Ch. 1 Data Types

Tuesday, April 7, 2020 9:36 AM

Jun's Tasks

- [x] Make a python list containing the words I, am, happy and print them out using a for loop
- [x] Use the two different types of for loops in python (for ... in vs for i in range...) to do #1
- [x] Write a python function that takes in a string, and prints out the first and last
- [x] characters
- [x] Write code that creates a new file, and writes any sentence you want into it
- [x] Write another piece of code that opens the file you wrote, and reads the sentence you wrote

Python- Atomic Data Types

https://runestone.academy/runestone/books/published/pythonds/Introduction/GettingStartedwithData.html

Data Types/Objects	Use	Example
Int/float	Numbers with operations	Print(2*4)
Boolean	Truth/False Values	Print (5==10)
Identifiers/variables	= Assignment Statements	theSum = 0
List	Square Bracket, Comma Delimited	[1,True,2]
Strings	Quotes to indicate sequential collection of numbers and letters	"David"
Tuples	Lists, but immutable- parenthesis	(2, True, 4.96)
Set	A heterogeneous collection of immutable Python data objects, curly braces	{3, 6, "cat", 4.5, False}
Dictionaries	A set with associated pair of items, a key and its colon separated value, curly braces	Capitals = {"Iowa" : "DesMoines", "Wisconsin" : "Madison"}

Identifiers
Variables hold References to Data Objects

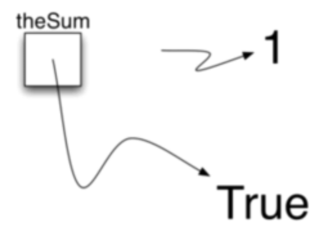


Figure 4: Assignment Changes the Reference

List Operations

Using operations on lists

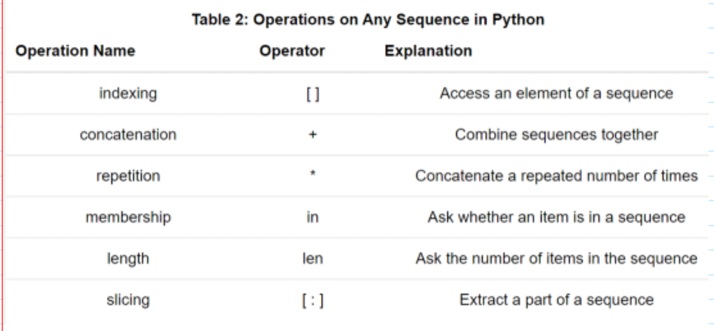


Table 2: Operations on Any Sequence in Python

Operation Name	Operator	Explanation
indexing	[ ]	Access an element of a sequence
concatenation	+	Combine sequences together
repetition	*	Concatenate a repeated number of times
membership	in	Ask whether an item is in a sequence
length	len	Ask the number of items in the sequence
slicing	[:]	Extract a part of a sequence

String Formatting

Python lets you specify the separator character(sep argument, default is a space) as well as an end argument (default is a new line)

Formatted Strings allow you to format things in line (plugging in values)

```
print("%s is %d years old." % (aName, age))
```

- % is a string operator called format operator, they're replaced from left to right
- s, , u, i, e, g, c are conversion characters and give the data type specification depending on the data type you're substituting

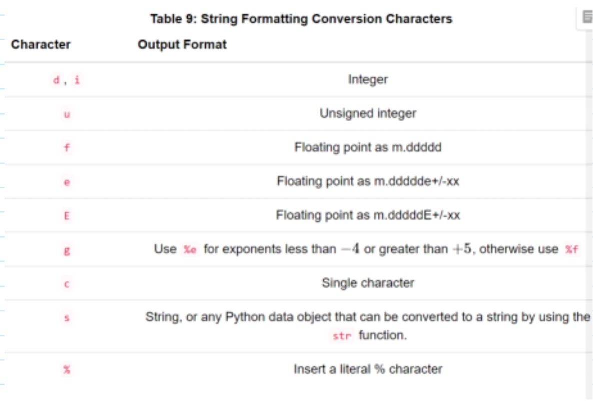


Table 9: String Formatting Conversion Characters

Character	Output Format
d , i	Integer
u	Unsigned integer
f	Floating point as m.ddddd
e	Floating point as m.ddddde+/-xx
E	Floating point as m.dddddE+/-xx
g	Use %e for exponents less than −4 or greater than +5, otherwise use %f
c	Single character
s	String, or any Python data object that can be converted to a string by using the str function.
%	Insert a literal % character

There are additional formatting options that allow you to adjust field width, left or right adjusted, leading zeros, chaacters to the right of the decimals,

