# Python Condition and Control flow Statement

**By Md Saif Hossain**

Saif.ndub@gmail.com

# Python Conditions & Control Flow

**Python Conditions and If statements:** Python supports the usual logical conditions from mathematics:

I. Equals: a == b
II. Not Equals: a != b
III. Less than: a < b
IV. Less than or equal to: a <= b
V. Greater than: a > b
VI. Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and "loops".

```
a = 33
b = 200
if b > a:
   print("b is greater than a")
```

# Control Flow Statements

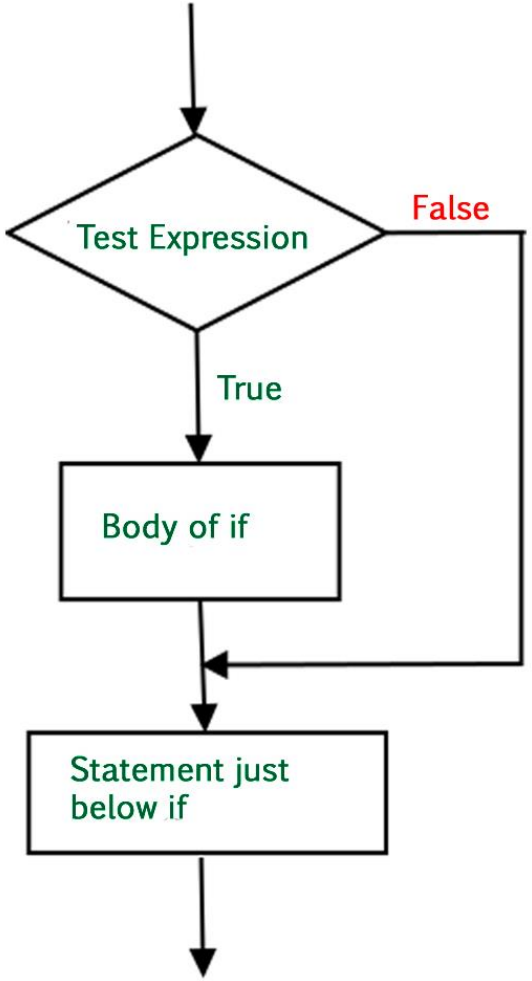
## If statement

If statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not. An "if statement" is written by using the if keyword. We can use if statement in many ways like as:

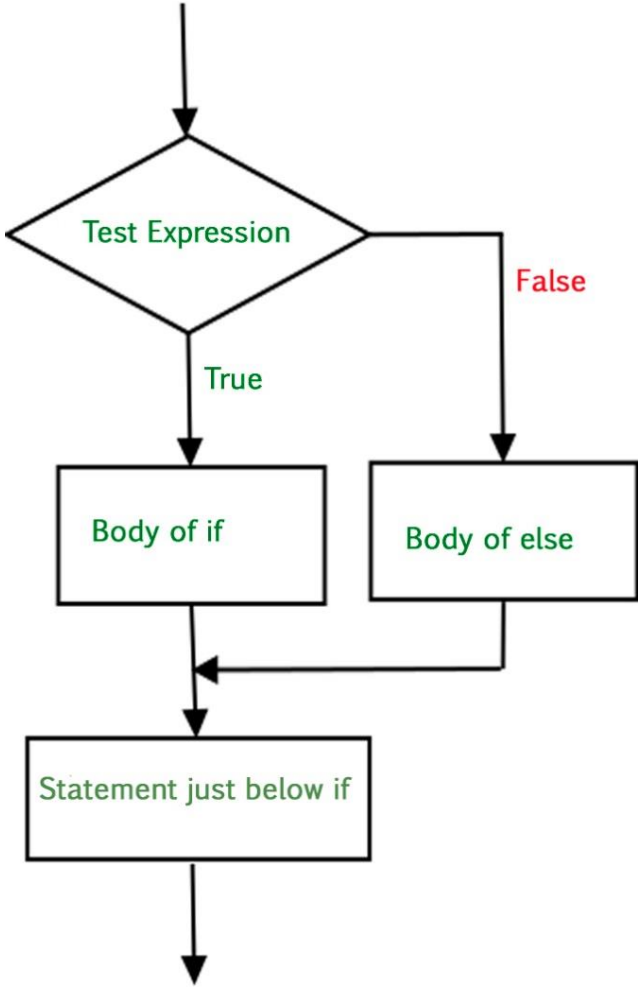
1. if  Statement
2. if—else Statement
3. if—elif Statement
4. if—elif—else Statement
5. Nested if Statement

