Python RegEx

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern.

RegEx Module

Python has a built-in package called re, which can be used to work with Regular Expressions.

Import the re module:

import re

RegEx in Python

When you have imported the re module, you can start using regular expressions:

Example

Search the string to see if it starts with "The" and ends with "Spain":

import re  
  
txt = "The rain in Spain"  
x = re.search("^The.\*Spain$", txt)

Try it Yourself »

RegEx Functions

The re module offers a set of functions that allows us to search a string for a match:

|  |  |
| --- | --- |
| **Function** | **Description** |
| [findall](https://www.w3schools.com/python/python_regex.asp#findall) | Returns a list containing all matches |
| [search](https://www.w3schools.com/python/python_regex.asp#search) | Returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match anywhere in the string |
| [split](https://www.w3schools.com/python/python_regex.asp#split) | Returns a list where the string has been split at each match |
| [sub](https://www.w3schools.com/python/python_regex.asp#sub) | Replaces one or many matches with a string |

ADVERTISEMENT

Metacharacters

Metacharacters are characters with a special meaning:

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Example** | **Try it** |
| [] | A set of characters | "[a-m]" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta1) |
| \ | Signals a special sequence (can also be used to escape special characters) | "\d" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta2) |
| . | Any character (except newline character) | "he..o" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta3) |
| ^ | Starts with | "^hello" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta4) |
| $ | Ends with | "planet$" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta5) |
| \* | Zero or more occurrences | "he.\*o" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta6) |
| + | One or more occurrences | "he.+o" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta7) |
| ? | Zero or one occurrences | "he.?o" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta10) |
| {} | Exactly the specified number of occurrences | "he.{2}o" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta8) |
| | | Either or | "falls|stays" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta9) |
| () | Capture and group |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Special Sequences

A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Example** | **Try it** |
| \A | Returns a match if the specified characters are at the beginning of the string | "\AThe" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq1) |
| \b | Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\bain" r"ain\b" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq2) [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq2-2) |
| \B | Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\Bain" r"ain\B" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq3) [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq3-2) |
| \d | Returns a match where the string contains digits (numbers from 0-9) | "\d" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq4) |
| \D | Returns a match where the string DOES NOT contain digits | "\D" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq5) |
| \s | Returns a match where the string contains a white space character | "\s" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq6) |
| \S | Returns a match where the string DOES NOT contain a white space character | "\S" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq7) |
| \w | Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore \_ character) | "\w" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq8) |
| \W | Returns a match where the string DOES NOT contain any word characters | "\W" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq9) |
| \Z | Returns a match if the specified characters are at the end of the string | "Spain\Z" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq10) |

Sets

A set is a set of characters inside a pair of square brackets [] with a special meaning:

|  |  |  |
| --- | --- | --- |
| **Set** | **Description** | **Try it** |
| [arn] | Returns a match where one of the specified characters (a, r, or n) is present | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set1) |
| [a-n] | Returns a match for any lower case character, alphabetically between a and n | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set2) |
| [^arn] | Returns a match for any character EXCEPT a, r, and n | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set3) |
| [0123] | Returns a match where any of the specified digits (0, 1, 2, or 3) are present | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set4) |
| [0-9] | Returns a match for any digit between 0 and 9 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set5) |
| [0-5][0-9] | Returns a match for any two-digit numbers from 00 and 59 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set6) |
| [a-zA-Z] | Returns a match for any character alphabetically between a and z, lower case OR upper case | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set7) |
| [+] | In sets, +, \*, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set8) |

The findall() Function

The findall() function returns a list containing all matches.

Example

Print a list of all matches:

import re  
  
txt = "The rain in Spain"  
x = re.findall("ai", txt)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_findall)

The list contains the matches in the order they are found.

If no matches are found, an empty list is returned:

Example

Return an empty list if no match was found:

import re  
  
txt = "The rain in Spain"  
x = re.findall("Portugal", txt)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_findall2)

The search() Function

The search() function searches the string for a match, and returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

Example

Search for the first white-space character in the string:

import re  
  
txt = "The rain in Spain"  
x = re.search("\s", txt)  
  
print("The first white-space character is located in position:", x.start())

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_search)

If no matches are found, the value None is returned:

Example

Make a search that returns no match:

import re  
  
txt = "The rain in Spain"  
x = re.search("Portugal", txt)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_search2)

The split() Function

The split() function returns a list where the string has been split at each match:

Example

Split at each white-space character:

import re  
  
txt = "The rain in Spain"  
x = re.split("\s", txt)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_split)

You can control the number of occurrences by specifying the maxsplit parameter:

Example

Split the string only at the first occurrence:

import re  
  
txt = "The rain in Spain"  
x = re.split("\s", txt, 1)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_split2)

The sub() Function

The sub() function replaces the matches with the text of your choice:

Example

Replace every white-space character with the number 9:

import re  
  
txt = "The rain in Spain"  
x = re.sub("\s", "9", txt)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_sub)

You can control the number of replacements by specifying the count parameter:

Example

Replace the first 2 occurrences:

import re  
  
txt = "The rain in Spain"  
x = re.sub("\s", "9", txt, 2)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_sub2)

Match Object

A Match Object is an object containing information about the search and the result.

**Note:** If there is no match, the value None will be returned, instead of the Match Object.

Example

Do a search that will return a Match Object:

import re  
  
txt = "The rain in Spain"  
x = re.search("ai", txt)  
print(x) #this will print an object

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_match)

The Match object has properties and methods used to retrieve information about the search, and the result:

.span() returns a tuple containing the start-, and end positions of the match.  
.string returns the string passed into the function  
.group() returns the part of the string where there was a match

Example

Print the position (start- and end-position) of the first match occurrence.

The regular expression looks for any words that starts with an upper case "S":

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.span()**)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_match_span)

Example

Print the string passed into the function:

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.string**)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_match_string)

Example

Print the part of the string where there was a match.

The regular expression looks for any words that starts with an upper case "S":

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.group()**)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_match_group)

Password validation in Python

It is a general requirement to have a reasonably complex password. In this article we will see how to validate if a given password meats certain level of complexity. For that will use the regular expression module known as re.

Example -1

First we create a regular expression which can satisfy the conditions required to call it a valid password. Then we e match the given password with the required condition using the search function of re. In the below example the complexity requirement is we need at least one capital letter, one number and one special character. We also need the length of the password to be between 8 and 18.

Example

import re

pswd = 'XdsE83&!'

reg = "^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*#?&])[A-Za-z\d@$!#%\*?&]{8,18}$"

# compiling regex

match\_re = re.compile(reg)

# searching regex

res = re.search(match\_re, pswd)

# validating conditions

if res:

   print("Valid Password")

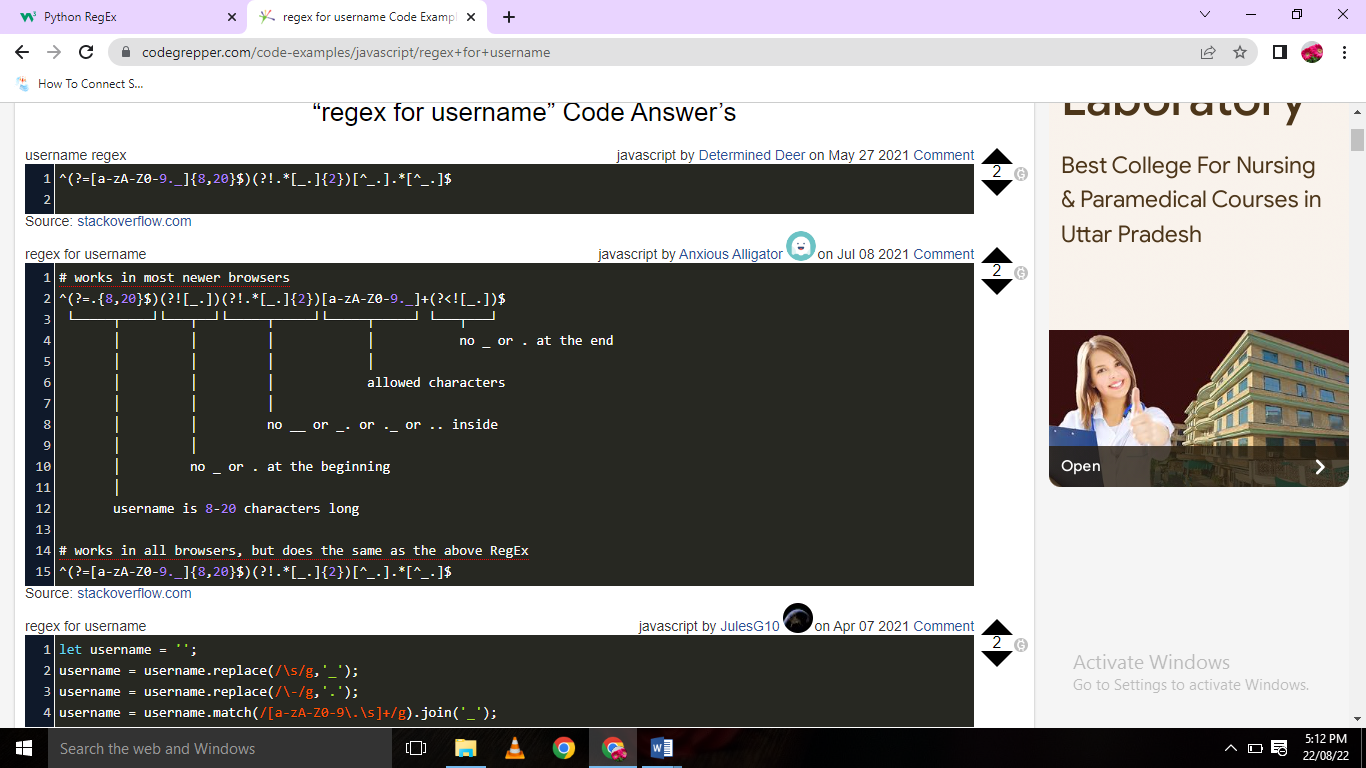
else:

   print("Invalid Password")

