f"There are {numberOfLetters} letters in '{givenWord}'.")
```

```
There are 34 letters in 'Supercalifragilisticexpialidocious'.
```

In [3]:
```
# Solution 2b.
givenWord2 = 'Supercalifragilisticexpialidocious'
substring = 'ice' # The substring to search for
contain_Substring = substring in givenWord2 # To check if the substring is in the word
if contain_Substring:
    print(f"Yes,'{givenWord2}' contains the substring '{substring}'.")
else:
    print(f"No,'{givenWord2}' does not contain the substring '{substring}'.")
```

```
Yes,'Supercalifragilisticexpialidocious' contains the substring 'ice'.
```

In [9]:
```
# Solution 2c.
givenwords3 = [
    'Supercalifragilisticexpialidocious',
    'Honorificabilitudinitatibus',
    'Bababadalgharaghtakamminarronnkonn',
]

# To store the longest words and their length
longest_words = []
longest_word_length = 0

# To iterate through the words
for word in givenwords3:
    # To calculate the length of the current word
    word_length = len(word)

    # To print the length of the current word
    print(f"The length of '{word}' is {word_length}.")

    # To check if the current word is longer than or equal to the longest words found so
    if word_length > longest_word_length:
        longest_words = [word]
        longest_word_length = word_length
    elif word_length == longest_word_length:
        longest_words.append(word)

# To print the longest words
if len(longest_words) > 1:
    print("The longest words are:")
    for longest_word in longest_words:
        print(f"'{longest_word}' with a length of {longest_word_length}.")
else:
    print(f"The longest word is '{longest_words[0]}' with a length of {longest_word_leng
```

```
The length of 'Supercalifragilisticexpialidocious' is 34.
The length of 'Honorificabilitudinitatibus' is 27.
The length of 'Bababadalgharaghtakamminarronnkonn' is 34.
The longest words are:
'Supercalifragilisticexpialidocious' with a length of 34.
'Bababadalgharaghtakamminarronnkonn' with a length of 34.
```

In [10]:
```
# Solution 2d.
composers_list = ['Berlioz', 'Borodin', 'Brian', 'Bartok', 'Bellini', 'Buxtehude', 'Bern
first_composer = min(composers_list)
```

```
last_composer = max(composers_list)
print(f"The first composer in the dictionary is '{first_composer}'.")
print(f"The last composer in the dictionary is '{last_composer}'.")
```

```
The first composer in the dictionary is 'Bartok'.
The last composer in the dictionary is 'Buxtehude'.
```

Question 3

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
squareroot of {s(s - a)(s -b)(s -c)}, where s = (a+b+c)/2.
>>> triangleArea(2,2,2)
1.7320508075688772
```

In [12]:
```
import math

def triangleArea(a, b, c):
    s = (a + b + c) / 2 # To calculate the semi-perimeter (s)
    calculate_area = math.sqrt(s * (s - a) * (s - b) * (s - c)) # To calculate the area
    return calculate_area

# To display triangleArea(2,2,2)
result = triangleArea(2, 2, 2)
print(result)
```

```
1.7320508075688772
```

Question 4

Write a program in python to separate odd and even integers in separate arrays.

```
Go to the editor
Test Data :
Input the number of elements to be stored in the array :5
Input 5 elements in the array :
element - 0 : 25
element - 1 : 47
element - 2 : 42
element - 3 : 56
element - 4 : 32
Expected Output:
The Even elements are:
42 56 32
The Odd elements are :
25 47
```

In [17]:
```
input_elements = int(input("Input the number of elements to be stored in the array : "))

even_number = []
odd_number = []

for i in range(input_elements):
    j = int(input(f"element - {i} : "))
```

```
    if j%2==0:
        even_number.append(j)
    else:
        odd_number.append(j)

print("The Even elements are:")
print(even_number)
print("The Odd elements are:")
print(odd_number)
```

```
Input the number of elements to be stored in the array : 5
element - 0 : 25
element - 1 : 47
element - 2 : 42
element - 3 : 56
element - 4 : 32
The Even elements are:
[42, 56, 32]
The Odd elements are:
[25, 47]
```

