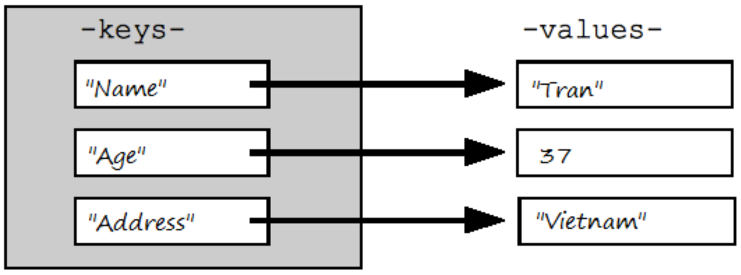
Dictionaries

In python, dictionaries are a data type that is similar to a list but uses **keys** to access information instead of number an **index**. Please see diagram



You can think of a dictionary as a fancy **unordered** list where you get to create specific names for each location of data instead of using an index number.

This is great for humans. We like identifying things by name rather than numbers. If I want to look up information about "Tommy", I can ask for information about Tommy instead of having tommy's information stored in index box 112343.

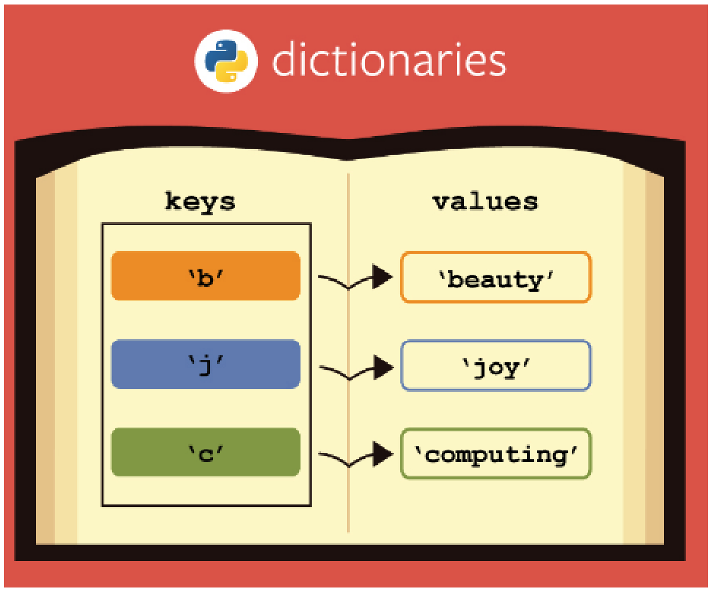
The values in a dictionary aren't numbered - they aren't in any specific order, either. You *can* add, remove, and modify the values and keys in dictionaries.

The **dictionary** data type in Python is known in other programming languages as **associative arrays**, **hashes**, or **mapping**. The concept is the same for all languages and is important to understand.

Creating a dictionary:

Properties of Dictionary:

- There is **no defined order** to keys and values (you cannot sort a dictionary using keys)
- Can't **locate values with an index** (no slicing etc).
- Only one entry per `key` is allowed.
- The `values` in the dictionary can be any data type (lists, strings, ints, floats)
- `keys` must be **immutable** like numbers, tuples or strings.



Exericse#1 – Enter the following examples into an IDE of your choice. Examine the input and output carefully to see what is going on.

Please type them in (so you get practice with the syntax). Don’t copy and paste.

Creating Dictionaries and getting Key Values:

```
Dict1 = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
print (Dict1['Tiffany'])
print (Dict1['Robert'])
```

Adding to a Dictionary:

From previous Dict1 try:

```
Dict1[‘Phil’]=31
Print(Dict1)
```

try:

```
basket = { 'oranges': 12, 'pears': 5, 'apples': 4 }
basket['bananas'] = 5
print(basket)
```

Update method:

```
dict = {'Tim':18,'Charlie':12,'Tiffany':22,'Robert':25}
dict.update({"Sarah":9})
print (dict)
```

del method. Deleting values from a dictionary:

try:

```
basket = {'apple': 4,'bananda':2,'tomoato':1,'carrot':9}
del basket['apple']
print (basket)
```

Iterating through a dictionary:

```
Try:

backpack = {'binder': 1,'pencils': 5,'eraser': 1,'textbook':3}
for x in backpack:
  print(x)

for x in backpack:
  print(backpack[x])

Also Try:

backpack = {'binder': 1,'pencils': 5,'eraser': 1,'textbook': 3}

print(backpack.values())
print(backpack.keys())
```

Exercise#2 (you try)

Create a **dictionary** that has movie titles for **`keys`** and amount of money each moving makes in its opening weekend (in millions) for each **`value`**. Use at least 5 movies.