# Flowchart of if & if….else

Flowchart of if Statement

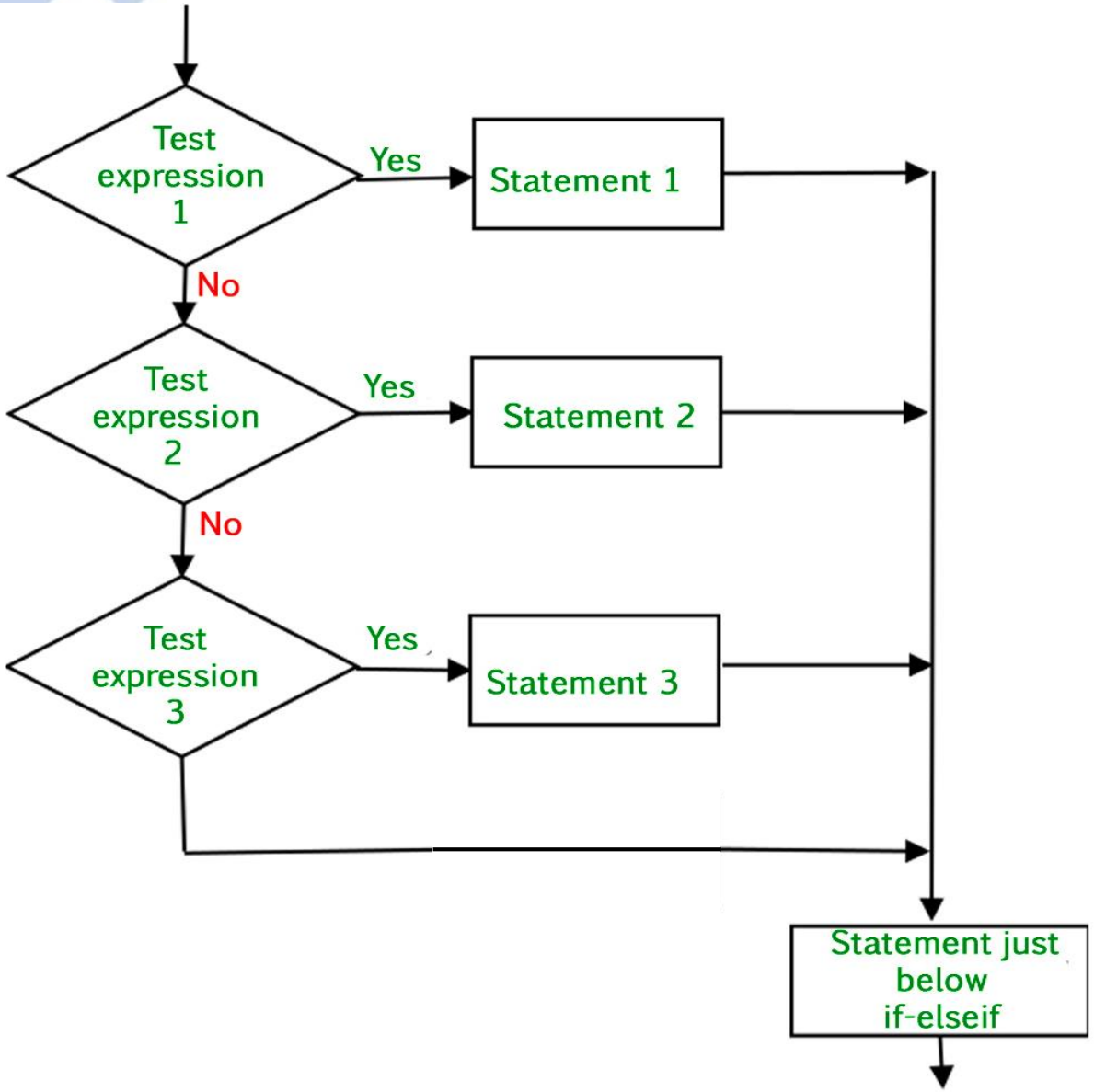



Flowchart of if….else Statement

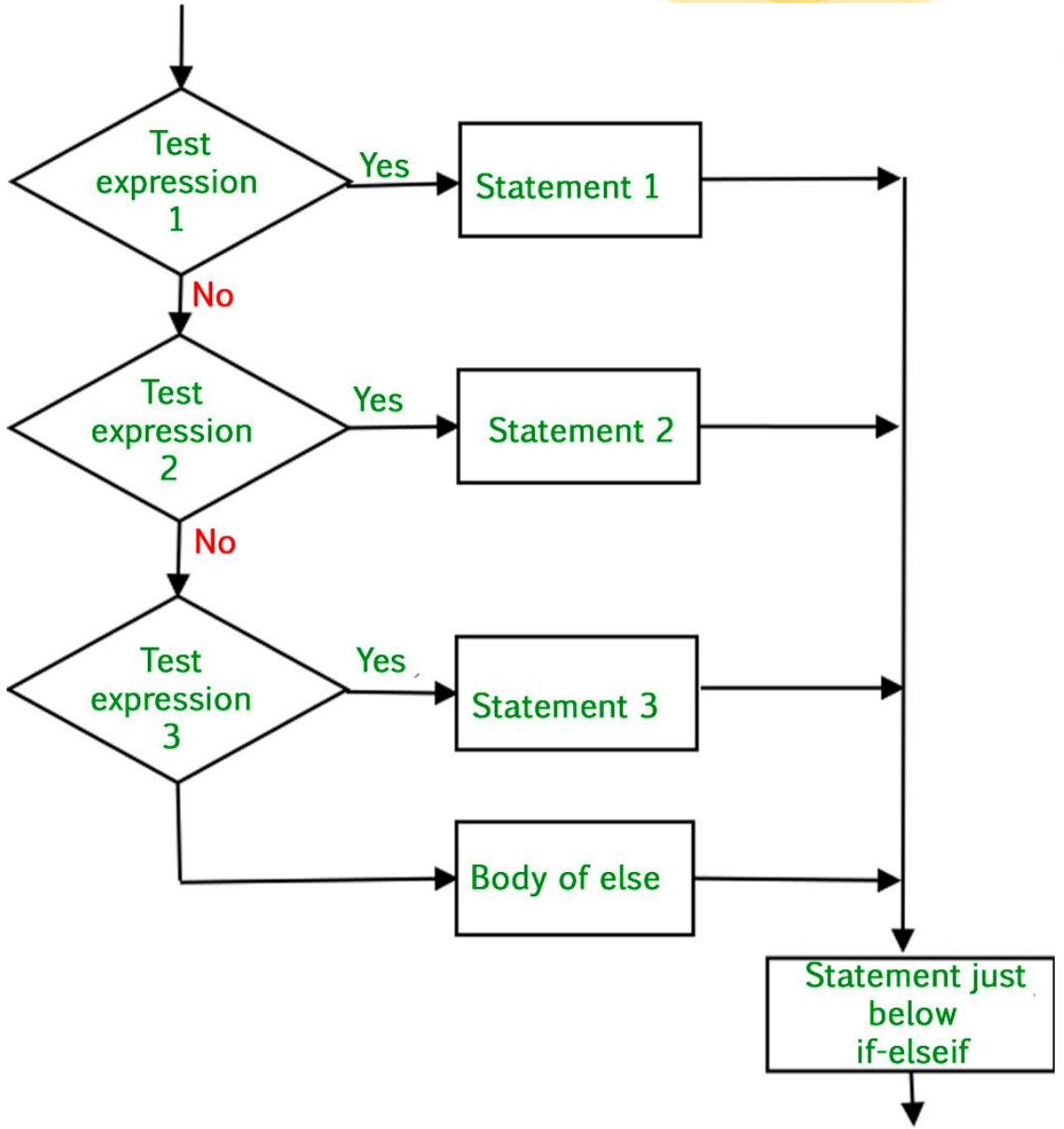


# Flowchart of if…elif & if…elif…else

Flowchart of if….elif Statement




Flowchart of if….elif….else Statement



# Control Flow Statements

## Example

### if

```
i = 10

if i > 15:
    print("10 is less than 15")
print("I am Not in if")
```

### If……elif

```
i = 25
if i == 10:
    print("i is 10")
elif i == 15:
    print("i is 15")
elif i == 20:
    print("i is 20")
```