Question 5

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
```

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 [18]:
```
def inside(x, y, x1, y1, x2, y2):
    return x1 <= x <= x2 and y1 <= y <= y2

# To check given inputs
print(inside(1, 1, 0, 0, 2, 3))
print(inside(-1, -1, 0, 0, 2, 3))
```

```
True
False
```

In [19]:
```
# Given rectangle coordinates
rectangle1 = (0.3, 0.5, 1.1, 0.7)
rectangle2 = (0.5, 0.2, 1.1, 2)

# To check if (1, 1) is inside both rectangles
is_inside_rect1 = inside(1, 1, *rectangle1)
is_inside_rect2 = inside(1, 1, *rectangle2)

# To display the results
print("Is (1, 1) inside the first rectangle?", is_inside_rect1)
print("Is (1, 1) inside the second rectangle?", is_inside_rect2)
```

```
Is (1, 1) inside the first rectangle? False
Is (1, 1) inside the second rectangle? True
```

Question 6

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 pig-Latin form. Your function should still work if the input word contains upper case
characters. Your output should always be lower case however.

```
>>> pig('happy')
'appyhay'
>>> pig('Enter')
'enterway'
```

In [21]:
```
def pig(givenWord6):

    givenWord6 = givenWord6.lower() # Convert word to lowercase to ensure consistency

    vowels = {'a', 'e', 'i', 'o', 'u'}

    if givenWord6[0] not in vowels: # Check if the word starts with a consonant
        pig_latin_word = givenWord6[1:] + givenWord6[0] + 'ay' # Move the first letter t
    else:
        pig_latin_word = givenWord6 + 'way' # If it starts with a vowel, simply append '

    return pig_latin_word

# Display the given inputs in questions
print(pig('happy'))
print(pig('Enter'))
```

```
appyhay
enterway
```

Question 7

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.

```
>>> bldcount('bloodtype.txt')
There are 10 patients of blood type A.
There is one patient of blood type B.
There are 10 patients of blood type AB.
There are 12 patients of blood type O.
There are no patients of blood type OO.
```

In [24]:
```
def bldcount(filename):

    file = open(filename, 'r') # To open the file for reading

    data = file.readline() # To read the first line from the file

    words_list = [] # To initialize an empty list to store the words
```

```
    blood_types = ['A', 'B', 'AB', 'O', 'OO'] # To efine the blood types to count

    words_list.append(data.split(" ")) # To split the words and store them in a list

    for blood_type in blood_types: # To iterate through the blood types and count patien

        count = words_list[0].count(blood_type) # To count the occurrences of the blood

        print("There are {} patients of blood type {}.".format(count, blood_type)) # To

# To call the function
bldcount('bloodtype1.txt')
```

```
There are 15 patients of blood type A.
There are 1 patients of blood type B.
There are 13 patients of blood type AB.
There are 15 patients of blood type O.
There are 0 patients of blood type OO.
```

Question 8

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)
122.96544
>>> curconv('JPY', 100)
1.241401
```

The currency rates you will need are stored in file currencies.txt:

```
AUD 1.0345157 Australian Dollar
CHF 1.0237414 Swiss Franc
CNY 0.1550176 Chinese Yuan
DKK 0.1651442 Danish Krone
EUR 1.2296544 Euro
GBP 1.5550989 British Pound
HKD 0.1270207 Hong Kong Dollar
INR 0.0177643 Indian Rupee
JPY 0.01241401 Japanese Yen
MXN 0.0751848 Mexican Peso
MYR 0.3145411 Malaysian Ringgit
NOK 0.1677063 Norwegian Krone
NZD 0.8003591 New Zealand Dollar
PHP 0.0233234 Philippine Peso
SEK 0.148269 Swedish Krona
SGD 0.788871 Singapore Dollar
THB 0.0313789 Thai Baht
```