Then do the following:

1. Print out just the keys
2. Print out just the values
3. Delete a movie from your dictionary
4. Add a movie from your dictionary
5. Iterate through your dictionary values and print out the movie with the highest money made in opening weekend.
6. Create a **list** from your dictionary that has the movies in order of how much money each made (greatest to least)

A dictionary of lists:

A dictionary connects two pieces of information. Remember **`values`** can be any kind of data type. Let's try using **`lists`** as **`values`**. This a common practice.

```
# This program stores people's favorite numbers, and displays them.
Please try the following: type it in and see how the input and output relate.

favorite_numbers = {'eric': [3, 11, 19, 23, 42],
                    'dave': [2, 4, 5],
                    'willie': [5, 35, 120],
                    }

print(favorite_numbers)
print(favorite_numbers['dave'])
print(favorite_numbers['dave'][0])
print(favorite_numbers['dave'][1])
print(favorite_numbers['dave'][2])
print(favorite_numbers['willie'][2])

favorite_numbers['dave'][1]=9
print(favorite_numbers)
```

Here is another way to create a dictionary with list as values:

Please try:

```
list1 = ['a', 'b', 'c']

list2 = [1, 2, 3, 4]

my_dict = {'list1': list1, 'list2': list2}
```

Exercise#3

The following dictionary is supposed to represent a characters attributes in a fantasy video game.

```
Given the following dictionary:

inventory = {
             'gold' : 500,
             'pouch' : ['flint', 'twine', 'gemstone'],
             'backpack' : ['flute','dagger','bedroll','bread loaf']
            }
```

Try to do the following:

1. Add a key to inventory called 'pocket'.
2. Set the value of 'pocket' to be a **`list`** consisting of the following items: 'seashell', 'berries', 'Lint', ‘sand’.
3. `.sort()`the items in the list stored under the 'backpack' key.
4. Then `.remove`('dagger') from the list of items stored under the 'backpack' key.
5. Change the number stored under the 'gold' key to 250.

Exercise#4

Use all the dictionary methods below on the characters attributes dictionary above:
Make sure you show some output that is the result of how you used each method.
Add a print statement to clarify to the user what each method does.

```
mydict.clear       mydict.get      mydict.pop         mydict.update

mydict.copy        mydict.items    mydict.popitem     mydict.values

mydict.fromkeys    mydict.keys     mydict.setdefault
```

Exercise#5

- Create a new dictionary, where the keys of the dictionary are the names of mountains in British Columbia, the values of the dictionary should be a list of each mountain's. Elevation in meters, and then in feet:
  `Mountains={'everest': [8848, 29029]}`
- Use google to get the information.

Then:

1. Print out just the mountains' names.
2. Print out just the mountains' elevations in meters.
3. Print out a “neatly formatted table that shows the name of each mountain, elevation in meters, and elevation in feet.

Exercise #6

Turning tuple pairs into a dictionary:

This can be a useful way to call tuple pairs by a **`key`** name rather than an **`index.`**

Try typing in following states and map codes:

```
pairs = [('California', 37253956), ('Colorado', 5029196), ('Connecticut',
3574097), ('Delaware', 897934)]
print(pairs[0][0])
print(pairs[0][1])
print(pairs[1][0])
print(pairs[1][1])
```

now try:

```
population = dict(pairs)
print (population)
```

Now print the following using dictionary methods and keys:

1. All the state names
2. All the map codes
3. Colorado’s map code
4. Delaware’s map code
5. A new dictionary with 3 additional states and map codes
6. A new dictionary with Connecticut and California removed.

Populating a dictionary using comprehensions:

try the following examples:

```
squares = {x: x*x for x in range(6)}
print(squares)


dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
# Double each value in the dictionary
double_dict1 = {k:v*2 for (k,v) in dict1.items()}
print(double_dict1)

import string
dict1={i:j for (i, j) in zip(string.ascii_lowercase[:5], range(5))}
print(dict1)
```

Exercise #7

Use comprehensions to create the following dictionaries:

1. A dictionary with number keys from 1 to 15 and values 10,20,30…150
2. A dictionary with letters for keys (a to z) and number values 1-26
3. A dictionary with number keys from 1 to 15 and triple each key for the values

Exercise#8

Write a Python program to sum all the **`values`** in a dictionary

Exercise#9

Write a Python **function** that takes in a team name as a parameter and outputs the total number of goals *the team* has scored. Make up several teams to choose from.