### If……else

```
i = 20
if i < 15:
    print("i is smaller than 15")
    print("i'm in if Block")
else:
    print("i is greater than 15")
    print("i'm in else Block")
print("i'm not in if and not in else Block")
```

### If……elif……else

```
i = 20
if i == 10:
    print("i is 10")
elif i == 15:
    print("i is 15")
elif i == 20:
    print("i is 20")
else:
    print("i is not present")
```

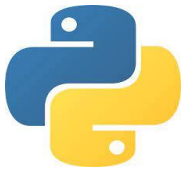
# Python Conditions

If statement, without indentation (will raise an error):

```
a = 33
b = 200
if b > a:
print("b is greater than a")
# you will get an error
```

**Else:** The else keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
   print("b is greater than a")
else:
   print("b is not greater than a")
```



# Python Conditions

**Elif:** The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

```
a = 33
b = 33
if b > a:
   print("b is greater than a")
elif a == b:
   print("a and b are equal")
```

You can also have an else without the elif:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

# Python Conditions (Ternary Operators)

**Shorthand if:** If you have only one statement to execute, you can put it on the same line as the if statement. This technique is known as **Ternary Operators**, or **Conditional Expressions**. One line if statement is like that:

Structure: X `if` condition `else` Y

**One line if else statement:**

```
a = 20
b = 330
print("A") if a > b else print("B")
```

**Similar Code:**

```
a = 20
b = 330
if a > b:
   print("A")
else:
   print("B")
```

# Python Conditions (Ternary Operators)

**Shorthand If:** One line if else statement, with 3 conditions:

Structure: W `if` condition_1 (`else` X `if` condition_2) (`else` Y `if` condition_3)….. `else` Z

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

**Similar Code:** This code is similar to the code below :

```
a = 330
b = 330
if a > b:
   print("A")
elif a == b:
   print("=")
else:
   print("B")
```

# Python Conditions (Multiple Conditions)

**And:** The and keyword is a logical operator, and is used to combine conditional statements: Test if a is greater than b, AND if c is greater than a:

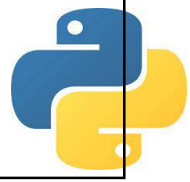
```
a = 200
b = 33
c = 500
if a > b and c > a:
   print("Both conditions are True")
```

**Or:** The or keyword is a logical operator, and is used to combine conditional statements:

```
a = 200
b = 33
c = 500
if a > b or a > c:
   print("At least one of the
conditions is True")
```

**Not:** The not keyword is a logical operator, and is used to reverse the result of the conditional statement:

```
a = 33
b = 200
if not a > b:
   print("a is NOT greater than b")
```



# Python Conditions (Nested if)

**Nested If:** You can have if statements inside if statements, this is called *nested* if statements.

```
x = 41
if x > 10:
   print("Above ten.")
   if x > 20:
      print("and also above 20!!")
      if x > 30:
         print("and also above 30!!!")
   else:
      print("but not above 20.")
else:
   print("Nothing to do.")
```

# Python Conditions (Nested if)

**Nested if another example:**

```
x = 41
if x > 10:
    print("Above ten.")
    if x > 20:
       print("and also above 20!!")
       if x > 30:
          print("and also above 30!!!")
          if x > 40:
             print("and also above 40!!!!")
       else:
          print("but not above 20.")
    else:
      print("but not above 20.")
```

# Python Conditions

**The pass Statement:** if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

```
a = 33
b = 200

if b > a:
    pass

# having an empty if statement like this, would
raise an error without the pass statement
```

# Letter Grade Program

```
print("Enter Marks Obtained in 5 Subjects: ")
markOne = int(input())
markTwo = int(input())
markThree = int(input())
markFour = int(input())
markFive = int(input())

tot = markOne+markTwo+markThree+markFour+markFive
avg = tot/5

if avg>=90 and avg<=100:
    print("Your Grade is A+")
elif avg>=80 and avg<90:
    print("Your Grade is A")
elif avg>=70 and avg<80:
    print("Your Grade is B")
elif avg>=60 and avg<70:
    print("Your Grade is C")
elif avg>=50 and avg<60:
    print("Your Grade is F")
```

# Leap Year Program

```
yr=int(input("Enter a year:"))

if yr % 4 == 0:
   print("It's a leap year...!!!!")
else:
   print("It's not a leap year....!!!")
```

# Weight Converter Program

```
weight = float(input("Weight: "))
unit = input("(K)g or (L)bs: ")

