--STRINGS--

>>>>>Assigning Strings<<<<<

#Strings are sequences of characters, using the syntax of either single quotes, double quotes or triple quotes:
#Example:

```
print('Hello')
print("Hello")
```

#using quote under a quote:

```
print("I am 'Ucchash'")
print("I am 'Ucchash'")

print("He's a good guy")
print('He is a "good" guy')
```

#three quotes used to write strings with multiple lines:

```
long_string = """
Amar
Sonar
Bangla
Ami
Tomay
Valobashi"""

print(long_string
```

>>>>>Escape Sequences<<<<<

#Types of escape sequences:
# Hash to indicate a comment
#Backslash (\) is an escape character
#backslash indicates what comes after this is a string
(\") , (\') , (\\)
# (\n) for new line
#(\t) for new tab (double space)
#triple quotation can be used to split string

Example 01:

```
course1 = "Python \"programming"
print(course1)
course2 = "Python \'programming"
print(course2)
course3 = "Python \\programming"
print(course3)
course4 = "Python \nprogramming"
print(course4)
course5 = "Python \tprogramming"
print(course5)
```

Example 02:

```
print('The per shop owner said "No, no \'e\'s uh... he is resting"')
#the backslash treats the quotation as normal string, that's how
we can use multiple equations if needed. #backslash indicates
what comes after this is a string
```

Example 03:

```
split_string = "This string has been \n split over \n several \n
lines"
print(split_string) # "\n" is the new line command

tabbed_sting = "1\t2\t3\t4\t"
print(tabbed_sting) #"\t" is the tab command (double space)

another_split_string = """This string has been
split over
several
lines"""  #triple quotation can be used to split string like this

print(another_split_string)
```

Example 04:

```
my_string = """My \
name \
is \
Ucchash \
Sarkar"""
#Backslash removes the new line method
print(my_string)
```

Example 05:

```
print("hello Mike welcome to the club")
print("hello Mike \nwelcome to the club")
print("hello Mike \nwelcome to the club\nwe are glad to have you here")
```

>>>>>String Concatenation<<<<<

>> Combining two string types:

```
print("Hello" + " World")
```

>>With assigned variables:

```
greeting = "Hello"
name = "Ucchash"
print(greeting+name)
print(greeting+" "+name) #add a space between two strings
```

>>Augmented strings:

```
y = "Hello "
print(y)
y+="World"
print(y)
```

>>Input a string:

```
greeting = "Hello"
name = input("Please enter yout name: ") #the user input function accepts input as string (default)
print(greeting+ " "+name ) #add a space between two strings
```

>>Type conflict and conversion:

```
Print(type(x) #to know the type
```

# to convert then know the type:

```
print(type(int(str(100)))) # final result is an integer type
```

#adding different data types:

```
print("Hello") #string
print("5") #string
print("Hello" + "World") #string+string
print("5" + "5") #string+string
print(5+5) #integer+integer
print("Hello " + "I " + "am " + "Ucchash") #all strings
print("Hello" + "5")  #all strings
print("Hello" + 5) #TypeError, cannot combine two different types
print("Hello " *3)
print("Hello" * "3") #TypeError
```

```
print("1" +1)  #TypeError , cannot combine two different types
```

Example 01:

```
greeting = "Hello"
name = "Ucchash"
print(greeting + name)
age = 24
print(age)
print(name + age) #error : cannot combine two different types
print(name + str(age)) #convert "int" into "string" to combine
them
print("Hello" *5)
print("Hello" *5 +4) #error : cant combine two diff types
```

Example 02:

```
x = input("x:") #the user input function accepts input as string
(default)
y = x+1
print(y) #TypeError, we need to convert the string into integer
to add with another integer

x2 = int(x) #string is converted into integer
y2 = x2 + 1 #now we can add two integers
print(y2)
```

Example 03:

```
y = input("y: ")
z = int(y) + 1 #converted into integer then addition
print(f"y: {y}, z: {z}") #used formatted string method to print the
values
```

>>>>>String Indexing<<<<<

#indexing starts from zero
#Python does not include the last index position given in the code!!
#From the ending the count begins from (-1),(-2),(-3) and so on..
#Example:
#string =   "Ucchash"
#indexing = "0123456"

>>Indexing method:

```
name = "Ucchash_Sarkar"
print(name)
print(name[0]) #starts counting from zero
print(name[5])
print(name[13]) #total 14 characters so last position is 13
print(name[14]) #shows error because last position is 13
#IndexError: string index out of range
```