Example:

Goals Assists

```
Team1 = {'jim' : [1, 2], 'alex' : [4, 5], 'todd' : [7, 8]}
```

Expected result for this team would be: 12 goals.

Exercise #10

Write a Python program to find and display the **top three** selling items in a shop (in grams).

```
Sample_data = {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}
```

Expected Output:

```
item4 55
item1 45.5
item3 41.3
```

Exercise #11

Write a Python program to sort a dictionary from highest to lowest **`value`**.

```
Sample data = {'Math':81, 'Physics':83, 'Chemistry':87}

Expected output: [('Chemistry', 87), ('Physics', 83), ('Math', 81)]
```

Exercise#12

Write a Python program that can find **matching** key **`values`** in 3 or more dictionaries.
Be creative with this one.

```
dict1={'Managers': 1, 'engineers': 3, 'technicians': 5}
dict2 {'Managers': 2, 'engineers': 2, 'technicians': 5}
```

Expected output: `both companies have 5 technicians.`

Exercise#13

Create a dictionary that includes the following:

```
"banana": [1, 15]       [price in $, number of items in stock]
"apple": [2, 35]
"orange": [1.5, 27]
"pear": [3, 36]
```

Create a program that can print out a key along with its price and stock information when prompted by a user. Make sure to have the output in the following format **`EXACTLY`**:

```
APPLE
price: $2
In stock: 35
```

Now, create a program that can determine how much money you would make if you sold all of your food in the store

Exercises 14:

Write a **function** called `word_frequencies(mylist)` that accepts a **list** of **strings** called `mylist` and returns a dictionary where the keys are the words from `mylist` and the values are the number of times that word appears in `mylist`.

Exercise #15

Write a function called `accept_login(guestlistfile, username, password)` with three parameters:

1. **A dictionary** (guestlistfile) - of username **`keys`** and password **`values`**.
2. **username** - a string for a login name.
3. **password** - a string for a password.

The function should return **True** if the user exists and the password is correct and **False** otherwise.

See if you can find a txt file of user names and passwords you can import into python and create a dictionary from. I'm sure you can find a sample txt file somewhere on the internet.

Exercise#17

TeacherEase
Software for the Digital Classroom

Create a dictionary called **`Marks_Book`** that includes the dictionaries shown below.
Now create a user interface that will allow a user to generate the following:

1. A list of *all* the students in the class.

2. A list of any students marks in one of the categories shown below (homework, quizzes, tests, labs). example: `Dave’s` **`test`** `marks are [75.0, 90.0]`

3. An average for any student’s marks in a particular category: `Alice’` **`homework`** `average is 97`

4. A student’s overall average if each category is weighted as follows (homework 20%, quizzes 20%, tests 60%: `Alice has an overall average is 92%`

Make sure you interface is “user friendly”. So that anyone running your program can use it.

Example: “welcome to Marks Book!...what would you like to do?”:

Press:

1 - if you want to get a list of your students
2 – if you want to get a student’s overall average
3 – want more details about a student grades
4…….etc….

```
lloyd = {
    "name": "Dave",
    "homework": [90.0, 97.0, 75.0, 92.0],
    "quizzes": [88.0, 40.0, 94.0],
    "tests": [75.0, 90.0]
}
alice = {
    "name": "Alice",
    "homework": [100.0, 92.0, 98.0, 100.0],
    "quizzes": [82.0, 83.0, 91.0],
    "tests": [89.0, 97.0]
}
tyler = {
    "name": "Tyler",
    "homework": [0.0, 87.0, 75.0, 22.0],
    "quizzes": [0.0, 75.0, 78.0],
    "tests": [100.0, 100.0]
}
```

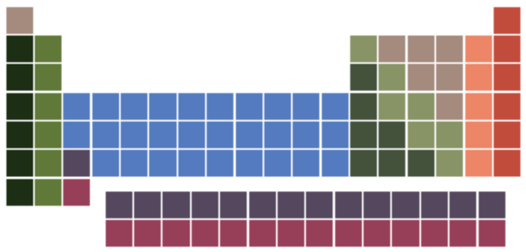
Exercise#18

Periodic Table App

The Periodic Table of the Elements was developed to organize information about the elements that make up the universe. Using **dictionaries**, write an app that lets you access information about each element in the periodic table.

Make sure you include the following information:

1. symbol,
2. name,
3. atomic number,
4. row,
5. column…
6. and anything else you want.



