COP 3035

Intro Programming in Python

Summer 2024

Lecture 5 – part 1

Lab 2 - Due Date: 05/28/2024 (Today)

Exam 1 – 05/31/2024 (Friday)

Lab 3 - Due Date: 06/03/2024

Homework 2 - Due date: 06/07/2024

Exam 1 – Tips and Topics

- Check document in Canvas

Lecture 5 – part 2

Review

Review

Strings Join() method

Print formatting

- Formatting with placeholders
- Formatting with the `.format()` method
- Formatted String Literals (f-strings)
- Alignment, padding and precision

Object types

Lists

Lists indexing, slicing, concatenation

String join() Method

```
separator.join(iterable)
```

separator: A string that acts as the delimiter. It gets inserted between the elements of the iterable.

iterable: An iterable (e.g., list, tuple, set, dictionary, or even a string) containing the string elements to be joined

The output is a string where consecutive members of the iterable are joined with the separator.

Three ways to do formatting

Formatting with placeholders

```
print('First: %s, Second: %5.2f, Third: %r' %('hi!',3.1415,'bye!'))
```

Formatting with the `.format()` method

```
print('First: {a}, Second: {b}, Third: {c}'.format(a=1,b='Two',c=12.3))
```

Formatted String Literals (f-strings)

```
print(f"My 10 character, four decimal number is:{num:{10}.{6}}")
```

Alignment, padding and precision

```
number = 40.56789

print('  { 0:!^15.3f }  '.format(number))
```

"{<index> : <padding character> <alignment character> <block size> <precision>}"

!!!!40.568!!!!!

https://docs.python.org/3/library/string.html#formatstrings

https://docs.python.org/3/reference/lexical_analysis.html#f-strings

Name	Type	Description
Integers	int	Whole numbers, such as: **3** **300** **200**
Floating point	float	Numbers with a decimal point: 2.3 **4.6** **100.0**
Strings	str	Ordered sequence of characters: **"hello"** **'Sammy'** **"2000"** **"楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Lists

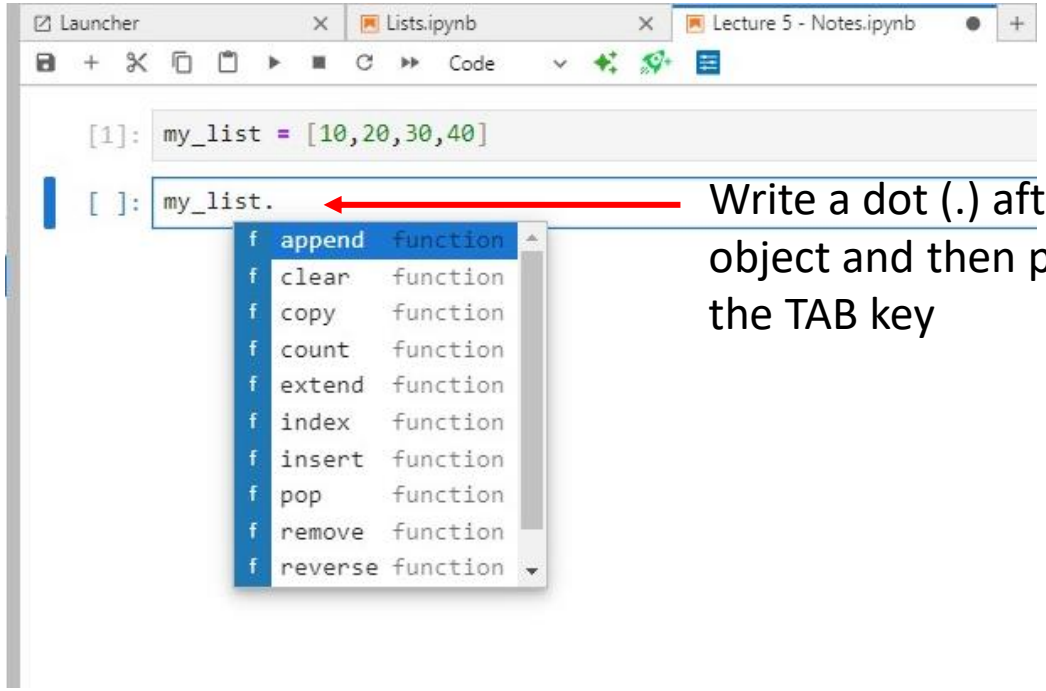
- Lists are ordered sequences that can hold a variety of object types.
- They are denoted by [] brackets and commas to separate objects in the list.
  [1,2,3,4,5]
- Lists support **indexing and slicing**.
- Lists can also be nested and offer a variety of useful methods that can be invoked on them.