Output

Running the above code gives us the following result −

Valid Password



**Validate Email Address with Python**

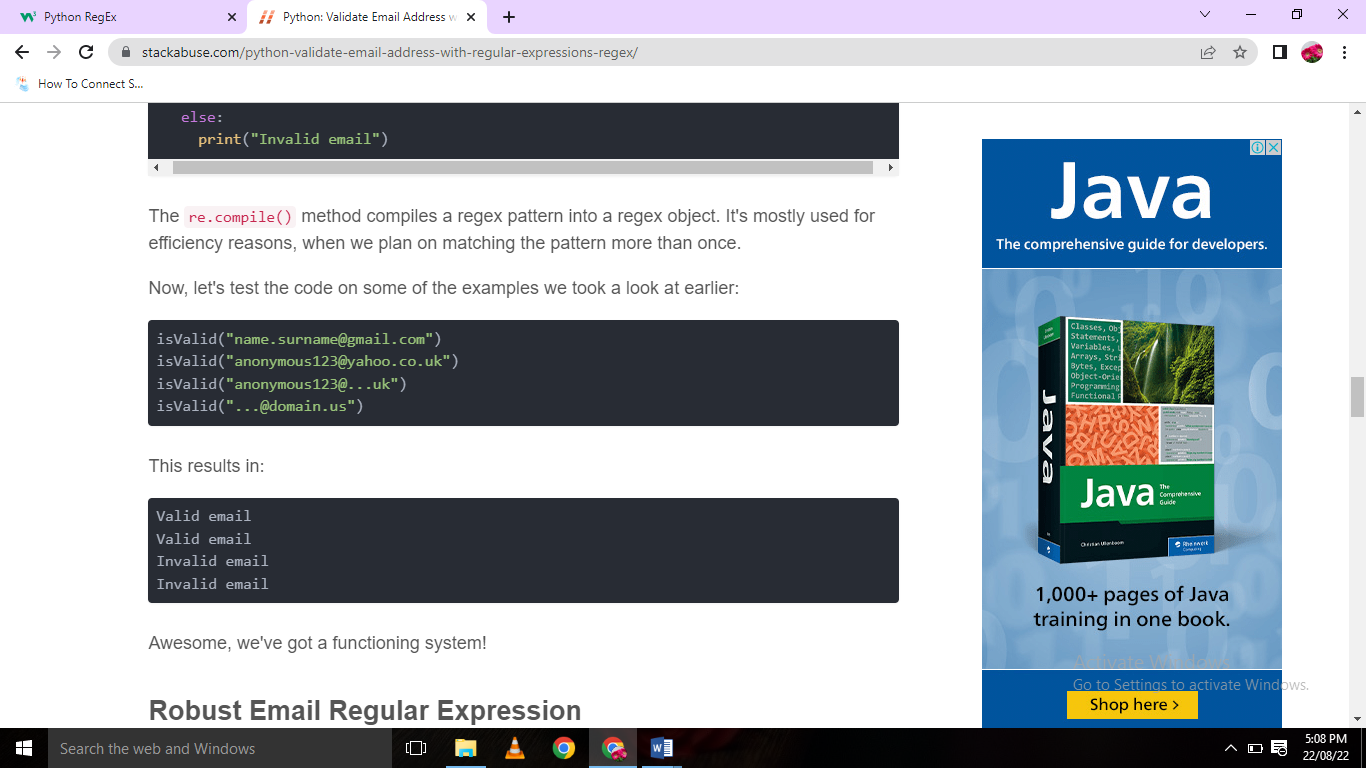
The re module contains classes and methods to represent and work with Regular Expressions in Python, so we'll import it into our script. The method that we will be using is re.fullmatch(pattern, string, flags). This method returns a match object only if the *whole* string matches the pattern, in any other case it returns None.

**Note:** re.fullmatch() was introduced in Python 3.4, before that, re.match() was used instead. On newer versions, fullmatch() is prefered.

Let's compile() the Regular Expression from before, and define a simple function that accepts an email address and uses the expression to validate it:

The re.compile() method compiles a regex pattern into a regex object. It's mostly used for efficiency reasons, when we plan on matching the pattern more than once.

Now, let's test the code on some of the examples we took a look at earlier:



# Machine Learning

[❮ Previous](https://www.w3schools.com/python/matplotlib_pie_charts.asp)[Next ❯](https://www.w3schools.com/python/python_ml_mean_median_mode.asp)

Machine Learning is making the computer learn from studying data and statistics.

Machine Learning is a step into the direction of artificial intelligence (AI).

Machine Learning is a program that analyses data and learns to predict the outcome.

## Where To Start?

In this tutorial we will go back to mathematics and study statistics, and how to calculate important numbers based on data sets.

We will also learn how to use various Python modules to get the answers we need.

And we will learn how to make functions that are able to predict the outcome based on what we have learned.

## Data Set

In the mind of a computer, a data set is any collection of data. It can be anything from an array to a complete database.

Example of an array:

[99,86,87,88,111,86,103,87,94,78,77,85,86]

Example of a database:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Carname | Color | Age | Speed | AutoPass |
| BMW | red | 5 | 99 | Y |
| Volvo | black | 7 | 86 | Y |
| VW | gray | 8 | 87 | N |
| VW | white | 7 | 88 | Y |
| Ford | white | 2 | 111 | Y |
| VW | white | 17 | 86 | Y |
| Tesla | red | 2 | 103 | Y |
| BMW | black | 9 | 87 | Y |
| Volvo | gray | 4 | 94 | N |
| Ford | white | 11 | 78 | N |
| Toyota | gray | 12 | 77 | N |
| VW | white | 9 | 85 | N |
| Toyota | blue | 6 | 86 | Y |

By looking at the array, we can guess that the average value is probably around 80 or 90, and we are also able to determine the highest value and the lowest value, but what else can we do?

And by looking at the database we can see that the most popular color is white, and the oldest car is 17 years, but what if we could predict if a car had an AutoPass, just by looking at the other values?

That is what Machine Learning is for! Analyzing data and predicting the outcome!

In Machine Learning it is common to work with very large data sets. In this tutorial we will try to make it as easy as possible to understand the different concepts of machine learning, and we will work with small easy-to-understand data sets.

ADVERTISEMENT

## Data Types

To analyze data, it is important to know what type of data we are dealing with.

We can split the data types into three main categories:

* **Numerical**
* **Categorical**
* **Ordinal**

**Numerical** data are numbers, and can be split into two numerical categories:

* Discrete Data  
  - numbers that are limited to integers. Example: The number of cars passing by.
* Continuous Data  
  - numbers that are of infinite value. Example: The price of an item, or the size of an item

**Categorical** data are values that cannot be measured up against each other. Example: a color value, or any yes/no values.

**Ordinal** data are like categorical data, but can be measured up against each other. Example: school grades where A is better than B and so on.

By knowing the data type of your data source, you will be able to know what technique to use when analyzing them.

# Machine Learning - Mean Median Mode

[❮ Previous](https://www.w3schools.com/python/python_ml_getting_started.asp)[Next ❯](https://www.w3schools.com/python/python_ml_standard_deviation.asp)

## Mean, Median, and Mode

What can we learn from looking at a group of numbers?

In Machine Learning (and in mathematics) there are often three values that interests us:

* **Mean** - The average value
* **Median** - The mid point value
* **Mode** - The most common value

Example: We have registered the speed of 13 cars:

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

What is the average, the middle, or the most common speed value?

## Mean

The mean value is the average value.

To calculate the mean, find the sum of all values, and divide the sum by the number of values:

(99+86+87+88+111+86+103+87+94+78+77+85+86) / 13 = 89.77

The NumPy module has a method for this. Learn about the NumPy module in our [NumPy Tutorial](https://www.w3schools.com/python/numpy/default.asp).

### Example

Use the NumPy mean() method to find the average speed:

import numpy  
  
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
x = numpy.mean(speed)  
  
print(x)

[Run example »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_mean)

ADVERTISEMENT

## Median

The median value is the value in the middle, after you have sorted all the values:

77, 78, 85, 86, 86, 86, 87, 87, 88, 94, 99, 103, 111

It is important that the numbers are sorted before you can find the median.

The NumPy module has a method for this:

### Example

Use the NumPy median() method to find the middle value:

import numpy  
  
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
x = numpy.median(speed)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_median)

If there are two numbers in the middle, divide the sum of those numbers by two.

77, 78, 85, 86, 86, 86, 87, 87, 94, 98, 99, 103  
  
(86 + 87) / 2 = 86.5

### Example

Using the NumPy module:

import numpy  
  
speed = [99,86,87,88,86,103,87,94,78,77,85,86]  
  
x = numpy.median(speed)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_median2)

## Mode

The Mode value is the value that appears the most number of times:

99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86 = 86

The SciPy module has a method for this. Learn about the SciPy module in our [SciPy Tutorial](https://www.w3schools.com/python/scipy_intro.asp).

### Example

Use the SciPy mode() method to find the number that appears the most:

from scipy import stats  
  
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]  
  
x = stats.mode(speed)  
  
print(x)

Machine Learning - Standard Deviation

[❮ Previous](https://www.w3schools.com/python/python_ml_mean_median_mode.asp)[Next ❯](https://www.w3schools.com/python/python_ml_percentile.asp)

What is Standard Deviation?

Standard deviation is a number that describes how spread out the values are.

A low standard deviation means that most of the numbers are close to the mean (average) value.

A high standard deviation means that the values are spread out over a wider range.

Example: This time we have registered the speed of 7 cars:

speed = [86,87,88,86,87,85,86]

The standard deviation is:

0.9

Meaning that most of the values are within the range of 0.9 from the mean value, which is 86.4.

Let us do the same with a selection of numbers with a wider range:

speed = [32,111,138,28,59,77,97]

The standard deviation is:

37.85

Meaning that most of the values are within the range of 37.85 from the mean value, which is 77.4.

As you can see, a higher standard deviation indicates that the values are spread out over a wider range.