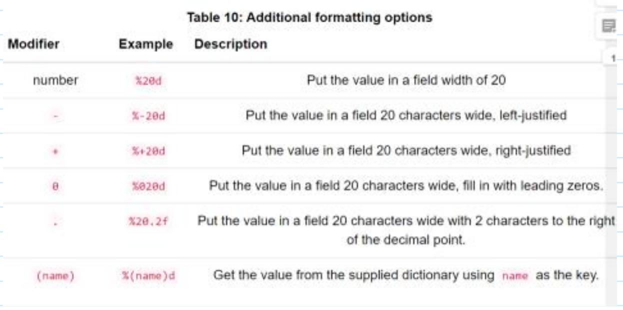
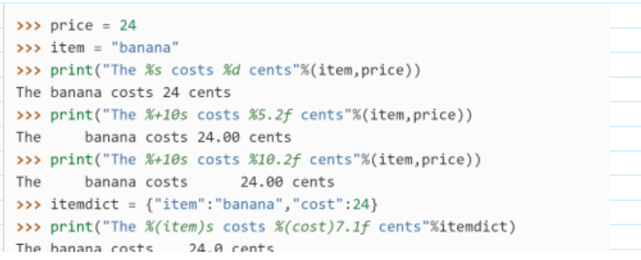


Table 10: Additional formatting options

Modifier	Example	Description
number	%20d	Put the value in a field width of 20
-	%-20d	Put the value in a field 20 characters wide, left-justified
+	%+20d	Put the value in a field 20 characters wide, right-justified
0	%020d	Put the value in a field 20 characters wide, fill in with leading zeros.
.	%20.2f	Put the value in a field 20 characters wide with 2 characters to the right of the decimal point.
(name)	%(name)d	Get the value from the supplied dictionary using name as the key.



```
>>> price = 24
>>> item = "banana"
>>> print("The %s costs %d cents"%(item,price))
The banana costs 24 cents
>>> print("The %+10s costs %5.2f cents"%(item,price))
The     banana costs 24.00 cents
>>> print("The %+10s costs %10.2f cents"%(item,price))
The     banana costs      24.00 cents
>>> itemdict = {"item":"banana","cost":24}
>>> print("The %(item)s costs %(cost)7.1f cents"%itemdict)
The banana costs    24.0 cents
```

Invoking a method on an object using the dot notation:

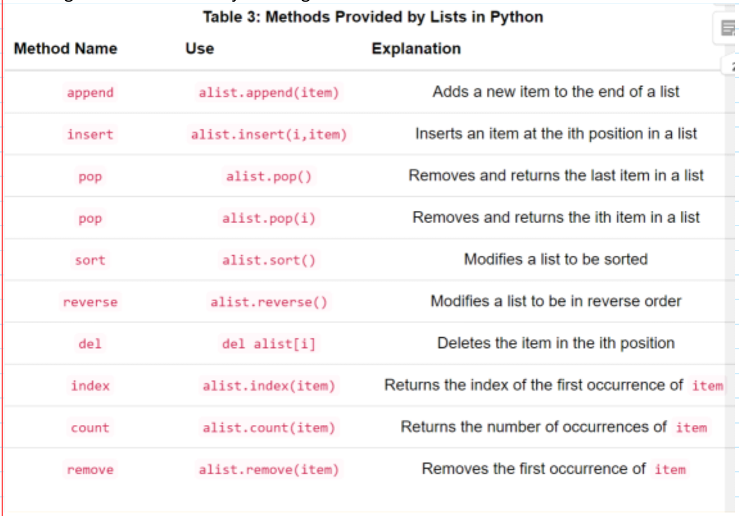


Table 3: Methods Provided by Lists in Python

Method Name	Use	Explanation
`append`	`alist.append(item)`	Adds a new item to the end of a list
`insert`	`alist.insert(i,item)`	Inserts an item at the ith position in a list
`pop`	`alist.pop()`	Removes and returns the last item in a list
`pop`	`alist.pop(i)`	Removes and returns the ith item in a list
`sort`	`alist.sort()`	Modifies a list to be sorted
`reverse`	`alist.reverse()`	Modifies a list to be in reverse order
`del`	`del alist[i]`	Deletes the item in the ith position
`index`	`alist.index(item)`	Returns the index of the first occurrence of `item`
`count`	`alist.count(item)`	Returns the number of occurrences of `item`
`remove`	`alist.remove(item)`	Removes the first occurrence of `item`

Methods on Strings
Lists are mutable; strings are immutable. For example, you can change an item in a list by using indexing and assignment. With a string that change is not allowed

