Python Advance Assignment-9

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

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string

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

string.ascii_letters
The concatenation of the ascii_lowercase and ascii_uppercase constants described below. This value is not locale-dependent.

string.ascii_lowercase
The lowercase letters 'abcdefghijklmnopqrstuvwxyz'. This value is not locale-dependent and will not change.

string.ascii_uppercase
The uppercase letters 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'. This value is not locale-dependent and will not change.

string.digits
The string '0123456789'.

string.hexdigits
The string '0123456789abcdefABCDEF'.

string.octdigits
The string '01234567'.

string.punctuation
String of ASCII characters which are considered punctuation characters in the C locale: !"#$%&'()*+,-
./:;<=>?@[\]^_`{|}~.

string.printable
String of ASCII characters which are considered printable. This is a combination of digits, ascii_letters, punctuation, and whitespace.

string.whitespace
A string containing all ASCII characters that are considered whitespace. This includes the characters space, tab, linefeed, return, formfeed, and vertical tab.

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

```
# Import unidecode module from unidecode
from unidecode import unidecode

# Get transliteration for following
# non-ASCII text (Chinese)
print(unidecode("谢谢你"))

# Get transliteration for following
# non-ASCII text (Japanese)
print(unidecode("ありがとう。"))

# Get transliteration for following
# non-ASCII text (Russian)
```

```
print(unidecode("улыбаться Владимир Путин"))
```

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

The two file types may look the same on the surface, but they encode data differently. While both binary and text files contain data stored as a series of bits (binary values of 1s and 0s), the bits in text files represent characters, while the bits in binary files represent custom data.

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

```
def n_possible_values(nbits: int) -> int:
   return 2 ** nbits
```

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

```
unicode_text = u'ƶʒʓʔʕʗʘʙʚʛʜʝʞ'
encoded_unicode = unicode_text.encode("utf8")

a_file = open("textfile.txt", "wb")
a_file.write(encoded_unicode)

a_file = open("textfile.txt", "r")
```
r reads contents of a file

```
contents = a_file.read()

print(contents)
```
OUTPUT
```
ƶʒʓʔʕʗʘʙʚʛʜʝʞ
```

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

The first 128 Unicode code points represent the ASCII characters, which means that any ASCII text is also a UTF-8 text. UCS-2 uses two bytes (16 bits) for each character but can only encode the first 65,536 code points, the so-called Basic Multilingual Plane (BMP).

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

str.isalnum()	String consists of only alphanumeric characters (no symbols)
str.isalpha()	String consists of only alphabetic characters (no symbols)
str.islower()	String’s alphabetic characters are all lower case
str.isnumeric()	String consists of only numeric characters
str.isspace()	String consists of only whitespace characters
str.istitle()	String is in title case
str.isupper()	String’s alphabetic characters are all upper case