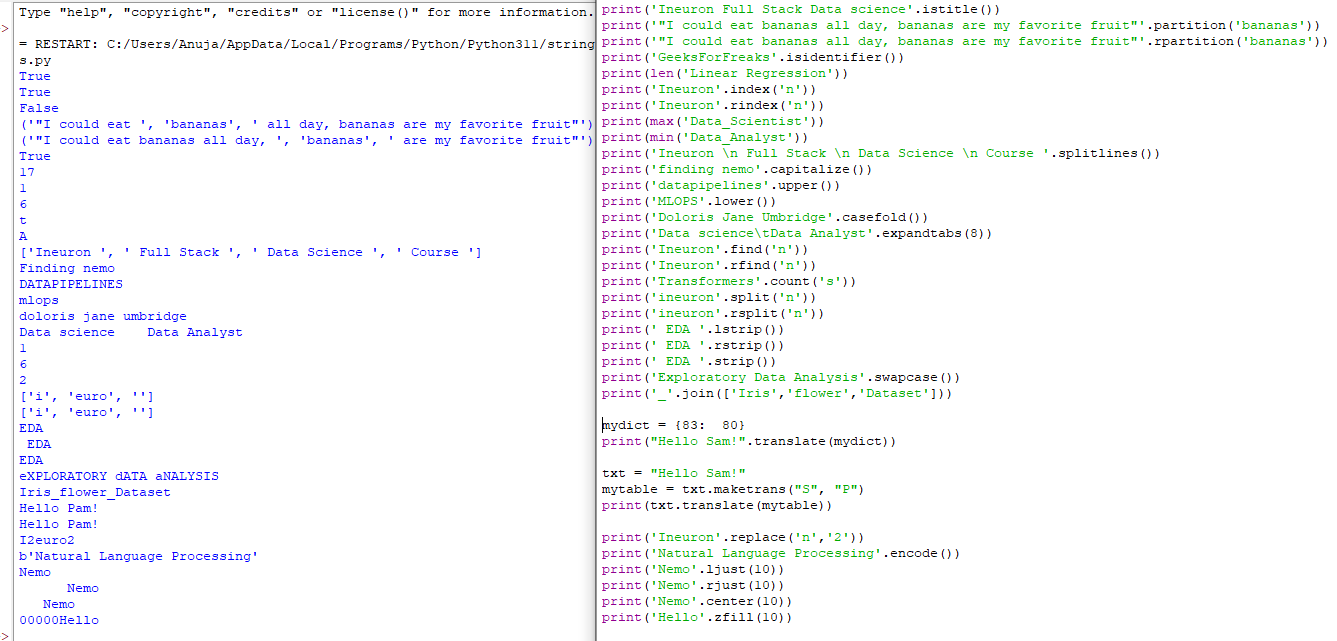
**Q1. In Python 3.X, what are the names and functions of string object types?**