The NumPy module has a method to calculate the standard deviation:

Example

Use the NumPy std() method to find the standard deviation:

import numpy  
  
speed = [86,87,88,86,87,85,86]  
  
x = numpy.std(speed)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_numpy_std)

Example

import numpy  
  
speed = [32,111,138,28,59,77,97]  
  
x = numpy.std(speed)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_numpy_std2)

ADVERTISEMENT

Variance

Variance is another number that indicates how spread out the values are.

In fact, if you take the square root of the variance, you get the standard deviation!

Or the other way around, if you multiply the standard deviation by itself, you get the variance!

To calculate the variance you have to do as follows:

1. Find the mean:

(32+111+138+28+59+77+97) / 7 = 77.4

2. For each value: find the difference from the mean:

 32 - 77.4 = -45.4  
111 - 77.4 =  33.6  
138 - 77.4 =  60.6  
 28 - 77.4 = -49.4  
 59 - 77.4 = -18.4  
 77 - 77.4 = - 0.4  
 97 - 77.4 =  19.6

3. For each difference: find the square value:

(-45.4)2 = 2061.16  
 (33.6)2 = 1128.96  
 (60.6)2 = 3672.36  
(-49.4)2 = 2440.36  
(-18.4)2 =  338.56  
(- 0.4)2 =    0.16  
 (19.6)2 =  384.16

4. The variance is the average number of these squared differences:

(2061.16+1128.96+3672.36+2440.36+338.56+0.16+384.16) / 7 = 1432.2

Luckily, NumPy has a method to calculate the variance:

Example

Use the NumPy var() method to find the variance:

import numpy  
  
speed = [32,111,138,28,59,77,97]  
  
x = numpy.var(speed)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_numpy_var)

Standard Deviation

As we have learned, the formula to find the standard deviation is the square root of the variance:

√1432.25 = 37.85

Or, as in the example from before, use the NumPy to calculate the standard deviation:

Example

Use the NumPy std() method to find the standard deviation:

import numpy  
  
speed = [32,111,138,28,59,77,97]  
  
x = numpy.std(speed)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_numpy_std2)

Symbols

Standard Deviation is often represented by the symbol Sigma: σ

Variance is often represented by the symbol Sigma Square: σ2

Machine Learning - Percentiles

[❮ Previous](https://www.w3schools.com/python/python_ml_standard_deviation.asp)[Next ❯](https://www.w3schools.com/python/python_ml_data_distribution.asp)

What are Percentiles?

Percentiles are used in statistics to give you a number that describes the value that a given percent of the values are lower than.

Example: Let's say we have an array of the ages of all the people that lives in a street.

ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]

What is the 75. percentile? The answer is 43, meaning that 75% of the people are 43 or younger.

The NumPy module has a method for finding the specified percentile:

Example

Use the NumPy percentile() method to find the percentiles:

import numpy  
  
ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]  
  
x = numpy.percentile(ages, 75)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_percentile1)

Example

What is the age that 90% of the people are younger than?

import numpy  
  
ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]  
  
x = numpy.percentile(ages, 90)  
  
print(x)

# Machine Learning - Data Distribution

[❮ Previous](https://www.w3schools.com/python/python_ml_percentile.asp)[Next ❯](https://www.w3schools.com/python/python_ml_normal_data_distribution.asp)

## Data Distribution

Earlier in this tutorial we have worked with very small amounts of data in our examples, just to understand the different concepts.

In the real world, the data sets are much bigger, but it can be difficult to gather real world data, at least at an early stage of a project.

### How Can we Get Big Data Sets?

To create big data sets for testing, we use the Python module NumPy, which comes with a number of methods to create random data sets, of any size.

### Example

Create an array containing 250 random floats between 0 and 5:

import numpy  
  
x = numpy.random.uniform(0.0, 5.0, 250)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_numpy_uniform)

## Histogram

To visualize the data set we can draw a histogram with the data we collected.

We will use the Python module Matplotlib to draw a histogram.

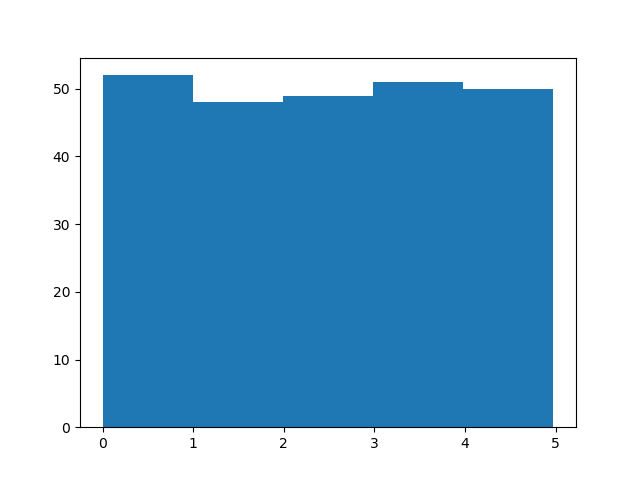
Learn about the Matplotlib module in our [Matplotlib Tutorial](https://www.w3schools.com/python/matplotlib_intro.asp).

### Example

Draw a histogram:

import numpy  
import matplotlib.pyplot as plt  
  
x = numpy.random.uniform(0.0, 5.0, 250)  
  
plt.hist(x, 5)  
plt.show()

### Result:



[Run example »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_numpy_uniform_hist)

### Histogram Explained

We use the array from the example above to draw a histogram with 5 bars.

The first bar represents how many values in the array are between 0 and 1.

The second bar represents how many values are between 1 and 2.

Etc.

Which gives us this result:

* 52 values are between 0 and 1
* 48 values are between 1 and 2
* 49 values are between 2 and 3
* 51 values are between 3 and 4
* 50 values are between 4 and 5

**Note:** The array values are random numbers and will not show the exact same result on your computer.

## Big Data Distributions

An array containing 250 values is not considered very big, but now you know how to create a random set of values, and by changing the parameters, you can create the data set as big as you want.

### Example

Create an array with 100000 random numbers, and display them using a histogram with 100 bars:

import numpy  
import matplotlib.pyplot as plt  
  
x = numpy.random.uniform(0.0, 5.0, 100000)  
  
plt.hist(x, 100)  
plt.show()

Machine Learning - Normal Data Distribution

[❮ Previous](https://www.w3schools.com/python/python_ml_data_distribution.asp)[Next ❯](https://www.w3schools.com/python/python_ml_scatterplot.asp)

Normal Data Distribution

In the previous chapter we learned how to create a completely random array, of a given size, and between two given values.

In this chapter we will learn how to create an array where the values are concentrated around a given value.

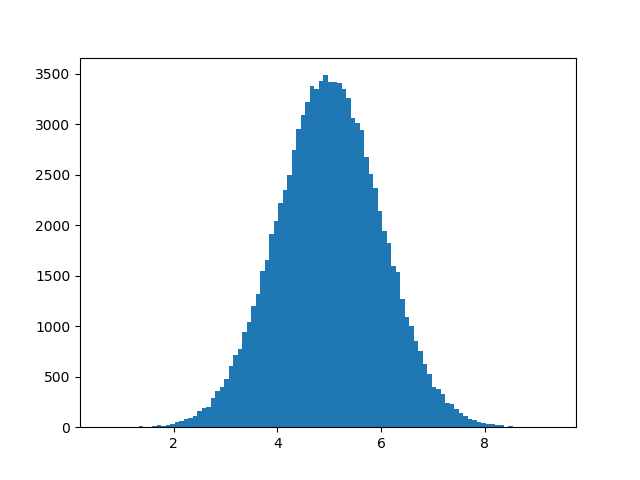
In probability theory this kind of data distribution is known as the *normal data distribution*, or the *Gaussian data distribution*, after the mathematician Carl Friedrich Gauss who came up with the formula of this data distribution.

Example

A typical normal data distribution:

import numpy  
import matplotlib.pyplot as plt  
  
x = numpy.random.normal(5.0, 1.0, 100000)  
  
plt.hist(x, 100)  
plt.show()

Result:



[Run example »](https://www.w3schools.com/python/trypython.asp?filename=demo_ml_numpy_normal_hist)

**Note:** A normal distribution graph is also known as the *bell curve* because of it's characteristic shape of a bell.

Histogram Explained

We use the array from the numpy.random.normal() method, with 100000 values,  to draw a histogram with 100 bars.

We specify that the mean value is 5.0, and the standard deviation is 1.0.

Meaning that the values should be concentrated around 5.0, and rarely further away than 1.0 from the mean.

And as you can see from the histogram, most values are between 4.0 and 6.0, with a top at approximately 5.0.

# Python - GUI Programming (Tkinter)

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

* **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.
* **wxPython** − This is an open-source Python interface for wxWindows [http://wxpython.org](http://wxpython.org/).
* **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine [http://www.jython.org](http://www.jython.org/).

There are many other interfaces available, which you can find them on the net.

## Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the *Tkinter* module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

## Example

#!/usr/bin/python

import Tkinter

top = Tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()

This would create a following window −



## Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | [Button](https://www.tutorialspoint.com/python/tk_button.htm)  The Button widget is used to display buttons in your application. |
| 2 | [Canvas](https://www.tutorialspoint.com/python/tk_canvas.htm)  The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application. |
| 3 | [Checkbutton](https://www.tutorialspoint.com/python/tk_checkbutton.htm)  The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| 4 | [Entry](https://www.tutorialspoint.com/python/tk_entry.htm)  The Entry widget is used to display a single-line text field for accepting values from a user. |
| 5 | [Frame](https://www.tutorialspoint.com/python/tk_frame.htm)  The Frame widget is used as a container widget to organize other widgets. |
| 6 | [Label](https://www.tutorialspoint.com/python/tk_label.htm)  The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| 7 | [Listbox](https://www.tutorialspoint.com/python/tk_listbox.htm)  The Listbox widget is used to provide a list of options to a user. |
| 8 | [Menubutton](https://www.tutorialspoint.com/python/tk_menubutton.htm)  The Menubutton widget is used to display menus in your application. |
| 9 | [Menu](https://www.tutorialspoint.com/python/tk_menu.htm)  The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. |
| 10 | [Message](https://www.tutorialspoint.com/python/tk_message.htm)  The Message widget is used to display multiline text fields for accepting values from a user. |
| 11 | [Radiobutton](https://www.tutorialspoint.com/python/tk_radiobutton.htm)  The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| 12 | [Scale](https://www.tutorialspoint.com/python/tk_scale.htm)  The Scale widget is used to provide a slider widget. |
| 13 | [Scrollbar](https://www.tutorialspoint.com/python/tk_scrollbar.htm)  The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes. |
| 14 | [Text](https://www.tutorialspoint.com/python/tk_text.htm)  The Text widget is used to display text in multiple lines. |
| 15 | [Toplevel](https://www.tutorialspoint.com/python/tk_toplevel.htm)  The Toplevel widget is used to provide a separate window container. |
| 16 | [Spinbox](https://www.tutorialspoint.com/python/tk_spinbox.htm)  The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values. |
| 17 | [PanedWindow](https://www.tutorialspoint.com/python/tk_panedwindow.htm)  A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically. |
| 18 | [LabelFrame](https://www.tutorialspoint.com/python/tk_labelframe.htm)  A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. |
| 19 | [tkMessageBox](https://www.tutorialspoint.com/python/tk_messagebox.htm)  This module is used to display message boxes in your applications. |

Let us study these widgets in detail −

## Standard attributes

Let us take a look at how some of their common attributes.such as sizes, colors and fonts are specified.

* [Dimensions](https://www.tutorialspoint.com/python/tk_dimensions.htm)
* [Colors](https://www.tutorialspoint.com/python/tk_colors.htm)
* [Fonts](https://www.tutorialspoint.com/python/tk_fonts.htm)
* [Anchors](https://www.tutorialspoint.com/python/tk_anchors.htm)
* [Relief styles](https://www.tutorialspoint.com/python/tk_relief.htm)
* [Bitmaps](https://www.tutorialspoint.com/python/tk_bitmaps.htm)
* [Cursors](https://www.tutorialspoint.com/python/tk_cursors.htm)

Let us study them briefly −

## Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

* [The *pack()* Method](https://www.tutorialspoint.com/python/tk_pack.htm) − This geometry manager organizes widgets in blocks before placing them in the parent widget.
* [The *grid()* Method](https://www.tutorialspoint.com/python/tk_grid.htm) − This geometry manager organizes widgets in a table-like structure in the parent widget.
* [The *place()* Method](https://www.tutorialspoint.com/python/tk_place.htm) − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

Button

The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

## Syntax

Here is the simple syntax to create this widget −

w = Button ( master, option=value, ... )

## Parameters

* **master** − This represents the parent window.
* **options** − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **activebackground**  Background color when the button is under the cursor. |
| 2 | **activeforeground**  Foreground color when the button is under the cursor. |
| 3 | **Bd**  Border width in pixels. Default is 2. |
| 4 | **Bg**  Normal background color. |
| 5 | **command**  Function or method to be called when the button is clicked. |
| 6 | **Fg**  Normal foreground (text) color. |
| 7 | **Font**  Text font to be used for the button's label. |
| 8 | **height**  Height of the button in text lines (for textual buttons) or pixels (for images). |
| 9 | **highlightcolor**  The color of the focus highlight when the widget has focus. |
| 10 | **Image**  Image to be displayed on the button (instead of text). |
| 11 | **Justify**  How to show multiple text lines: LEFT to left-justify each line; CENTER to center them; or RIGHT to right-justify. |
| 12 | **Padx**  Additional padding left and right of the text. |
| 13 | **Pady**  Additional padding above and below the text. |
| 14 | **Relief**  Relief specifies the type of the border. Some of the values are SUNKEN, RAISED, GROOVE, and RIDGE. |
| 15 | **State**  Set this option to DISABLED to gray out the button and make it unresponsive. Has the value ACTIVE when the mouse is over it. Default is NORMAL. |
| 16 | **underline**  Default is -1, meaning that no character of the text on the button will be underlined. If nonnegative, the corresponding text character will be underlined. |
| 17 | **Width**  Width of the button in letters (if displaying text) or pixels (if displaying an image). |
| 18 | **Wraplength**  If this value is set to a positive number, the text lines will be wrapped to fit within this length. |

## Methods

Following are commonly used methods for this widget −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **flash()**  Causes the button to flash several times between active and normal colors. Leaves the button in the state it was in originally. Ignored if the button is disabled. |
| 2 | **invoke()**  Calls the button's callback, and returns what that function returns. Has no effect if the button is disabled or there is no callback. |

## Example

Try the following example yourself −

import Tkinter

import tkMessageBox

top = Tkinter.Tk()

def helloCallBack():

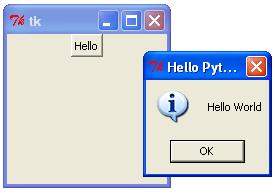
tkMessageBox.showinfo( "Hello Python", "Hello World")

B = Tkinter.Button(top, text ="Hello", command = helloCallBack)

B.pack()

top.mainloop()

When the above code is executed, it produces the following result −



Radio Button

This widget implements a multiple-choice button, which is a way to offer many possible selections to the user and lets user choose only one of them.

In order to implement this functionality, each group of radiobuttons must be associated to the same variable and each one of the buttons must symbolize a single value. You can use the Tab key to switch from one radionbutton to another.

## Syntax

Here is the simple syntax to create this widget −

w = Radiobutton ( master, option, ... )

## Parameters

* **master** − This represents the parent window.
* **options** − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **Activebackground**  The background color when the mouse is over the radiobutton. |
| 2 | **Activeforeground**  The foreground color when the mouse is over the radiobutton. |
| 3 | **Anchor**  If the widget inhabits a space larger than it needs, this option specifies where the radiobutton will sit in that space. The default is anchor=CENTER. |
| 4 | **Bg**  The normal background color behind the indicator and label. |
| 5 | **Bitmap**  To display a monochrome image on a radiobutton, set this option to a bitmap. |
| 6 | **Borderwidth**  The size of the border around the indicator part itself. Default is 2 pixels. |
| 7 | **Command**  A procedure to be called every time the user changes the state of this radiobutton. |
| 8 | **Cursor**  If you set this option to a cursor name (*arrow, dot etc.*), the mouse cursor will change to that pattern when it is over the radiobutton. |
| 9 | **Font**  The font used for the text. |
| 10 | **Fg**  The color used to render the text. |
| 11 | **Height**  The number of lines (not pixels) of text on the radiobutton. Default is 1. |
| 12 | **Highlightbackground**  The color of the focus highlight when the radiobutton does not have focus. |
| 13 | **Highlightcolor**  The color of the focus highlight when the radiobutton has the focus. |
| 14 | **Image**  To display a graphic image instead of text for this radiobutton, set this option to an image object. |
| 15 | **Justify**  If the text contains multiple lines, this option controls how the text is justified: CENTER (the default), LEFT, or RIGHT. |
| 16 | **Padx**  How much space to leave to the left and right of the radiobutton and text. Default is 1. |
| 17 | **Pady**  How much space to leave above and below the radiobutton and text. Default is 1. |
| 18 | **Relief**  Specifies the appearance of a decorative border around the label. The default is FLAT; for other values. |
| 19 | **Selectcolor**  The color of the radiobutton when it is set. Default is red. |
| 20 | **Selectimage**  If you are using the image option to display a graphic instead of text when the radiobutton is cleared, you can set the selectimage option to a different image that will be displayed when the radiobutton is set. |
| 21 | **State**  The default is state=NORMAL, but you can set state=DISABLED to gray out the control and make it unresponsive. If the cursor is currently over the radiobutton, the state is ACTIVE. |
| 22 | **Text**  The label displayed next to the radiobutton. Use newlines ("\n") to display multiple lines of text. |
| 23 | **Textvariable**  To slave the text displayed in a label widget to a control variable of class *StringVar*, set this option to that variable. |
| 24 | **Underline**  You can display an underline (\_) below the nth letter of the text, counting from 0, by setting this option to n. The default is underline=-1, which means no underlining. |
| 25 | **Value**  When a radiobutton is turned on by the user, its control variable is set to its current value option. If the control variable is an *IntVar*, give each radiobutton in the group a different integer value option. If the control variable is a *StringVar*, give each radiobutton a different string value option. |
| 26 | **Variable**  The control variable that this radiobutton shares with the other radiobuttons in the group. This can be either an IntVar or a StringVar. |
| 27 | **Width**  Width of the label in characters (not pixels!). If this option is not set, the label will be sized to fit its contents. |
| 28 | **Wraplength**  You can limit the number of characters in each line by setting this option to the desired number. The default value, 0, means that lines will be broken only at newlines. |

## Methods

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **deselect()**  Clears (turns off) the radiobutton. |
| 2 | **flash()**  Flashes the radiobutton a few times between its active and normal colors, but leaves it the way it started. |
| 3 | **invoke()**  You can call this method to get the same actions that would occur if the user clicked on the radiobutton to change its state. |
| 4 | **select()**  Sets (turns on) the radiobutton. |

## Example

Try the following example yourself −

from Tkinter import \*

def sel():

selection = "You selected the option " + str(var.get())

label.config(text = selection)

root = Tk()

var = IntVar()

R1 = Radiobutton(root, text="Option 1", variable=var, value=1,

command=sel)

R1.pack( anchor = W )

R2 = Radiobutton(root, text="Option 2", variable=var, value=2,

command=sel)

R2.pack( anchor = W )

R3 = Radiobutton(root, text="Option 3", variable=var, value=3,

command=sel)

R3.pack( anchor = W)