if unit.upper() == "K":
    converted = weight / 0.4
    print(f"Your weight is {converted} pounds")
else:
    converted = weight * 0.4
    print(f"Your weight is {converted} kilos")
```

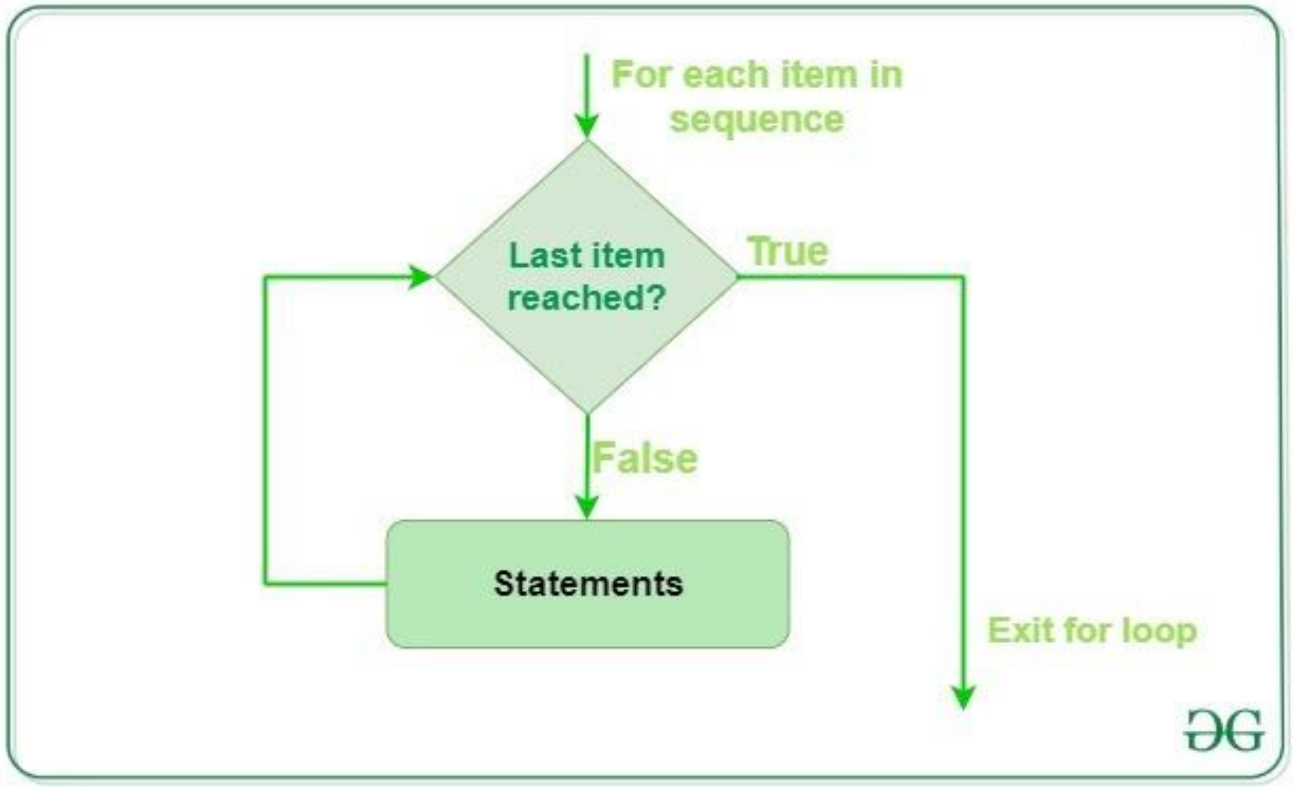
# Python LOOP

**Python Loops:** Python has two primitive loop commands:

- for loops
- while loops

### For Loops Syntax

```
for variable in iterable:
    # statements

for value in sequence:
    # block of code
```



# Python LOOP (FOR)

**Python For Loops:** Python For loop is used for sequential traversal i.e. it is used for iterating over a sequence like String, List, Tuple, Set, or Dictionary. With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

**Note:** In Python, for loops only implement the collection-based iteration.

1. For Loop in range()
2. For Loop in String
3. For Loop with List
4. For Loop with Tuple
5. For Loop with Zip()
6. For Loop in Dictionary
7. For Loop inside a For Loop
8. Else With For loop

# Python For LOOP (Range)

**The range() Function:** To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):
  print(x)
```

range() function takes mainly **three** arguments like as: range(start, stop, step)

**start:** integer starting from which the sequence of integers is to be returned
**stop:** integer before which the sequence of integers is to be returned.
The range of integers ends at a stop – 1.
**step:** The increment between each integer in the sequence.

# Python For LOOP (Range)

The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

```
for x in range(2, 6):
  print(x)
```

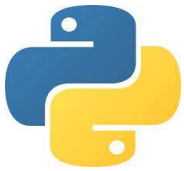
```
for x in range(2, 30, 3):
   print(x)
```

The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(2, 30, 3):

**Looping Through a String:** Even strings are iterable objects, they contain a sequence of characters:

```
for x in "banana":
  print(x)
```

```
str = "Geeks"
for i in str:
    print(i)
```



# Python LOOP (FOR)

**For loop with list:** This code uses a for loop to iterate over a list of strings, printing each item in the list on a new line. The loop assigns each item to the variable fruit and continues until all items in the list named fruits have been processed.

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
  print(fruit)
```

**For loop with tuple:** A tuple can be iterated using a for loop with tuple unpacking. In each iteration, the values from the inner tuple are assigned to the variables of the loop respectively, (here a and b).

```
tp = ((1, 2), (3, 4), (5, 6))
# print(type(tp))
for a, b in tp:
    print("Item of tuple:", a, b)
```

# Python LOOP (FOR)

**For loop in zip:** The zip() function used to iterate over two or multiple lists in parallel. The for loop assigns the corresponding elements of both lists to the variables inside the loop (such as: fruit and color) in each iteration.

```
fruits = ["apple", "banana", "cherry"]
colors = ["red", "yellow", "green"]
for fruit, color in zip(fruits, colors):
    print(fruit, "is", color)
```

```
fruits = ["apple", "banana", "cherry"]
colors = ["red", "yellow", "green"]
prices = [200, 100, 400]
for fruit, color, price in zip(fruits, colors, prices):
    print(fruit, "is", color, "and price:", price)
```

# Python For Loop

**For loop in a dictionary (key-value):** A for loop to iterate over a dictionary and print each key-value pair on a new line. The loop assigns each key to the variable i and uses string formatting to print the key and its corresponding value.