# printing backwards:

```
print(name[-1]) #last one
print(name[-2]) #second last one
```

# printing a range:

```
print(name[0:7]) #1st to 7th value (does not show the last one)
#this should count 1st to 8th but python ignores the last call
print(name[:7]) #1st to 7th value (zero to six)
print(name[7:]) #7th to last value (seventh to last)
print(name[-4:-2]) #backward 4th to 2nd (ignores to 2nd)
```

>>Slicing:

# grab a subsection of multiple characters
# a "slice" of the string
# Syntax is: [start:end:step]

```
my_string = "Amar Sonar Bangla"
print(my_string[0:6:2]) #zero to sixth with step 2
print(my_string[0:9:3]) #zero to ninth with step 3
print(my_string[0::2]) #zero to last with step 2
print(my_string[:9:2]) #zero to nineth with step 2
print(my_string[::2]) #zero to last with step 2
print(my_string[::-1]) #reverse the string , (step size = -1)
```

#Example of stepping:

```
number1 = "9.223.372.036.854.775.807"
print(number1[1::4]) #one to last with step of 4
#discards the numbers shows only the dots
number2 = "1, 2, 3, 4, 5"
print(number2[0::3])#zero to last with step of 3
#discard the commas and spaces shows only the numbers
```

>>Strings are immutable:

```
my_string = "0123456789"
my_string[5] = "8"
print(my_string)
#TypeError: 'str' object does not support item assignment
#but we can change the whole string
my_string = "8"
print(my_string)
#so we can't reassign a part of a string but we can change it
entirely
```

>>>>>String Formatting<<<<<

>> Method-1 (percent "%" method):

# (%s) for strings
# (%d) for decimals (integers)
# (%f) for floating's

Example 01:

```
print("Mango : %s color, %s taste, %s origin" %("Red" , "Sweet",
"Dinajpur"))
print("Identity: name= %s , age = %d years , weight = %f kg"
%("Ucchash", 23, 65.5))
print("Identity: name= %s , age = %d years , weight = %.2f kg"
%("Ucchash", 23, 65.5))
#(%.2f we want a single digit after floating point)
```

Example 02:

```
age = 24
print("My age is %d years" % age) # "%d" to format "integer"

age = 24
print("My age is %d %s, %d %s" % (age, "years", 6, "months")) #
"%s" to format "string"

for i in range(1, 12):
   print("No. %2d squared is %3d and cubed is %4d" %(i, i ** 2, i
** 3)) # "%2", "%3" and "%4" allocates two, three & four empty
```

```
positions to store the value

print("Pi is approximately %12f" % (22 / 7)) #"%f" to format
"floats" , "%12f" allocates twelve empty positions to store the
value
```

>> Method-2 (dot "." format method):

Example 01 :

```
age = 24
print("My age is {0} years".format(age)) #assigns "age" into the
first "third bracket"
```

Example 02:

#multiple assignment of values in multiple brackets (1st "value" to 1st "third bracket", 2nd "value" to 2nd "third bracket" and so on...

```
print("There are {0} days in {1}, {2}, {3}, {4}, {5}, {6} and {7}
".format(31, "January", "March", "May", "July", "August",
"October", "December"))
```

Example 03:

#assigning 1st value to all {1}'s 2nd value to all {2}'s and so on...

```
print("January: {2}, February: {0}, March: {2}, April: {1}, May: {2},
June: {1}, July: {2}, August: {2}, September: {1}, October: {2},
November: {1}, December: {2}".format(28, 30, 31))
```

```
#Using triple quotes to make new lines:
```

```
print("""January: {2}
February: {0}
March: {2}
April: {1}
May: {2}
June: {1}
July: {2}
August: {2}
September: {1}
October: {2}
November: {1}
December: {2}""".format(28, 30, 31))
```

Example 04:

```
for i in range(1, 12):
    print("No. {0:2} squared is {1:4} and cubed is {2:4}".format(i, i
** 2, i ** 3)) #allocating two empty positions for 0th , four for
1st and four for 2nd position

for i in range(1, 12):
    print("No. {0:2} squared is {1:<4} and cubed is {2:<4}".format(i, i
** 2, i ** 3)) #starts from the left side
```

Example 05:

#allocating how many values we want after the decimal point:

```
print("Pi is approximately {0:12.50f}".format(22 / 7)) #after
decimal 50 values
print("Pi is approximately {0:52.50f}".format(22 / 7)) #total 52,
after decimal = 50 values
print("Pi is approximately {0:62.50f}".format(22 / 7)) #total 62,
after decimal = 50 values
print("Pi is approximately {0:72.50f}".format(22 / 7)) #total 72,
after decimal = 50 values
```