In [27]:

```
def curconv(target_currency, amount):
    currency_data = {}

    currency_file = open('currencies.txt') # To open and read the currency conversion fi
    file_lines = currency_file.readlines()

    # To parse the lines to create a dictionary with currency codes as keys and conversi
    for line in file_lines:
        code = line[:3].strip() # To extracts the first 3 characters from the line (assu
```

```
        values = line[4:].split("\t") # To extracts the characters from the 4th position
        currency_data[code] = (values[0].strip(), values[1].strip())

    # To retrieve the conversion value for the target currency and calculate the convert
    conversion_rate = float(currency_data[target_currency][0])
    converted_amount = amount * conversion_rate

    # To display the converted amount
    print(converted_amount)

# Example usage:
curconv('EUR', 100)
```

```
122.96544
```

In [28]:
```
curconv('JPY', 100)
```

```
1.241401
```

Question 9

Each of the following will cause an exception (an error). Identify what type of exception each will cause.

Trying to add incompatible variables, as in adding 6 + ‘a’
Referring to the 12th item of a list that has only 10 items
Using a value that is out of range for a function’s input, such as calling math.sqrt(-1.0)
Using an undeclared variable, such as print(x) when x has not been defined
Trying to open a file that does not exist, such as mistyping the file name or looking in the wrong directory.

In [47]:
```
# Trying to add an integer and a string will result in a TypeError.

try:
    result = 6 + 'a'
except TypeError as e:
    print(f"Exception type: {type(e).__name__}")
```

```
Exception type: TypeError
```

In [40]:
```
# Trying to access an element beyond the list's index range, an IndexError will occur.

try:
    my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    item = my_list[12]
except IndexError as e:
    print(f"Exception type: {type(e).__name__}")
```

```
Exception type: IndexError
```

In [41]:
```
# Trying to calculate the square root of a negative number will lead to a ValueError.

import math

try:
    result = math.sqrt(-1.0)
except ValueError as e:
    print(f"Exception type: {type(e).__name__}")
```

```
Exception type: ValueError
```

In [42]:
```
# Trying to print an undeclared variable (in this case, 'var') will result in a NameErro
```

```
try:
    print(var)
except NameError as e:
    print(f"Exception type: {type(e).__name__}")
```

```
Exception type: NameError
```

In [43]:
```
# Trying to open a non-existent file (such as 'non_existent_file.txt'), the code will tr

try:
    with open('non_existent_file.txt', 'r'):
        pass
except FileNotFoundError as e:
    print(f"Exception type: {type(e).__name__}")
```

```
Exception type: FileNotFoundError
```

Question 10

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.
>>> frequencies('The quick red fox got bored and went home.')
[1, 1, 1, 3, 5, 1, 1, 2, 1, 0, 1, 0, 1, 2, 4, 0, 1, 2, 0, 2,
1, 0, 1, 1, 0, 0]
>>> frequencies('apple')

In [50]:
```
def frequencies(text):

    letters = 'abcdefghijklmnopqrstuvwxyz' # To define the string of letters

    char_counts = [0] * 26 # For each index representing a different character (a to z).

    text = text.lower() # To convert the input text to lowercase for consistency

    for char in text: # To iterate through each character in the input text
        if char in letters:
            index = letters.index(char) # To calculate the index of the character in let
            char_counts[index] += 1

    return char_counts

result1 = frequencies('The quick red fox got bored and went home.')
print(result1)
```

```
[1, 1, 1, 3, 5, 1, 1, 2, 1, 0, 1, 0, 1, 2, 4, 0, 1, 2, 0, 3, 1, 0, 1, 1, 0, 0]
```

In [51]:
```
result2 = frequencies('apple')
print(result2)
```

```
[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```