```
my_dict = {
    'ab' : 12,
    'xy' : 45
}

print(type(my_dict))
for i in my_dict:
    print("%s is the key of value %d" %(i, my_dict[i]))
```

# Python LOOP (FOR)

The **break Statement:** With the break statement we can stop the loop before it has looped through all the items:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

# Python LOOP (FOR)

**The continue Statement:** With the continue statement we can stop the current iteration of the loop, and continue with the next:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Remember that the continue statement should be the top of the block of code in for loop. Otherwise, it doesn't work properly. Like as:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    continue
```

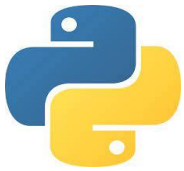
# Python LOOP (FOR)

**The pass Statement:** for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

```
for x in [0, 1, 2]:
  pass

# having an empty for loop like this, would raise an error without the pass statement
```

**Nested Loops:** A nested loop is a loop inside a loop. The "inner loop" will be executed one time for each iteration of the "outer loop":

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)

for i in range(1, 4):
    for j in range(1, 4):
        print(i, j)
```



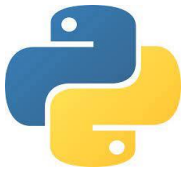
# Python LOOP (FOR)

**Else in For Loop:** The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

**The break statement with else**: Break the loop when x is 3, and see what happens with the else block:

```
for x in range(6):
  if x == 3:
     break
  print(x)
else:
  print("No break!")
```



# Python LOOP

**The while Loop:** With the while loop we can execute a set of statements as long as a condition is true.

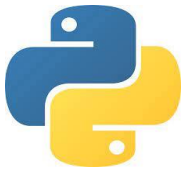
```
i = 1
while i < 6:
  print(i)
  i += 1
```

**The break Statement:** With the break statement we can stop the loop even if the while condition is true:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

**The continue Statement:** With the continue statement we can stop the current iteration, and continue with the next:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

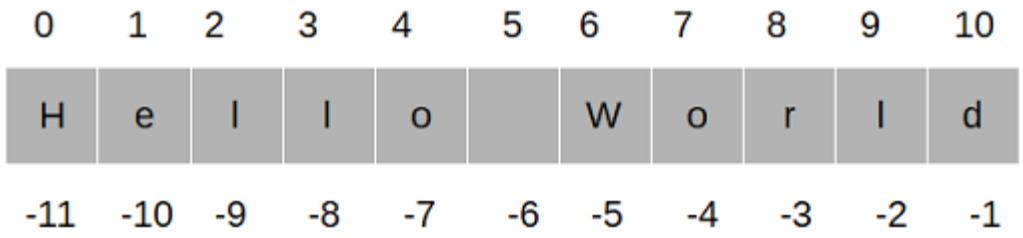


# Python Strings

Strings: Strings in python are surrounded by either single quotation marks or double quotation marks.

'Hello' is the same as "Hello". You can display a string literal with the print() function:

```
Example
print("Hello World")
print('Hello World')
```



# Python Strings

**Assign String to a Variable:** Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

```
Example
a = "Hello"
print(a)
```

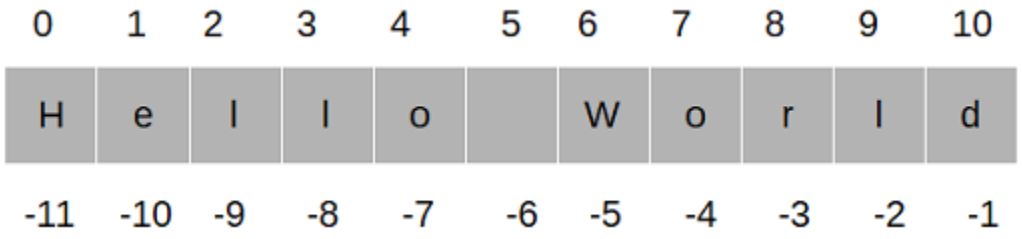
**Multiline Strings:** You can assign a multiline string to a variable by using three quotes:

```
Example
a = """Lorem ipsum dolor sit
amet,consectetur adipiscing elit,sed
do eiusmod tempor incididuntut
labore et dolore magna aliqua."""
print(a)
```

# Python Strings

**Strings are Arrays:** Like many other popular programming languages, strings in Python are arrays of bytes representing Unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1. Square brackets can be used to access elements of the string.




```
Example
a = "Hello World"
print(a[1])
```

# Python Strings

**Looping Through a String:** Since strings are arrays, we can loop through the characters in a string, with a for loop.

```
Example
for x in "banana":
  print(x)
```

**String Length:** To get the length of a string, use the len() function.
The len() function returns the length of a string:

```
Example
a = "Hello, World!"
print(len(a))
```

# Python Strings

**Check String:** To check if a certain phrase or character is present in a string, we can use the keyword in.

```
Example
txt = "The best things in life are free!"
print("free" in txt)
```

Use it in an if statement. Print only if "free" is present:

```
Example
txt = "The best things in life are!"
if "free" in txt:
   print("Yes, 'free' is present.")
else:
   print("Not Present")
```

# Python Strings

**Slicing Strings:** you can return a range of characters by using the slice syntax.
Specify the start index and the end index, separated by a colon, to return a part of the string.

```
Example
b = "Hello, World!"
print(b[2:5])
```

**Slice From the Start:** By leaving out the start index, the range will start at the first character:

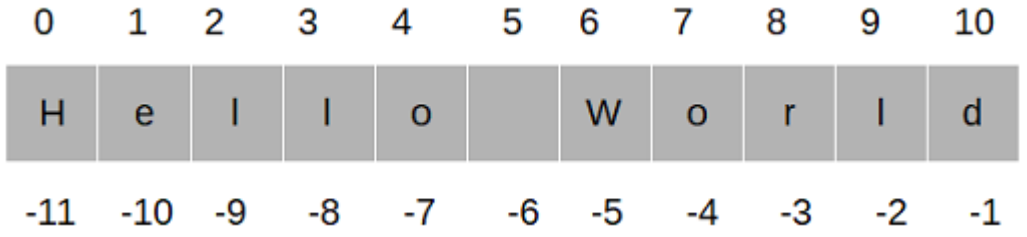
```
Example
b = "Hello, World!"
print(b[:5])
```

# Python Strings

**Negative Indexing:** Use negative indexes to start the slice from the end of the string:

```
Example
b = "Hello, World!"
print(b[-5:-2])
```

```
a = "Hello World"
print(a)
print(a[0])
print(a[-1])
print(a[0:3])
print(a[0:])
print(a[1:])
print(a[:4])
print(a[0:-1])
print(a[2:-3])
```



# Python Strings

**Modify Strings:** Python has a set of built-in methods that you can use on strings.

**Upper Case:** The upper() method returns the string in upper case:

Example

```
a = "Hello, World!"
print(a.upper())
```

**Lower Case:** The lower() method returns the string in lower case:

Example

```
a = "Hello, World!"
print(a.lower())
```

# Python Strings

**Modify Strings: Remove Whitespace:** Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

```
Example
a = "    Hello, World!     "
print(a.strip()) #returns "Hello, World!"
```

**Replace String:** The replace() method replaces a string with another string:

```
Example
a = "Hello, World!"
print(a.replace("H", "J"))
```

# Python Strings

**Modify Strings: Split String:** The split() method returns a list where the text between the specified separator becomes the list items. The split() method splits the string into substrings if it finds instances of the separator:

Example

```
a = "Hello, World!"
print(a.split(","))
# returns ['Hello', ' World!']
```

**String Concatenation:** To concatenate, or combine, two strings you can use the + operator.

Example

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

Example (Add " ")

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

# Python Strings

**String Format:** As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

```
age = 36
txt = "My name is John, I am " + age
print(txt)           # Wrong Code(Show an Error)
txt = "My name is John, I am " + str(age)
print(txt)           # This is Right Code
```

But we can combine strings and numbers by using the format() method! The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

**Example:** Use the format() method to insert numbers into strings:

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

# Python Strings

**String Format:** The format() method takes an unlimited number of arguments, and are placed into the respective placeholders:

**Example**

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {}
Taka."
print(myorder.format(quantity, itemno, price))
```

# Python Strings

**String Format:** You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

**Example**

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0}
pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

# Python Strings

f-strings are a great new way to format strings. Not only are they more readable, more concise, and less prone to error than other ways of formatting, they are also faster!

```
name = "Eric"
age = 74
# Hello, Eric. You are 74.
print("Hello, " + name + ". You are " + str(age) + ".")
print(f "Hello, {name}. You are {age}.")
```

# Python Strings

**Escape Character:** To insert characters that are illegal in a string, use an escape character. An escape character is a backslash \ followed by the character you want to insert. An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

**Example:** You will get an error if you use double quotes inside a string that is surrounded by double quotes:

```
txt = "We are the so-called "Vikings" from the north."
```

#You will get an error if you use double quotes inside a string that are surrounded by double quotes:

# Python Strings

**Escape Character:** To fix this problem, use the escape character \":

**Example:** The escape character allows you to use double quotes when you normally would not be allowed:

```
txt = "We are the so-called \"Vikings\" from the north."
```

# Python Strings

**Escape Characters:**

| Code | Result |
|---|---|
| \' | Single Quote |
| \\ | Backslash |
| \n | New Line |
| \r | Carriage Return |

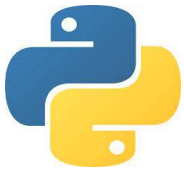
```
txt = 'It\'s alright.'
print(txt)

txt = "This will insert one \\
(backslash)."
print(txt)

txt = "Hello\nWorld!"
print(txt)

txt = "Hello\rWorld!"
print(txt)
```

# Python Strings

## Escape Characters:

| Code | Result |
|---|---|
| \t | Tab |
| \b | Backspace |
| \ooo | Octal value |
| \xhh | Hex value |

```
txt = "Hello\tWorld!"
print(txt)

#This example erases one character
(backspace):
txt = "Hello \bWorld!"
print(txt)

#A backslash followed by three integers
will result in a octal value:
txt = "\110\145\154\154\157"
print(txt)

#A backslash followed by an 'x' and a hex
number represents a hex value:
txt = "\x48\x65\x6c\x6c\x6f"
print(txt)
```



# String Methods/Functions

https://www.w3schools.com/python/python_ref_string.asp

| Method | Description |
|---|---|
| capitalize() | Converts the first character to upper case |
| casefold() | Converts string into lower case |
| center() | |
| count() | Returns the number of times a specified value occurs in a string |
| encode() | |
| endswith() | |
| expandtabs() | |
| find() | Searches the string for a specified value and returns the position of where it was found |
| format() | |
| index() | Searches the string for a specified value and returns the position of where it was found |
| isalnum() | Returns True if all characters in the string are alphanumeric |
| isalpha() | Returns True if all characters in the string are in the alphabet |
| isdecimal() | Returns True if all characters in the string are decimals |
| isdigit() | Returns True if all characters in the string are digits |
| isidentifier() | |

# String Methods/Functions

https://www.w3schools.com/python/python_ref_string.asp

| Method | Description |
|---|---|
| islower() | Returns True if all characters in the string are lower case |
| isnumeric() | Returns True if all characters in the string are numeric |
| isprintable() | Returns True if all characters in the string are printable |
| isspace() | Returns True if all characters in the string are whitespaces |
| istitle() | |
| isupper() | Returns True if all characters in the string are upper case |
| join() | Converts the elements of an iterable into a string |
| ljust() | Returns a left justified version of the string |
| rjust() | |
| lower() | Converts a string into lower case |
| lstrip() | Returns a left trim version of the string |
| rstrip() | |
| maketrans() | Returns a translation table to be used in translations |
| partition() | Returns a tuple where the string is parted into three parts |
| replace() | Returns a string where a specified value is replaced with a specified value |