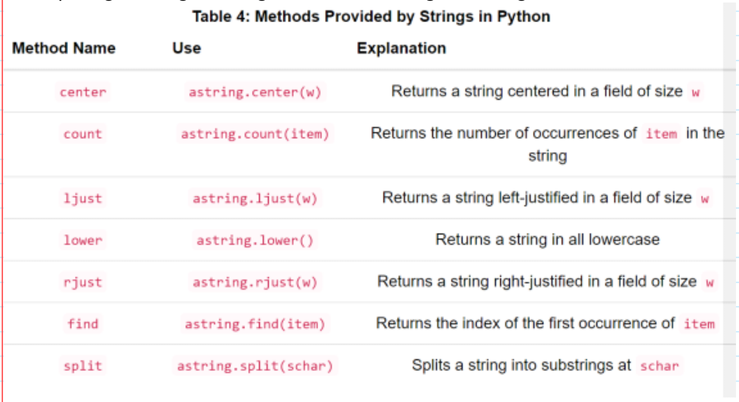


Table 4: Methods Provided by Strings in Python

Method Name	Use	Explanation
`center`	`astring.center(w)`	Returns a string centered in a field of size `w`
`count`	`astring.count(item)`	Returns the number of occurrences of `item` in the string
`ljust`	`astring.ljust(w)`	Returns a string left-justified in a field of size `w`
`lower`	`astring.lower()`	Returns a string in all lowercase
`rjust`	`astring.rjust(w)`	Returns a string right-justified in a field of size `w`
`find`	`astring.find(item)`	Returns the index of the first occurrence of `item`
`split`	`astring.split(schar)`	Splits a string into substrings at `schar`

Set operations and methods

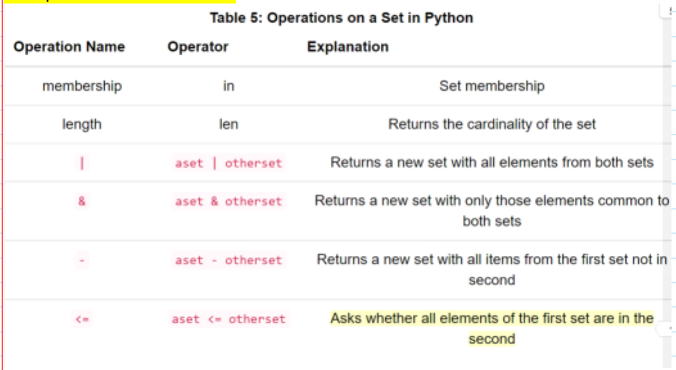
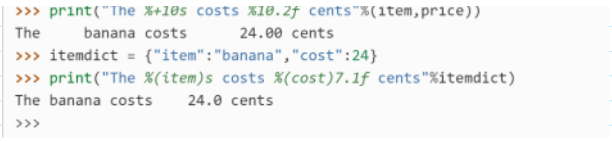


Table 5: Operations on a Set in Python

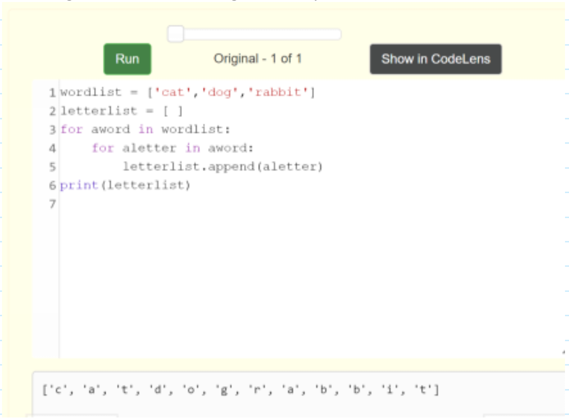
Operation Name	Operator	Explanation
membership	in	Set membership
length	len	Returns the cardinality of the set
`|`	`aset | otherset`	Returns a new set with all elements from both sets
`&`	`aset & otherset`	Returns a new set with only those elements common to both sets
`-`	`aset - otherset`	Returns a new set with all items from the first set not in second
`<=`	`aset <= otherset`	Asks whether all elements of the first set are in the second



```
>>> print("The %+10s costs %10.2f cents"%(item,price))
The     banana costs      24.00 cents
>>> itemdict = {"item":"banana","cost":24}
>>> print("The %(item)s costs %(cost)7.1f cents"%itemdict)
The banana costs    24.0 cents
>>>
```