* **<stirng>.isdecimal() ->** Returns True if all characters in a string are decimal.
* **<string>.isalnum() ->** Returns True if all characters in the string are AlphaNumeric.
* **<string>.istitle() ->** Returns True if first character in a string of each word is in Uppercase.
* **<string>.partition(<sub\_string>) ->** Splits string at first occurance of sub string and returns a tuple of 3 elements.
* **<string>.rpartition(<sub\_string>) ->** Splits string at last occurance of sub string and returns a tuple of 3 elements.
* **<string>.isidentifier() ->** Returns True if give string is a valid identifier name.
* **len(<string>) ->** Returns the length of the given string.
* **<string>.index(<sub\_string>) ->** Returns the lowest index of substring if substring is found in the string.
* **<string>.rindex(<sub\_string>) ->** Returns the highest index of substring if substring is found in the string.
* **max(<string>) ->** Returns the highest Alphabetical Character in the string as per ASCII.
* **min(<string>) ->** Returns the lowest Alphabetical Character in the string as per ASCII.
* **<string>.splitlines() ->** Returns a list of lines in the string.
* **<string>.capitalize() ->** Returns the string with first character capitalized.
* **<string>.upper() ->** Returns the string with all characters in uppercase.
* **<string>.lower() ->** Returns the string with all characters in lowercase
* **<string>.casefold() ->** Returns the string in lowercase which can be used for caseless comparisions.
* **<string>.expandtabs(no\_of\_spaces) ->** Replaces tabs in a string with specified no of spaces default is 8
* **<string>.find(<sub\_string>) ->** Returns lowest index of substring if substring is found in the string else returns -1.
* **<string>.rfind(<sub\_string>) ->** Returns highest index of substring if substring is found in the string else returns -1.
* **<string>.count(<char>) ->** Returns the no of occurances of the char in the given string.
* **<string>.split(<sep>) ->** Returns list of words seperated by given sep else seperated by whitespace.
* **<string>.rsplit(<sep>) ->** Returns list of words seperated by given sep else seperated by whitespace scanning from end.
* **<string>.lstrip() ->** Returns a copy of where leading whitespaces are removed.
* **<string>.rstrip() ->** Returns a copy of where trailed whitespaces are removed.
* **<string>.strip() ->** Returns a copy of where both leading and trailing whitespaces are removed.
* **<string>.swapcase() ->** Swaps lowercase characters with uppercase and vice versa.
* **<sep>.join(<list>) ->** Concatenates a list or tuple of words with intervening occuernces of sep.
* **<string>.translate(<mapping\_table>) ->** translates the characters using table.
* **<string>.maketrans(<dict>) ->** Creating a mapping translation tbale usable for **<string>.translate(<mapping\_table>)**
* **<string>.replace(<char\_1>,<char\_2>) ->** Replace all occurances of char\_1 with char\_2 in string.
* **<string>.encode() ->** Encodes string into any encoding supported by python. Default encoding is **UTF-8**.
* **<string>.ljust(<no\_of\_spaces>) ->** Left-justify in a field of given width.
* **<string>.rjust(<no\_of\_spaces>) ->** Right-justify in a field of given width.
* **<string>.center(<no\_of\_spaces>) ->** Center-justify in a field of given width.
* **<stirng>.zfill(<length>) ->** Zfill adds zeros to the begining of string until the specified length is reached.



**Q2. How do the string forms in Python 3.X vary in terms of operations?**

In Python 3, all strings are represented in Unicode.In Python 2 are stored internally as 8-bit ASCII, hence it is required to attach 'u' to make it Unicode. It is no longer necessary now.

**String Operations can be done in three ways:**

1. **Using f-strings**

**Letter “f” is placed before the beginning of the string, and the variables mentioned in curly braces will refer to the variables declared above.**

print("Enter your name")

name = input()

print(f"Hey !! {name}, welcome to the party...")

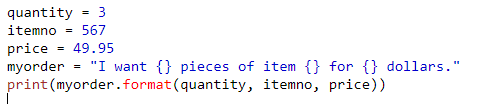
Enter your name

anuja

Hey !! anuja, welcome to the party...

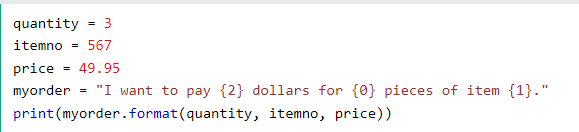
1. **By format() method**

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



I want 3 pieces of item 567 for 49.95 dollars.

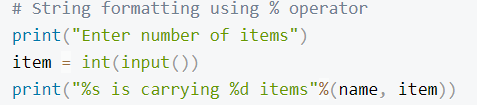
You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

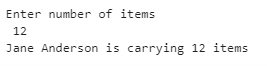


str1 = "{} {} {}".format("Travelling", 'is', 'life')

output – Travelling is life

1. **Using % operator**





**Q3. In 3.X, how do you put non-ASCII Unicode characters in a string?**

This approach is related to the inbuilt library *unidecode*. This library helps Transliterating non-ASCII characters in Python. It provides an u*nidecode()* method that takes Unicode data and tries to represent it in ASCII.  This method automatically determines scripting language and transliterates it accordingly. It accepts unicode string values and returns a transliteration in string format.