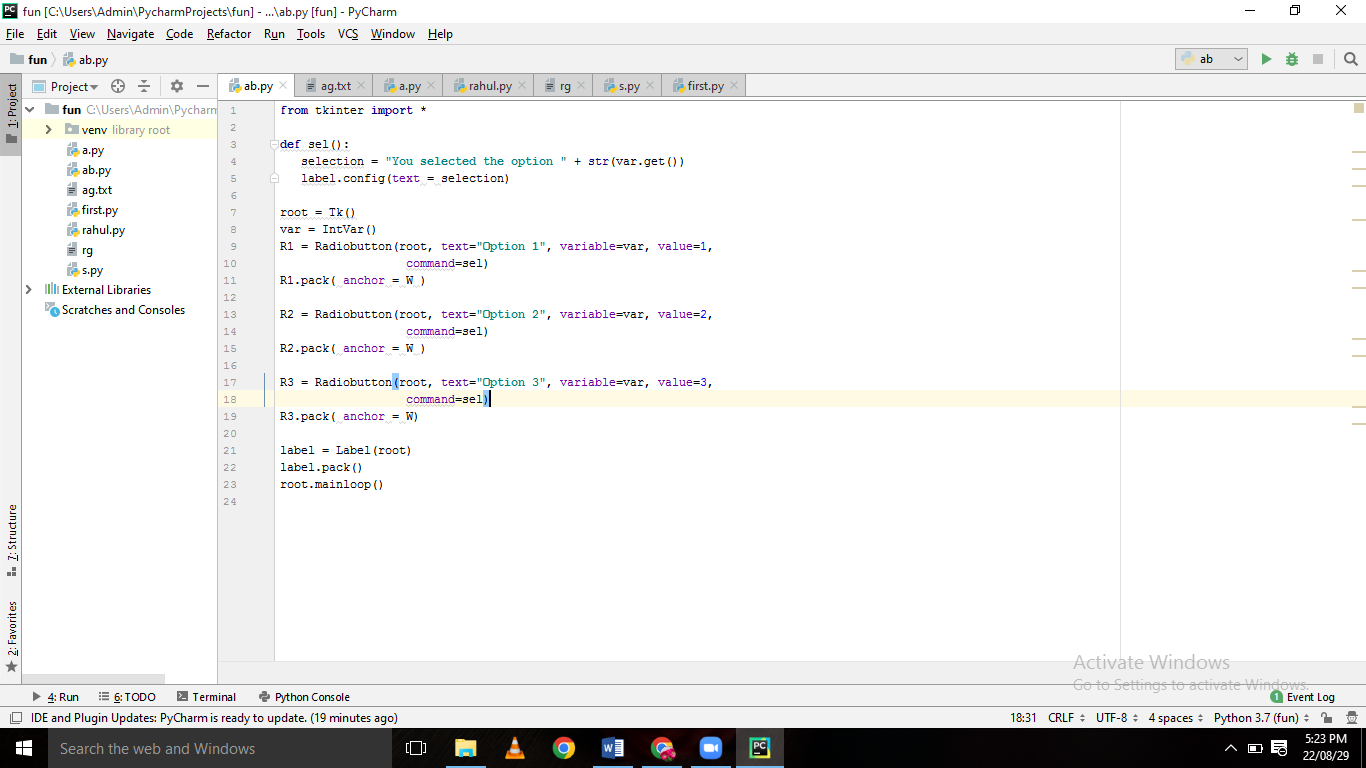
label = Label(root)

label.pack()

root.mainloop()

When the above code is executed, it produces the following result −





CheckBoxes

The Checkbutton widget is used to display a number of options to a user as toggle buttons. The user can then select one or more options by clicking the button corresponding to each option.

You can also display images in place of text.

## Syntax

Here is the simple syntax to create this widget −

w = Checkbutton ( master, option, ... )

## Parameters

* **master** − This represents the parent window.
* **options** − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **Activebackground**  Background color when the checkbutton is under the cursor. |
| 2 | **Activeforeground**  Foreground color when the checkbutton is under the cursor. |
| 3 | **Bg**  The normal background color displayed behind the label and indicator. |
| 4 | **Bitmap**  To display a monochrome image on a button. |
| 5 | **Bd**  The size of the border around the indicator. Default is 2 pixels. |
| 6 | **Command**  A procedure to be called every time the user changes the state of this checkbutton. |
| 7 | **Cursor**  If you set this option to a cursor name (*arrow, dot etc.*), the mouse cursor will change to that pattern when it is over the checkbutton. |
| 8 | **Disabledforeground**  The foreground color used to render the text of a disabled checkbutton. The default is a stippled version of the default foreground color. |
| 9 | **Font**  The font used for the text. |
| 10 | **Fg**  The color used to render the text. |
| 11 | **Height**  The number of lines of text on the checkbutton. Default is 1. |
| 12 | **Highlightcolor**  The color of the focus highlight when the checkbutton has the focus. |
| 13 | **Image**  To display a graphic image on the button. |
| 14 | **Justify**  If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT. |
| 15 | **Offvalue**  Normally, a checkbutton's associated control variable will be set to 0 when it is cleared (off). You can supply an alternate value for the off state by setting offvalue to that value. |
| 16 | **Onvalue**  Normally, a checkbutton's associated control variable will be set to 1 when it is set (on). You can supply an alternate value for the on state by setting onvalue to that value. |
| 17 | **Padx**  How much space to leave to the left and right of the checkbutton and text. Default is 1 pixel. |
| 18 | **Pady**  How much space to leave above and below the checkbutton and text. Default is 1 pixel. |
| 19 | **Relief**  With the default value, relief=FLAT, the checkbutton does not stand out from its background. You may set this option to any of the other styles |
| 20 | **Selectcolor**  The color of the checkbutton when it is set. Default is selectcolor="red". |
| 21 | **Selectimage**  If you set this option to an image, that image will appear in the checkbutton when it is set. |
| 22 | **State**  The default is state=NORMAL, but you can use state=DISABLED to gray out the control and make it unresponsive. If the cursor is currently over the checkbutton, the state is ACTIVE. |
| 23 | **Text**  The label displayed next to the checkbutton. Use newlines ("\n") to display multiple lines of text. |
| 24 | **Underline**  With the default value of -1, none of the characters of the text label are underlined. Set this option to the index of a character in the text (counting from zero) to underline that character. |
| 25 | **Variable**  The control variable that tracks the current state of the checkbutton. Normally this variable is an *IntVar*, and 0 means cleared and 1 means set, but see the offvalue and onvalue options above. |
| 26 | **Width**  The default width of a checkbutton is determined by the size of the displayed image or text. You can set this option to a number of characters and the checkbutton will always have room for that many characters. |
| 27 | **Wraplength**  Normally, lines are not wrapped. You can set this option to a number of characters and all lines will be broken into pieces no longer than that number. |

## Methods

Following are commonly used methods for this widget −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **deselect()**  Clears (turns off) the checkbutton. |
| 2 | **flash()**  Flashes the checkbutton a few times between its active and normal colors, but leaves it the way it started. |
| 3 | **invoke()**  You can call this method to get the same actions that would occur if the user clicked on the checkbutton to change its state. |
| 4 | **select()**  Sets (turns on) the checkbutton. |
| 5 | **toggle()**  Clears the checkbutton if set, sets it if cleared. |

## Example

Try the following example yourself −

from Tkinter import \*

import tkMessageBox

import Tkinter

top = Tkinter.Tk()

CheckVar1 = IntVar()

CheckVar2 = IntVar()

C1 = Checkbutton(top, text = "Music", variable = CheckVar1, \

onvalue = 1, offvalue = 0, height=5, \

width = 20)

C2 = Checkbutton(top, text = "Video", variable = CheckVar2, \

onvalue = 1, offvalue = 0, height=5, \

width = 20)

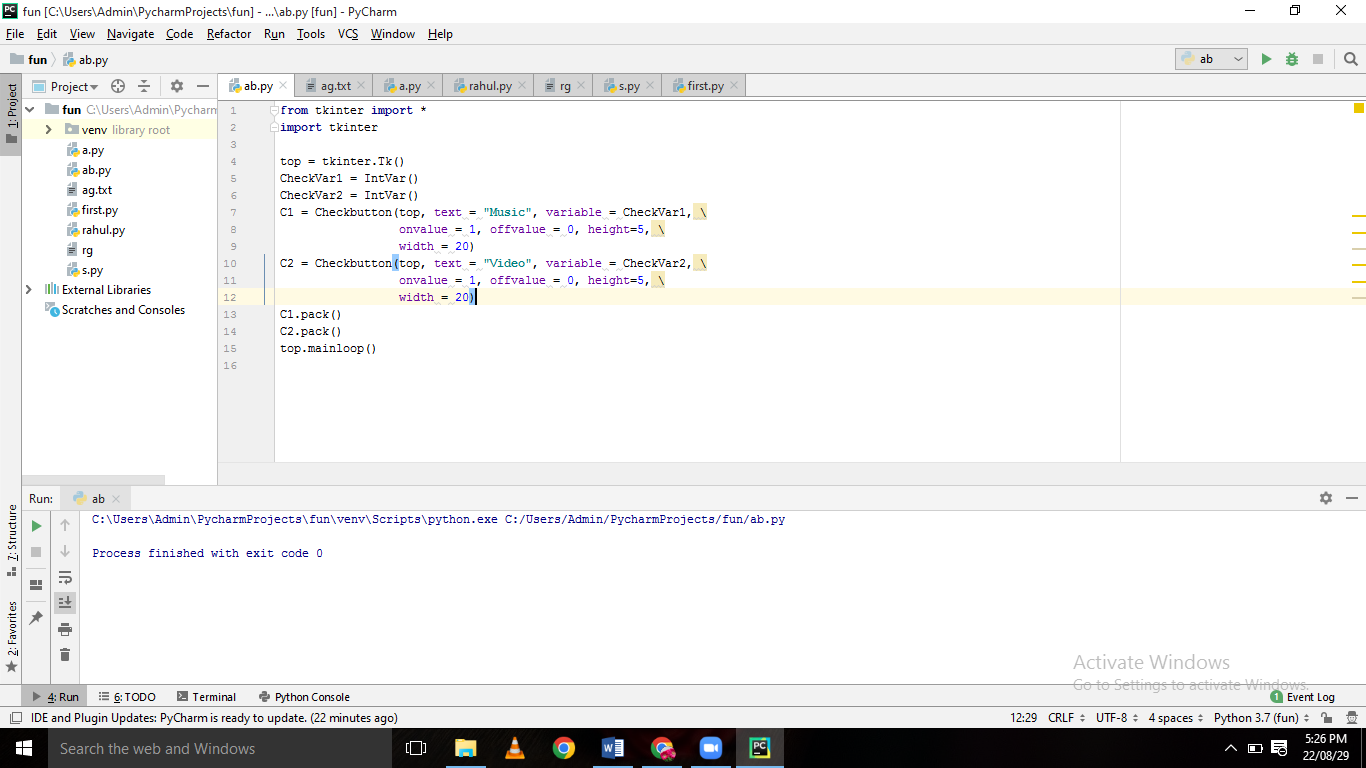
C1.pack()

C2.pack()

top.mainloop()

When the above code is executed, it produces the following result −





Canvas

The Canvas is a rectangular area intended for drawing pictures or other complex layouts. You can place graphics, text, widgets or frames on a Canvas.