# String Methods/Functions

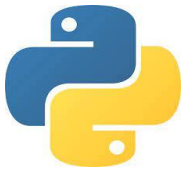
https://www.w3schools.com/python/python_ref_string.asp

| Method | Description |
|---|---|
| rfind() | Searches the string for a specified value and returns the last position of where it was found |
| rindex() | Searches the string for a specified value and returns the last position of where it was found |
| rjust() | Returns a right justified version of the string |
| rpartition() | Returns a tuple where the string is parted into three parts |
| rsplit() | Splits the string at the specified separator, and returns a list |
| rstrip() | Returns a right trim version of the string |
| split() | Splits the string at the specified separator, and returns a list |
| splitlines() | Splits the string at line breaks and returns a list |
| startswith() | Returns true if the string starts with the specified value |
| strip() | Returns a trimmed version of the string |
| swapcase() | Swaps cases, lower case becomes upper case and vice versa |
| title() | Converts the first character of each word to upper case |
| translate() | Returns a translated string |
| upper() | Converts a string into upper case |
| zfill() | Fills the string with a specified number of 0 values at the beginning |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| capitalize() | Converts the first character to upper case<br>**Example :**<br>`txt = "hello, and welcome to my world."`<br>`x = txt.capitalize()`<br>`print (x)` |
| casefold() | Converts string into lower case<br>**Example:**<br>`txt = "Hello, And Welcome To My World!"`<br>`x = txt.casefold()`<br>`print(x)` |
| center() | Returns a centered string<br>**Example:**<br>`txt = "banana"`<br>`x = txt.center(100)`<br>`print(x)` |



# Python Strings

## String Methods

| Method | Description |
|---|---|
| count() | Returns the number of times a specified value occurs in a string<br>**Example:** Return the number of times the value "apple" appears in the string:<br>`txt = "I love apples, apple are my favorite fruit"`<br>`x = txt.count("apple")`<br>`print(x)` |
| encode() | Returns an encoded version of the string<br>**Example:** UTF-8 encode the string:<br>`txt = "My name is Ståle"`<br>`x = txt.encode()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| endswith() | Returns true if the string ends with the specified value<br>**Example:** Check if the string ends with a punctuation sign (.):<br>`txt = "Hello, welcome to my world."`<br>`x = txt.endswith(“d.")`<br>`print(x)` |
| expandtabs() | Sets the tab size of the string<br>**Example:** Set the tab size to 2 whitespaces:<br>`txt = "H\te\tl\tl\to"`<br>`x =  txt.expandtabs(2)`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| find() | Searches the string for a specified value and returns the position of where it was found<br>**Example:** Where in the text is the word "welcome"?:<br>`txt = "Hello, welcome to my world."`<br>`x = txt.find("welcome")`<br>`print(x)`<br><br>Where in the text is the first occurrence of the letter "e" when you only search between position 5 and 10?: if not found then return -1.<br>`txt = "Hello, welcome to my world."`<br>`x = txt.find("e", 5, 10)`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| format() | Formats specified values in a string<br>**Example:** Insert the price inside the placeholder, the price should be in fixed point, two-decimal format:<br>`txt = "For only {price:.2f} dollars!"`<br>`print(txt.format(price = 49))` |
| index() | Searches the string for a specified value and returns the position of where it was found. Similar to find() method but if data not found index returns error but find return -1. |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isalnum() | Returns True if all characters in the string are alphanumeric. The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9). Example of characters that are not alphanumeric: (space)!#%&? etc.<br><br>Example: Check if all the characters in the text are alphanumeric:<br>`txt = "Company12"`<br>`x = txt.isalnum()`<br>`print(x)`<br><br>`txt = "Company 12"`<br>`x = txt.isalnum()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isalpha() | Returns True if all characters in the string are in the alphabet.<br>**Example:** Check if all the characters in the text are letters:<br><br>`txt = "CompanyX"`<br>`x = txt.isalpha()`<br>`print(x)`<br><br>`txt = "Company10"`<br>`x = txt.isalpha()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isdecimal() | Returns True if all characters in the string are decimals<br>**Example:** Check if all the characters in the Unicode object are decimals:<br><br>`txt = "\u0033" #unicode for 3`<br>`x = txt.isdecimal()`<br>`print(x)`<br><br>`txt = "33"`<br>`x = txt.isdecimal()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isdigit() | Returns True if all characters in the string are digits<br>**Example:** Check if all the characters in the text are digits:<br><br>`txt = "50800"`<br>`x = txt.isdigit()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isidentifier() | Returns True if the string is an identifier/ The isidentifier() method returns True if the string is a valid identifier, otherwise False. A string is considered a valid identifier if it only contains alphanumeric letters (a-z) and (0-9), or underscores (_). A valid identifier cannot start with a number, or contain any spaces. **Example:**<br>`txt = "Demo"`<br>`x = txt.isidentifier()`<br>`print(x)`<br><br>`txt = "Demo"`<br>`x = txt.isidentifier()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| islower() | Returns True if all characters in the string are lowercase.<br>**Example:** Check if all the characters in the text are in lowercase:<br><br>`txt = "hello world!"`<br>`x = txt.islower()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isnumeric() | Returns True if all characters in the string are numeric. The isnumeric() method returns True if all the characters are numeric (0-9), otherwise False. Exponents, like ² and ¾ are also considered to be numeric values. "-1" and "1.5" are NOT considered numeric values, because all the characters in the string must be numeric, and the - and the . are not.<br>Example: Check if all the characters in the text are numeric:<br><br>`txt = "565543"`<br>`x = txt.isnumeric()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isprintable() | Returns True if all characters in the string are printable<br>**Example:** Check if all the characters in the text are printable:<br><br>`txt = "Hello! Are you #1?"`<br>`x = txt.isprintable()`<br>`print(x)`<br><br>`txt = "Hello!\nAre you #1?"`<br>`x = txt.isprintable()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isspace() | Returns True if all characters in the string are whitespaces<br>**Example:** Check if all the characters in the text are whitespaces:<br><br>`txt = "  "`<br>`x = txt.isspace()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| istitle() | Returns True if the string follows the rules of a title<br>**Example:** Check if each word start with an upper case letter:<br><br>`txt = "Hello, And Welcome To My World!"`<br>`x = txt.istitle()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| isupper() | Returns True if all characters in the string are upper case<br>**Example:** Check if all the characters in the text are in upper case:<br><br>`txt = "THIS IS NOW!"`<br>`x = txt.isupper()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| join() | Joins the elements of an iterable to the end of the string<br>**Example:** Join all items in a tuple into a string, using a hash character as separator:<br><br>`myTuple = ("John", "Peter", "Vicky")`<br>`x = "#".join(myTuple)`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| ljust()<br><br>rjust() | Returns a left-justified version of the string.<br>**Example:** Return 20 characters long, left-justified version of the word "banana":<br><br>`txt = "banana"`<br>`x = txt.ljust(20)`<br>`print(x, "is my favorite fruit.")`<br><br>**Note:** In the result, there are actually 14 whitespaces to the right of the word banana.<br>`txt = "banana"`<br>`x = txt.rjust(20)`<br>`print(x, "is my favorite fruit.")` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| lower() | Converts a string into lowercase<br>**Example**: Lowercase the string:<br>`txt = "Hello my FRIENDS"`<br>`x = txt.lower()`<br>`print(x)` |
| lstrip() | Returns a left trim version of the string<br>**Example:** Remove spaces to the left of the string:<br>`txt = "     banana     "`<br>`print("of all fruits", txt, "is my favorite")`<br>`x = txt.lstrip()`<br>`print("of all fruits", x, "is my favorite")` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| rstrip() | Returns a left trim version of the string<br>**Example:** Remove spaces to the left of the string:<br>`txt = "     banana     "`<br>`print("of all fruits", txt, "is my favorite")`<br>`x = txt.rstrip()`<br>`print("of all fruits", x, "is my favorite")` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| maketrans() | Returns a translation table to be used in translations<br>**Example:** Create a mapping table, and use it in the translate() method to replace any "S" characters with a "P" character:<br><br>`txt = "Hello Sam!"`<br>`mytable = txt.maketrans("S", "P")`<br>`print(txt.translate(mytable))` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| partition() | Returns a tuple where the string is parted into three parts<br>**Example:** Search for the word "bananas", and return a tuple with three elements:<br><br>1 - everything before the "match"<br>2 - the "match"<br>3 - everything after the "match"<br><br>`txt = "I could eat bananas all day"`<br>`x = txt.partition("bananas")`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| replace() | Returns a string where a specified value is replaced with a specified value<br>**Example:** Replace the word "bananas":<br>`txt = "I like bananas"`<br>`x = txt.replace("bananas", "apples")`<br>`print(x)` |
| rfind() | Searches the string for a specified value and returns the last position of where it was found<br>**Example:** Where in the text is the last occurrence of the string "casa"?:<br>`txt = "Mi casa, su casa."`<br>`x = txt.rfind("casa")`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| rindex() | Searches the string for a specified value and returns the last position of where it was found<br>**Example:** Where in the text is the last occurrence of the string "casa"?:<br><br>`txt = "Mi casa, su casa."`<br>`x = txt.rindex("casa")`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| rpartition() | Returns a tuple where the string is parted into three parts<br>**Example:** Search for the last occurrence of the word "bananas", and return a tuple with three elements:<br>1 - everything before the "match"<br>2 - the "match"<br>3 - everything after the "match"<br><br>`txt = "I could eat bananas all day, bananas are my favorite fruit"`<br>`x = txt.rpartition("bananas")`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| rsplit() | Splits the string at the specified separator, and returns a list<br>**Example:** Split a string into a list, using comma, followed by a space (, ) as the separator:<br><br>`txt = "apple, banana, cherry"`<br>`x = txt.rsplit(", ")`<br>`print(x)` |
| rstrip() | Returns a right trim version of the string<br>**Example:** Remove any white spaces at the end of the string:<br><br>`txt = "     banana     "`<br>`x = txt.rstrip()`<br>`print("of all fruits", x, "is my favorite")` |