Example 06:

```
for i in range(1, 12):
    print("No. {} squared is {} and cubed is {:4}".format(i, i ** 2, i **
3)) #only allocating empty states for the last value
```

Example 07:

```
name = "Johnny"
age = 55
print("Hi " + name +". You are " + str(age) + " years old" )
print(f"Hi {name}. You are {age} years old")
print("Hi {}. you are {} years old".format("Johnny", "55"))
print("Hi {}. you are {} years old".format(name, age))
print("Hi {0}. you are {1} years old".format(name, age))
print("Hi {1}. you are {0} years old".format(name, age))
print("Hi {new_name}. You are {new_age} years
old".format(new_name = "Alex", new_age = 25))
```

Example 08:

```
numbers = [4, 5, 6]
new_string = "Numbers: {0}, {1}, {2}".format(numbers[0],
numbers[1], numbers[2])
print(numbers)
print(new_string)
```

Example 09:

```
date = [12, 12, 2020]
date_format = "Date: {}/{}/{}".format(date[0], date[1], date[2])
print(date)
print(date_format)
```

Example 10:

```
my_string = "x= {x}, y= {y}".format(x=5, y=6)
print(my_string)
```

>> Method-3 (f “{} strings {}” method):

#Simplest and most commonly used

Example 01:

```
first = "Ucchash"
last = "Sarkar"
full = f"{first} {last}"
justanexample = f"{len(first)} {2+2}"
print(full)
print(justanexample)
```

Example 02:

```
first = "Ucchash"
last = "Sarkar"
full1 = first + " " + last #combining two string
print(full1)
full2 = f"{first} {last}" #using formatted string method
print(full2)
example = f"{len(first)} {last}" #we can also use built-in functions
in the formatted strings
print(example)
example2 = f"{len(first)} {2+2}" #we can put any valid expressions
in between curley braces
print(example2)
```

>>>>>String Functions<<<<<

>>Built-in Functions:

#The length “len()” function:
#returns number of characters in the string
#this is not limited to strings only

Example 01:

```
course = "Python Programming"
print(len(course))
```

Example 02:

```
greet = "Hellooooo"
print(len(greet))
print(greet[0:len(greet)])
```

>>String Methods:

#(google - w3schools)
#these are functions related to strings
#everything in python is an object,
#objects have functions called methods,
#we can access the methods using dot(.) notation.

Example 01:

```
course = "--python programming ucchash--"
print(course.upper()) #makes everything upper case
print(course.lower()) #makes everything lower case
print(course.title()) #Makes first alphabet of every word
uppercase
print(course.capitalize()) #capitalizes the beginning of the
sentence
print(course.find("python")) #returns the index of “python”
starting if found
print(course.find("Antu")) # returns (-1) if not found
print(course.replace('p','x')) #replace all "p" with "x"
print(course.replace("python", "java")) #replace whole word
print(course.count("p")) #counting the occurance of "p"
print(course.index("u")) #1st occurance of "u"
print(course.index("ucchash")) #1st occurance of "ucchash"
print(course.index("antu")) #shows ValueError
print(course.startswith("--")) #true
print(course.startswith("abc")) #false
print(course.endswith("--")) #true
print(course.endswith("--")) #false
print(course) #original string unchanged (immutable nature)
```

Example 02:

# 1. strip() :used to delete all the leading and trailing characters mentioned in its argument.
# 2. lstrip() :used to delete all the leading characters mentioned in its argument.
# 3. rstrip() :used to delete all the trailing characters mentioned in its argument.

```
print(course.strip()) #strips any spaces from first or last
print(course.strip("--")) #strips given value from first or last
print(course.lstrip()) #strips any spaces from first
print(course.lstrip("--")) #strips given value from first
print(course.rstrip()) #strips any spaces from last
print(course.rstrip("--")) #strips given value from last
print(course) #original string unchanged (immutable nature)
```

>>Existence of characters:

# (True/False)

```
today = "Friday"
print("day" in today)
print("fri" in today)
print("sun" in today)
print("fri" in "yesterday")
print("sat" in "saturday")
print("day" not in today)
print("sun" not in today)
print("sat" not in "saturday")
```

>> Split & Join strings:

#Join a string with a list:

```
my_list = ["Apple", "Banana", "Mango"]
print(my_list)
print(",".join(my_list))
print(":".join(my_list))
```

#Split a string:

```
example3 = "Mango, Apple, Banana, Orange"
print(example3.split(","))
#splits the string into a list where a comma (",") is found.
print("_".join(example3))
#Adding a desired character ("_") between each character of
the given string
```

--END--