## Syntax

Here is the simple syntax to create this widget −

w = Canvas ( master, option=value, ... )

## Parameters

* **master** − This represents the parent window.
* **options** − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **Bd**  Border width in pixels. Default is 2. |
| 2 | **Bg**  Normal background color. |
| 3 | **Confine**  If true (the default), the canvas cannot be scrolled outside of the scrollregion. |
| 4 | **Cursor**  Cursor used in the canvas like *arrow, circle, dot etc.* |
| 5 | **Height**  Size of the canvas in the Y dimension. |
| 6 | **Highlightcolor**  Color shown in the focus highlight. |
| 7 | **Relief**  Relief specifies the type of the border. Some of the values are SUNKEN, RAISED, GROOVE, and RIDGE. |
| 8 | **Scrollregion**  A tuple (w, n, e, s) that defines over how large an area the canvas can be scrolled, where w is the left side, n the top, e the right side, and s the bottom. |
| 9 | **Width**  Size of the canvas in the X dimension. |
| 10 | **Xscrollincrement**  If you set this option to some positive dimension, the canvas can be positioned only on multiples of that distance, and the value will be used for scrolling by scrolling units, such as when the user clicks on the arrows at the ends of a scrollbar. |
| 11 | **Xscrollcommand**  If the canvas is scrollable, this attribute should be the .set() method of the horizontal scrollbar. |
| 12 | **Yscrollincrement**  Works like xscrollincrement, but governs vertical movement. |
| 13 | **Yscrollcommand**  If the canvas is scrollable, this attribute should be the .set() method of the vertical scrollbar. |

The Canvas widget can support the following standard items −

**arc** − Creates an arc item, which can be a chord, a pieslice or a simple arc.

coord = 10, 50, 240, 210

arc = canvas.create\_arc(coord, start=0, extent=150, fill="blue")

**image** − Creates an image item, which can be an instance of either the BitmapImage or the PhotoImage classes.

filename = PhotoImage(file = "sunshine.gif")

image = canvas.create\_image(50, 50, anchor=NE, image=filename)

**line** − Creates a line item.

line = canvas.create\_line(x0, y0, x1, y1, ..., xn, yn, options)

**oval** − Creates a circle or an ellipse at the given coordinates. It takes two pairs of coordinates; the top left and bottom right corners of the bounding rectangle for the oval.

oval = canvas.create\_oval(x0, y0, x1, y1, options)

**polygon** − Creates a polygon item that must have at least three vertices.

oval = canvas.create\_polygon(x0, y0, x1, y1,...xn, yn, options)

## Example

Try the following example yourself −

import Tkinter

top = Tkinter.Tk()

C = Tkinter.Canvas(top, bg="blue", height=250, width=300)

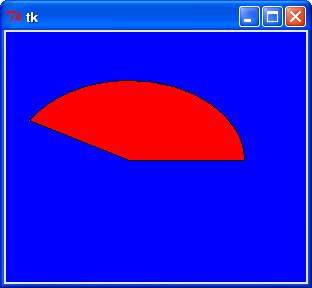
coord = 10, 50, 240, 210

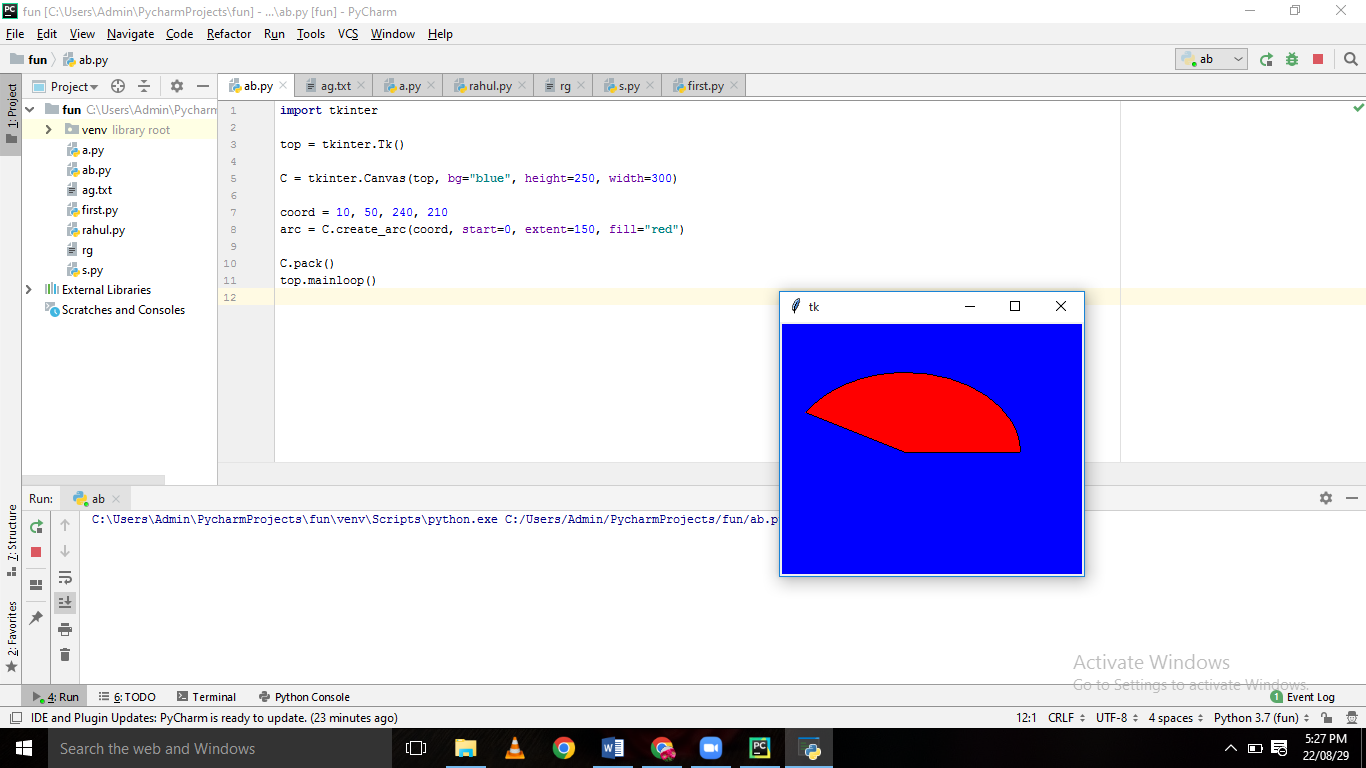
arc = C.create\_arc(coord, start=0, extent=150, fill="red")

C.pack()

top.mainloop()

When the above code is executed, it produces the following result −





Entry

The Entry widget is used to accept single-line text strings from a user.

* If you want to display multiple lines of text that can be edited, then you should use the *Text* widget.
* If you want to display one or more lines of text that cannot be modified by the user, then you should use the *Label* widget.

## Syntax

Here is the simple syntax to create this widget −

w = Entry( master, option, ... )

## Parameters

* **master** − This represents the parent window.
* **options** − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **Bg**  The normal background color displayed behind the label and indicator. |
| 2 | **Bd**  The size of the border around the indicator. Default is 2 pixels. |
| 3 | **Command**  A procedure to be called every time the user changes the state of this checkbutton. |
| 4 | **Cursor**  If you set this option to a cursor name (*arrow, dot etc.*), the mouse cursor will change to that pattern when it is over the checkbutton. |
| 5 | **Font**  The font used for the text. |
| 6 | **Exportselection**  By default, if you select text within an Entry widget, it is automatically exported to the clipboard. To avoid this exportation, use exportselection=0. |
| 7 | **Fg**  The color used to render the text. |
| 8 | **Highlightcolor**  The color of the focus highlight when the checkbutton has the focus. |
| 9 | **Justify**  If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT. |
| 10 | **Relief**  With the default value, relief=FLAT, the checkbutton does not stand out from its background. You may set this option to any of the other styles |
| 11 | **Selectbackground**  The background color to use displaying selected text. |
| 12 | **Selectborderwidth**  The width of the border to use around selected text. The default is one pixel. |
| 13 | **Selectforeground**  The foreground (text) color of selected text. |
| 14 | **Show**  Normally, the characters that the user types appear in the entry. To make a .password. entry that echoes each character as an asterisk, set show="\*". |
| 15 | **State**  The default is state=NORMAL, but you can use state=DISABLED to gray out the control and make it unresponsive. If the cursor is currently over the checkbutton, the state is ACTIVE. |
| 16 | **Textvariable**  In order to be able to retrieve the current text from your entry widget, you must set this option to an instance of the StringVar class. |
| 17 | **Width**  The default width of a checkbutton is determined by the size of the displayed image or text. You can set this option to a number of characters and the checkbutton will always have room for that many characters. |
| 18 | **Xscrollcommand**  If you expect that users will often enter more text than the onscreen size of the widget, you can link your entry widget to a scrollbar. |