**from** unidecode **import** unidecode

print(unidecode(u'ko\u017eu\u0161\u010dek'))

print(unidecode("आप नीचे अपनी भाषा और इनपुट उपकरण चुनें और लिखना आरंभ करें"))

print(unidecode("谢谢你"))

**print**(unidecode("ありがとう。"))

print(unidecode("улыбаться Владимир Путин"))

**Output:**

kozuscek

aap niice apnii bhaassaa aur inputt upkrnn cuneN aur likhnaa aarNbh kreN

Xie Xie Ni

arigatou.

ulybat'sia Vladimir Putin

**Q4. In Python 3.X, what are the key differences between text-mode and binary-mode files?**

|  |  |
| --- | --- |
| **text mode** | **Binary mode** |
| They are human readable | Not human readable |
| Text mode files also support universal end-of-line translation, and encoding specification arguments. | Binary-mode files also accept a bytearray object for the content to be written to the file. |
| Text mode files are stored in sequence of bytes consisting of 0s and 1s. using ASCII, UNICODE or any other encoding scheme, the value of each character of the text file is stored as bytes. So, while opening a text file, the text editor translates each ASCII value and shows us the equivalent character that is readable by the human being. | Binary files are also stored in terms of bytes (0s and 1s), but unlike text files, these bytes do not represent the ASCII values of characters. Rather, they represent the actual content such as image, audio, video, compressed versions of other files, executable files, etc. These files are not human readable. Thus, trying to open a binary file using a text editor will show some garbage values. We need specific software to read or write the contents of a binary file. |

**Q5. How can you interpret a Unicode text file containing text encoded in a different encoding than your platform's default?**

Use of **encode()** and **decode()** method can be used to you interpret a Unicode text file containing text encoded in a different encoding than your platform's default, by default encoding parameter is **UTF-8**

**Q6. What is the best way to make a Unicode text file in a particular encoding format?**

Use **str.encode()** and **file.write()** to make a Unicode text file in a particular encoding format, default encoding format is **UTF-18**.

* Call **str.encode(encoding)** with encoding set to **utf8** to encode str.
* Call **open(file, mode)** to open a file with mode set to **wb** . **wb** writes to files in binary mode & preserves UTF-8format.
* Call **file.write(data)** to write data to the file.

unicode\_text **=** u"улыбаться Владимир Путин"

encoded\_unicode\_text **=** unicode\_text**.**encode("utf-8")

print(encoded\_unicode\_text,end**=**"\n\n")

w\_file **=** open("w\_file.txt","wb")

w\_file**.**write(encoded\_unicode\_text)

w\_file**.**close()

r\_file **=** open("w\_file.txt","rb")

data **=** r\_file**.**read()

print(data)

**Q7. What qualifies ASCII text as a form of Unicode text?**

**Unicode** represents most written languages in the world. **ASCII** has its equivalent in Unicode. The difference between ASCII and Unicode is that ASCII represents lowercase letters (a-z), uppercase letters (A-Z), digits (0–9) and symbols such as punctuation marks while Unicode represents letters of English, Arabic, Greek etc. mathematical symbols, historical scripts, emoji covering a wide range of characters than ASCII.

**Q8. How much of an effect does the change in string types in Python 3.X have on your code?**

Python 3 stores strings as Unicode by default whereas Python 2 requires you to mark a string with a **u** if you want to store it as Unicode. Unicode strings are more versatile than ASCII strings, which are the Python 3.X default, as they can store letters from foreign languages as well as emoji and the standard Roman letters and numerals.