Provide a menu of options for users to:

1. See all the information that is stored about any element, by entering that element's symbol.

2. Choose a property and see that property for each element in the table.

To make things easier you should use the .txt file below.

Text file: https://www.itl.nist.gov/div898/software/dataplot/text/PERIODIC.TXT

You should also learn how to import the txt file and get the necessary information into your dictionary.

This link might help you learn how to do this:

https://www.computerhope.com/issues/ch001721.htm

You could also cut and paste the whole txt file and enter it as a string…if all else fails.

Bonus Features

Provide an option to view the symbols arranged **correctly** (like on a real periodic table).

Algorithm Practice:

Knowing good python tools isn't good enough to be a great programmer!....The following puzzles are University of Waterloo **Senior** contest questions. Dictionaries are not required to solve these. They are intended to exercise your problem solving skilss. Enjoy!

Problem S1: Don't pass me the ball!

Problem Description

A CCC soccer game operates under slightly different soccer rules. A goal is only counted if the 4 players, in order, who touched the ball prior to the goal have jersey numbers that are in strictly increasing numeric order with the highest number being the goal scorer.

Players have jerseys numbered from 1 to 99 (and each jersey number is worn by exactly one player).

Given a jersey number of the goal scorer, indicate how many possible combinations of players can produce a valid goal.

Input Specification

The input will be the positive integer J ($1 \leq J \leq 99$), which is the jersey number of the goal scorer.

Output Specification

The output will be one line containing the number of possible scoring combinations that could have J as the goal scoring jersey number.

Sample Input 1

```
4
```

Output for Sample Input 1

```
1
```

Sample Input 2

```
2
```

Output for Sample Input 2

```
0
```

Sample Input 3

```
90
```

Output for Sample Input 3

```
113564
```

Problem S2: Aromatic Numbers

Problem Description

This question involves calculating the value of *aromatic* numbers which are a combination of Arabic digits and Roman numerals.

An aromatic number is of the form $ARARAR \ldots AR$, where each A is an Arabic digit, and each R is a Roman numeral. Each pair AR contributes a value described below, and by adding or subtracting these values together we get the value of the entire aromatic number.

An Arabic digit A can be 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9. A Roman numeral R is one of the seven letters I, V, X, L, C, D, or M. Each Roman numeral has a base value:

Symbol	I	V	X	L	C	D	M
Base value	1	5	10	50	100	500	1000

The value of a pair AR is A times the base value of R. Normally, you add up the values of the pairs to get the overall value. However, wherever there are consecutive symbols $ARA'R'$ with R' having a *strictly bigger* base value than R, the value of pair AR must be *subtracted* from the total, instead of being *added*.

For example, the number 3M1D2C has the value $3 * 1000 + 1 * 500 + 2 * 100 = 3700$ and 3X2I4X has the value $3 * 10 - 2 * 1 + 4 * 10 = 68$.

Write a program that computes the values of aromatic numbers.

Input Specification
The input is a valid aromatic number consisting of between 2 and 20 symbols.

Output Specification
The output is the decimal value of the given aromatic number.

Sample Input 1
```
3M1D2C
```

Output for Sample Input 1
```
3700
```

Sample Input 2
```
2I3I2X9V1X
```

Output for Sample Input 2
```
-16
```

Problem S3: Absolutely Acidic

Problem Description

You are gathering readings of acidity level in a very long river in order to determine the health of the river. You have placed N sensors ($2 \leq N \leq 2\,000\,000$) in the river, and each sensor gives an integer reading R ($1 \leq R \leq 1\,000$). For the purposes of your research, you would like to know the frequency of each reading, and find the absolute difference between the two most frequent readings.

If there are more than two readings that have the highest frequency, the difference computed should be the *largest* such absolute difference between two readings with this frequency. If there is only one reading with the largest frequency, but more than one reading with the second largest frequency, the difference computed should be the *largest* absolute difference between the most frequently occurring reading and any of the readings which occur with second-highest frequency.

Input Specification

The first line of input will be the integer N ($2 \leq N \leq 2\,000\,000$), the number of sensors. The next N lines each contain the reading for that sensor, which is an integer R ($1 \leq R \leq 1\,000$). You should assume that there are at least two different readings in the input.

Output Specification

Output the positive integer value representing the absolute difference between the two most frequently occurring readings, subject to the tie-breaking rules outlined above.

Sample Input 1

```
5
1
1
1
4
3
```

Output for Sample Input 1

```
3
```

Sample Input 2

```
4
10
6
1
8
```

Output for Sample Input 2

```
9
```