## Methods

Following are commonly used methods for this widget −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **delete ( first, last=None )**  Deletes characters from the widget, starting with the one at index first, up to but not including the character at position last. If the second argument is omitted, only the single character at position first is deleted. |
| 2 | **get()**  Returns the entry's current text as a string. |
| 3 | **icursor ( index )**  Set the insertion cursor just before the character at the given index. |
| 4 | **index ( index )**  Shift the contents of the entry so that the character at the given index is the leftmost visible character. Has no effect if the text fits entirely within the entry. |
| 5 | **insert ( index, s )**  Inserts string s before the character at the given index. |
| 6 | **select\_adjust ( index )**  This method is used to make sure that the selection includes the character at the specified index. |
| 7 | **select\_clear()**  Clears the selection. If there isn't currently a selection, has no effect. |
| 8 | **select\_from ( index )**  Sets the ANCHOR index position to the character selected by index, and selects that character. |
| 9 | **select\_present()**  If there is a selection, returns true, else returns false. |
| 10 | **select\_range ( start, end )**  Sets the selection under program control. Selects the text starting at the start index, up to but not including the character at the end index. The start position must be before the end position. |
| 11 | **select\_to ( index )**  Selects all the text from the ANCHOR position up to but not including the character at the given index. |
| 12 | **xview ( index )**  This method is useful in linking the Entry widget to a horizontal scrollbar. |
| 13 | **xview\_scroll ( number, what )**  Used to scroll the entry horizontally. The what argument must be either UNITS, to scroll by character widths, or PAGES, to scroll by chunks the size of the entry widget. The number is positive to scroll left to right, negative to scroll right to left. |

## Example

Try the following example yourself −

from Tkinter import \*

top = Tk()

L1 = Label(top, text="User Name")

L1.pack( side = LEFT)

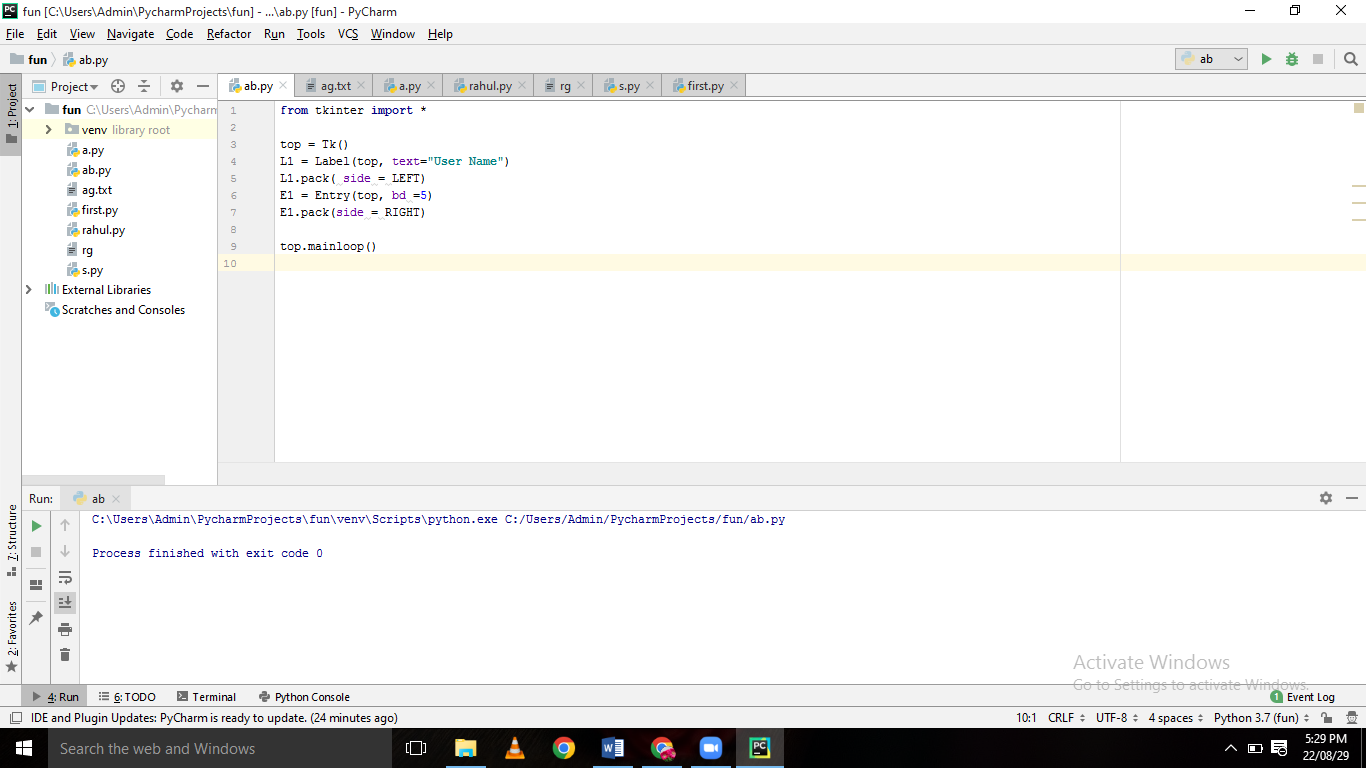
E1 = Entry(top, bd =5)

E1.pack(side = RIGHT)

top.mainloop()

When the above code is executed, it produces the following result −

TK Entry



Label

This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.

It is also possible to underline part of the text (like to identify a keyboard shortcut) and span the text across multiple lines.

## Syntax

Here is the simple syntax to create this widget −

w = Label ( master, option, ... )

## Parameters

* **master** − This represents the parent window.
* **options** − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **Anchor**  This options controls where the text is positioned if the widget has more space than the text needs. The default is anchor=CENTER, which centers the text in the available space. |
| 2 | **Bg**  The normal background color displayed behind the label and indicator. |
| 3 | **Bitmap**  Set this option equal to a bitmap or image object and the label will display that graphic. |
| 4 | **Bd**  The size of the border around the indicator. Default is 2 pixels. |
| 5 | **Cursor**  If you set this option to a cursor name (*arrow, dot etc.*), the mouse cursor will change to that pattern when it is over the checkbutton. |
| 6 | **Font**  If you are displaying text in this label (with the text or textvariable option, the font option specifies in what font that text will be displayed. |
| 7 | **Fg**  If you are displaying text or a bitmap in this label, this option specifies the color of the text. If you are displaying a bitmap, this is the color that will appear at the position of the 1-bits in the bitmap. |
| 8 | **Height**  The vertical dimension of the new frame. |
| 9 | **Image**  To display a static image in the label widget, set this option to an image object. |
| 10 | **Justify**  Specifies how multiple lines of text will be aligned with respect to each other: LEFT for flush left, CENTER for centered (the default), or RIGHT for right-justified. |
| 11 | **Padx**  Extra space added to the left and right of the text within the widget. Default is 1. |
| 12 | **Pady**  Extra space added above and below the text within the widget. Default is 1. |
| 13 | **Relief**  Specifies the appearance of a decorative border around the label. The default is FLAT; for other values. |
| 14 | **Text**  To display one or more lines of text in a label widget, set this option to a string containing the text. Internal newlines ("\n") will force a line break. |
| 15 | **Textvariable**  To slave the text displayed in a label widget to a control variable of class *StringVar*, set this option to that variable. |
| 16 | **Underline**  You can display an underline (\_) below the nth letter of the text, counting from 0, by setting this option to n. The default is underline=-1, which means no underlining. |
| 17 | **Width**  Width of the label in characters (not pixels!). If this option is not set, the label will be sized to fit its contents. |
| 18 | **Wraplength**  You can limit the number of characters in each line by setting this option to the desired number. The default value, 0, means that lines will be broken only at newlines. |

## Example

Try the following example yourself −

from Tkinter import \*

root = Tk()

var = StringVar()

label = Label( root, textvariable=var, relief=RAISED )

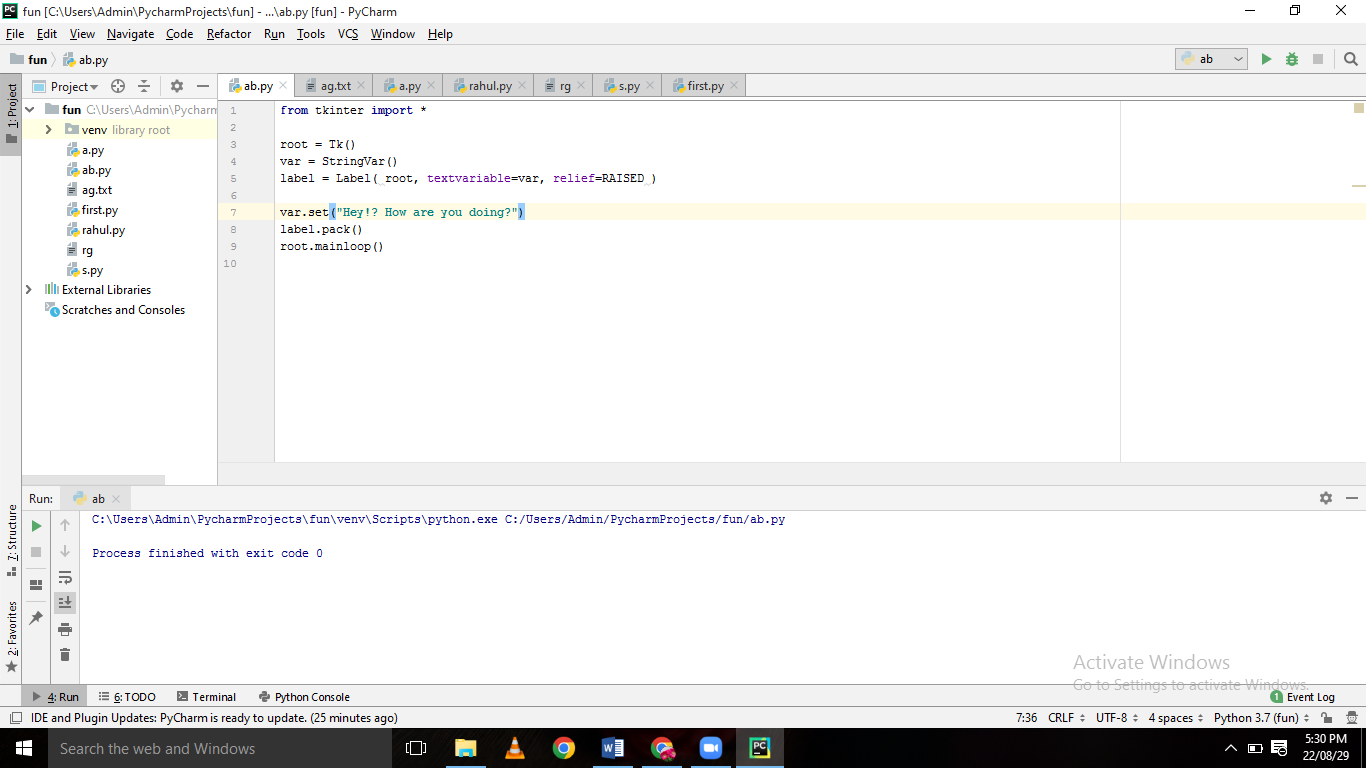
var.set("Hey!? How are you doing?")

label.pack()

root.mainloop()

When the above code is executed, it produces the following result −

TK Label



Frame

The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way. It works like a container, which is responsible for arranging the position of other widgets.

