Encoding Standards:

* ASCII:
  + American Standard Coding for Information Interchange.
  + Can represent 128 different characters.
* Unicode:
  + More robust than ASCII, can represent most of the world’s writing systems.
* UTF-8:
  + UTF = Unicode Transformation Format.
  + A variable width character encoding used for electronic communications derived from Unicode.
  + ASCII compatible.
* A Code Point is a numerical value that is mapped to a character.

String Operations:

* Escape Sequence “\”
  + It is used to escape characters that otherwise have a special meaning.
  + “\n” – makes a new line
  + “\t” – makes a tab
  + “\b” – makes a backspace
  + The literal backslash is represented by “\\” and only takes up 1 character.
  + “\’” – makes an apostrophe
  + “\”” – makes a single quotation mark.
* Ord()
  + Converts a single string into its Unicode code point
* Chr()
  + Converts an integer Code Point into its corresponding String character.
* Indexing:
  + Strings can be indexed just like any iterable in Python like a List.
* Slicing:
  + Strings can be sliced using up to 3 parameters: [start:end:iteration]
  + Iteration can be set to -1 to reverse the string’s characters.
* Strings are immutable. No characters in the string are allowed to be edited without making a new string.
* When comparing strings, there are two methods:
  + “==” Use this when comparing the actual content of the string.
  + “is” Use this when comparing if two strings are the same object.
  + Note that numbers must be converted into “str” types to be compared or concatenated.
* Built-in string methods:
  + .isupper() / .islower() – returns True if the string only contains uppercase / lowercase letters (ignores digits).
  + .istitle() – returns True if the first character is uppercase and the rest are lowercase.
  + .isalnum() – returns True if the string only contains alphanumeric characters.
  + .isascii() – returns True if the string contains only ASCII characters.
  + .isdigit() – returns True if the string contains only digits.
  + .join() – converts the elements of an iterable into a string.
  + .split() – converts a string into a list split by the given argument.
  + .index() – searches a string for a given substring and returns that substring’s index.
  + .find() – same as .index but does not cause an error if the substring is not found, instead returns -1.
  + .rfind() – same as .find but returns the last position the substring is found.

String Formating:

* String formatting is a way to dynamically incorporate pieces of data in presenting the string.
* Python has 4 different means of string formatting:
  + % operator

print(“This is a %s”%’sentence’)

print(“My friend’s names are %s and %s”%(‘Tom’,’Mandy’))

print(“The price is %1.2f”%23.4151515)

* + format()

print(“This is a {}.”.format(“sentence”))

print(“This {0} is a {1}”.format(“dog”,”golden retriever”))

print(“This shopping list contains {a}, {b}, and {c}.”.format(a = “apple”, b = “orange”, c = “lemon”))

* + f-strings

name = “Jacob”

print(f“My name is {name}.”)

a = 10

b = 5

print(f“The result is {2+(a/b)}.”)

* + String template class

n1 = “Hello”

n2 = “Jacob”

n = Template(“$greeting $name.”)

print(n.substitute(greeting = n1, name = n2))