# Python Strings

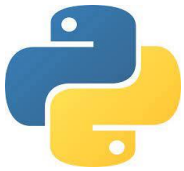
## String Methods

| Method | Description |
|---|---|
| split() | Splits the string at the specified separator, and returns a list<br>**Example:** Split a string into a list where each word is a list item:<br><br>`txt = "welcome to the jungle"`<br>`x = txt.split()`<br>`print(x)` |
| splitlines() | Splits the string at line breaks and returns a list<br>**Example:** Split a string into a list where each line is a list item:<br><br>`txt = "Thank you for the music\nWelcome to the jungle"`<br>`x = txt.splitlines()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| startswith() | Returns true if the string starts with the specified value<br>**Example:** Check if the string starts with "Hello":<br><br>`txt = "Hello, welcome to my world."`<br>`x = txt.startswith("Hello")`<br>`print(x)` |
| strip() | Returns a trimmed version of the string<br>Example: Remove spaces at the beginning and at the end of the string:<br><br>`txt = "     banana     "`<br>`x = txt.strip()`<br>`print("of all fruits", x, "is my favorite")` |



# Python Strings

## String Methods

| Method | Description |
|---|---|
| swapcase() | Swaps cases, lower case becomes upper case and vice versa<br>**Example:** Make the lower case letters upper case and the upper case letters lower case:<br><br>`txt = "Hello My Name Is PETER"`<br>`x = txt.swapcase()`<br>`print(x)` |
| title() | Converts the first character of each word to upper case<br>**Example:** Make the first letter in each word upper case:<br><br>`txt = "Welcome to my world"`<br>`x = txt.title()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| translate() | Returns a translated string<br>**Example:** Replace any "S" characters with a "P" character:<br><br>`#use a dictionary with ASCII codes to replace 83 (S) with 80 (P):`<br>`mydict = {83:  80}`<br>`txt = "Hello Sam!"`<br>`print(txt.translate(mydict))` |
| upper() | Converts a string into upper case<br>Example-Upper case the string:<br><br>`txt = "Hello my friends"`<br>`x = txt.upper()`<br>`print(x)` |

# Python Strings

## String Methods

| Method | Description |
|---|---|
| zfill() | Fills the string with a specified number of 0 values at the beginning<br>**Example:** Fill the string with zeros until it is 10 characters long:<br><br>`txt = "50"`<br>`x = txt.zfill(10)`<br>`print(x)` |