It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets. A frame can also be used as a foundation class to implement complex widgets.

## Syntax

Here is the simple syntax to create this widget −

w = Frame ( master, option, ... )

## Parameters

* **master** − This represents the parent window.
* **options** − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

|  |  |
| --- | --- |
| **Sr.No.** | **Option & Description** |
| 1 | **Bg**  The normal background color displayed behind the label and indicator. |
| 2 | **Bd**  The size of the border around the indicator. Default is 2 pixels. |
| 3 | **Cursor**  If you set this option to a cursor name (*arrow, dot etc.*), the mouse cursor will change to that pattern when it is over the checkbutton. |
| 4 | **Height**  The vertical dimension of the new frame. |
| 5 | **Highlightbackground**  Color of the focus highlight when the frame does not have focus. |
| 6 | **Highlightcolor**  Color shown in the focus highlight when the frame has the focus. |
| 7 | **Highlightthickness**  Thickness of the focus highlight. |
| 8 | **Relief**  With the default value, relief=FLAT, the checkbutton does not stand out from its background. You may set this option to any of the other styles |
| 9 | **Width**  The default width of a checkbutton is determined by the size of the displayed image or text. You can set this option to a number of characters and the checkbutton will always have room for that many characters. |

## Example

Try the following example yourself −

from Tkinter import \*

root = Tk()

frame = Frame(root)

frame.pack()

bottomframe = Frame(root)

bottomframe.pack( side = BOTTOM )

redbutton = Button(frame, text="Red", fg="red")

redbutton.pack( side = LEFT)

greenbutton = Button(frame, text="Brown", fg="brown")

greenbutton.pack( side = LEFT )

bluebutton = Button(frame, text="Blue", fg="blue")

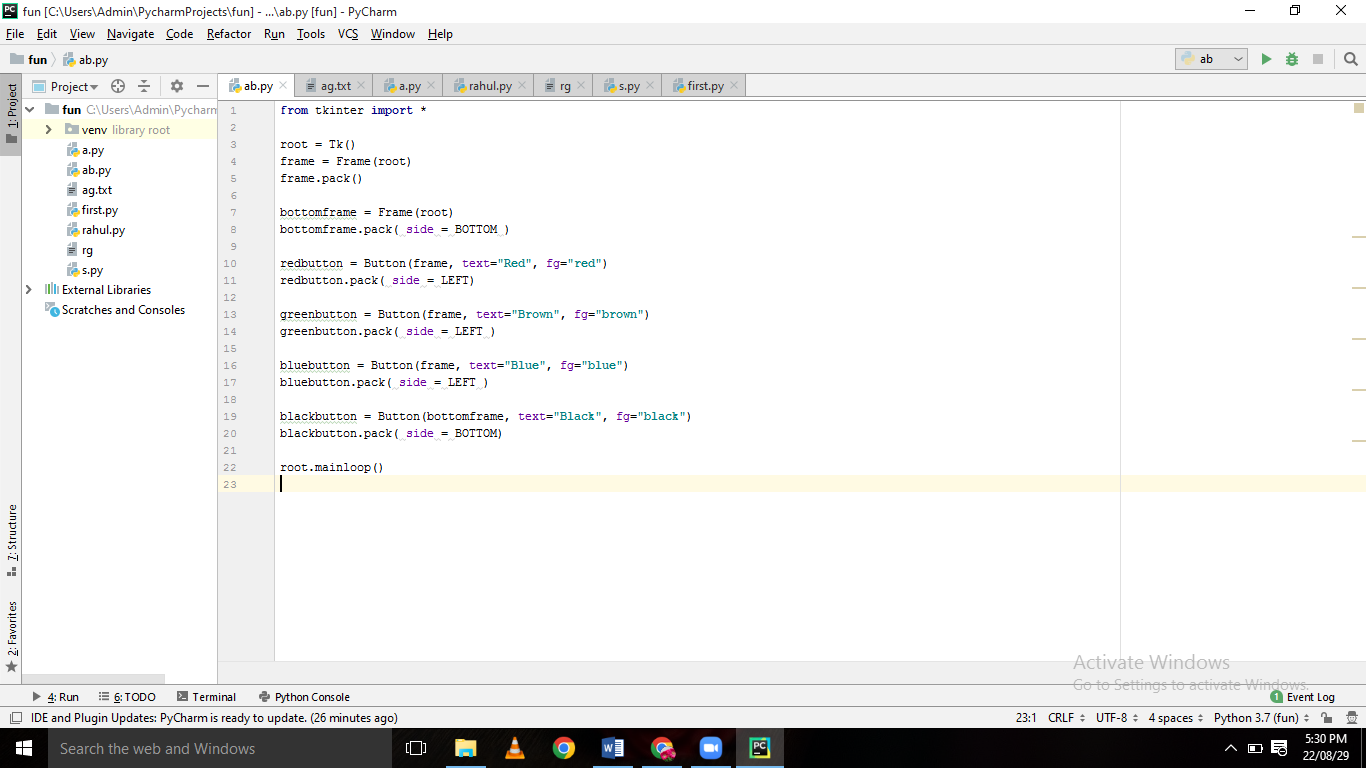
bluebutton.pack( side = LEFT )

blackbutton = Button(bottomframe, text="Black", fg="black")

blackbutton.pack( side = BOTTOM)

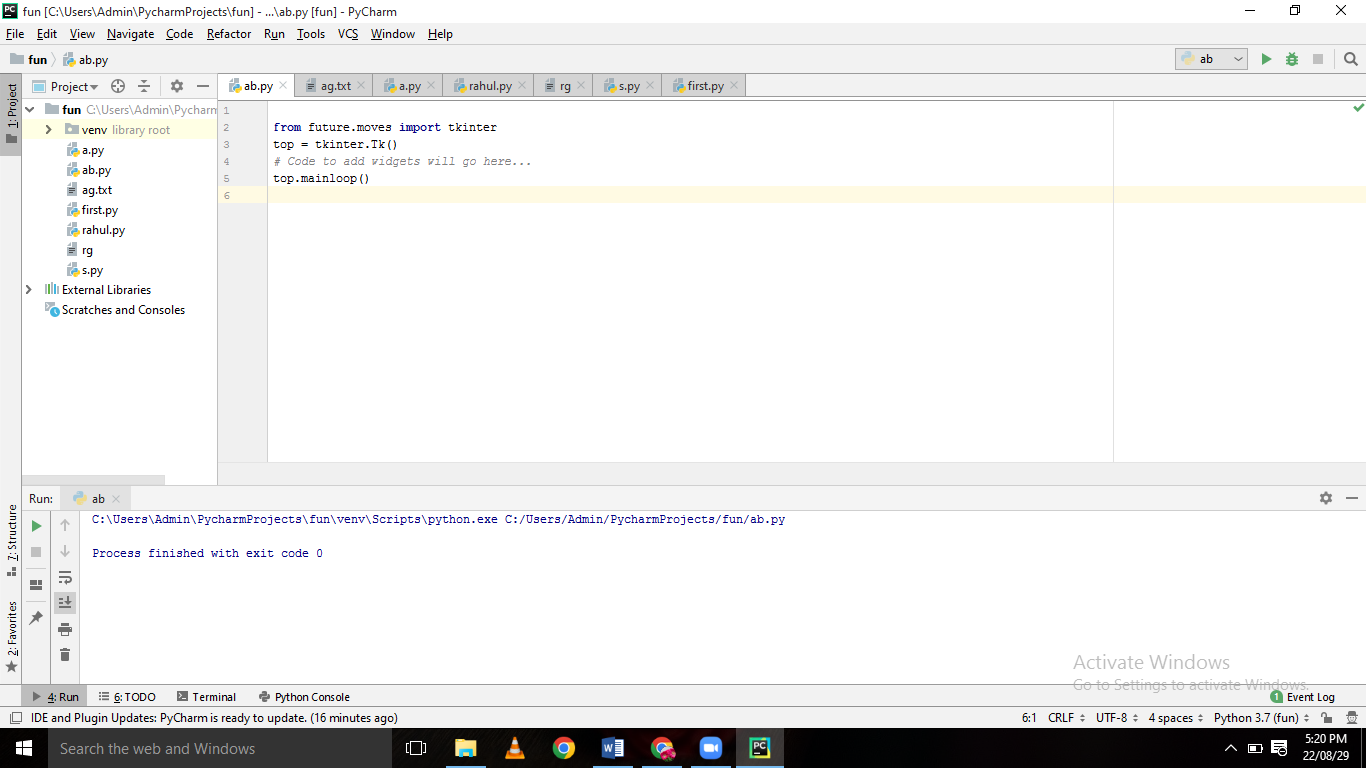
root.mainloop()

When the above code is executed, it produces the following result –





Program 1



Program 2