Using Logic

Breaking down a list of strings into respective letters



```
wordlist = ['cat','dog','rabbit']
letterlist = [ ]
for aword in wordlist:
    for aletter in aword:
        letterlist.append(aletter)
print(letterlist)
```

```
['c', 'a', 't', 'd', 'o', 'g', 'r', 'a', 'b', 'b', 'i', 't']
```

Use if and elif to produce conditional outcomes

```
if score >= 90:
   print('A')
elif score >=80:
   print('B')
elif score >= 70:
   print('C')
elif score >= 60:
   print('D')
else:
   print('F')
```

Invoke methods, loops and conditions all in one line

```
>>> sqlist=[x*x for x in range(1,11) if x%2 != 0]
>>> sqlist
[1, 9, 25, 49, 81]
>>>
```

The variable x takes on the values 1 through 10 as specified by the for construct. The value of x*x is then computed and added to the list that is being constructed. The general syntax for a list comprehension also allows a selection criteria to be added so that only certain items get added. For example,

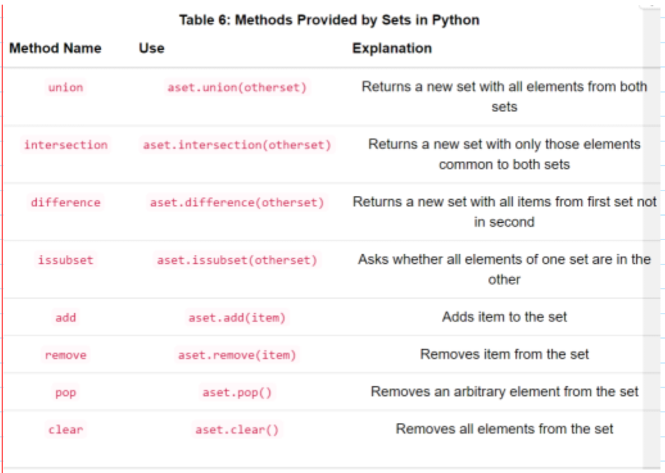


Table 6: Methods Provided by Sets in Python

Method Name	Use	Explanation
`union`	`aset.union(otherset)`	Returns a new set with all elements from both sets
`intersection`	`aset.intersection(otherset)`	Returns a new set with only those elements common to both sets
`difference`	`aset.difference(otherset)`	Returns a new set with all items from first set not in second
`issubset`	`aset.issubset(otherset)`	Asks whether all elements of one set are in the other
`add`	`aset.add(item)`	Adds item to the set
`remove`	`aset.remove(item)`	Removes item from the set
`pop`	`aset.pop()`	Removes an arbitrary element from the set
`clear`	`aset.clear()`	Removes all elements from the set

Dictionary Operations and Methods

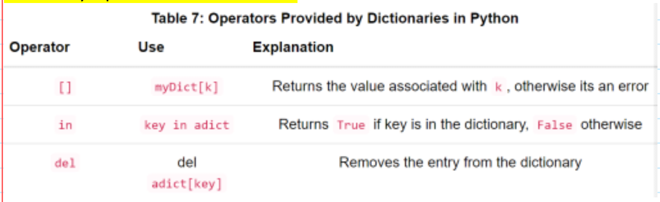


Table 7: Operators Provided by Dictionaries in Python

Operator	Use	Explanation
`[]`	`myDict[k]`	Returns the value associated with `k`, otherwise its an error
`in`	`key in adict`	Returns `True` if key is in the dictionary, `False` otherwise
`del`	del `adict[key]`	Removes the entry from the dictionary

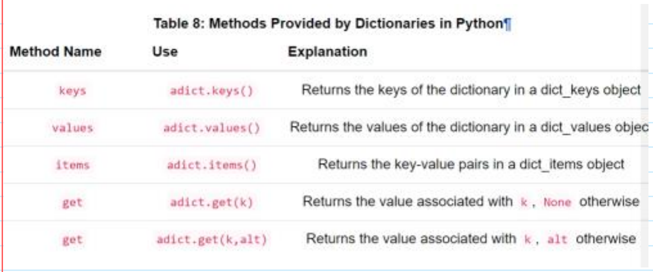


Table 8: Methods Provided by Dictionaries in Python¶

Method Name	Use	Explanation
`keys`	`adict.keys()`	Returns the keys of the dictionary in a dict_keys object
`values`	`adict.values()`	Returns the values of the dictionary in a dict_values objec
`items`	`adict.items()`	Returns the key-value pairs in a dict_items object
`get`	`adict.get(k)`	Returns the value associated with `k`, `None` otherwise
`get`	`adict.get(k,alt)`	Returns the value associated with `k`, `alt` otherwise