Lecture 5 – part 3

List methods

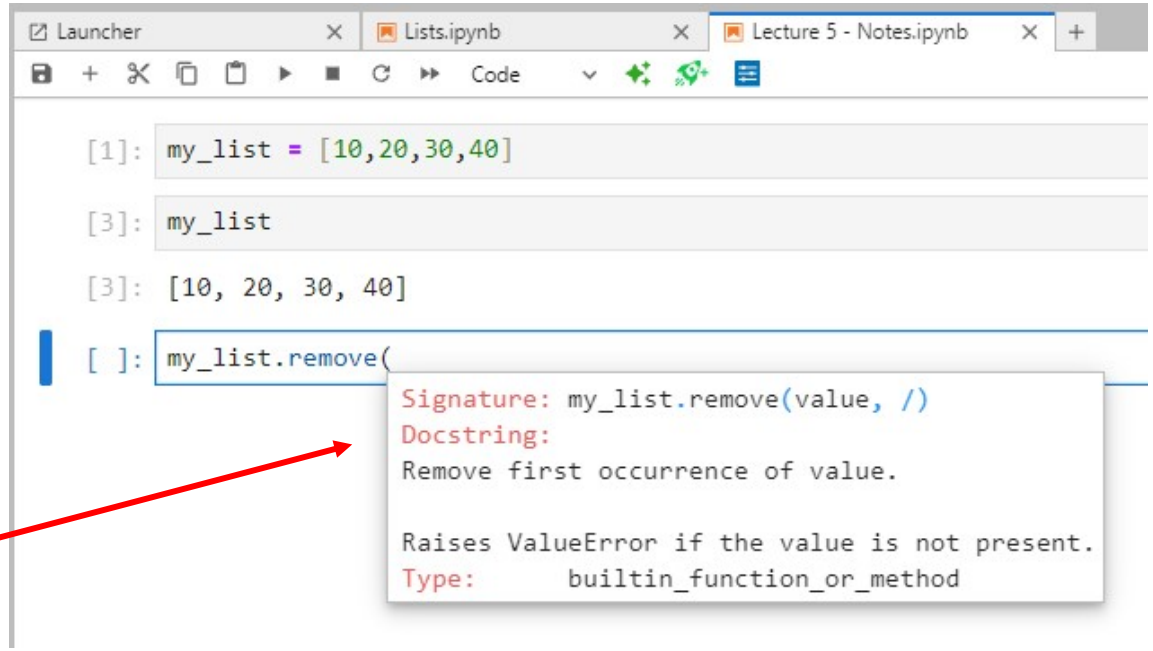
.append(), .pop(), .reverse(), others

Accessing the object built-in functions



Write a dot (.) after the object and then press the TAB key

Accessing the function documentation



Write the function until the first parenthesis and then press the Shift + TAB key

Method	Description	Example
`append()`	Adds an item to the end of the list	`my_list.append(4)`
`extend()`	Extends the list by appending elements from an iterable	`my_list.extend([5, 6])`
`insert()`	Inserts an item at a given position	`my_list.insert(1, 'a')`
`remove()`	Removes the first item with the specified value	`my_list.remove('a')`
`pop()`	Removes and returns the item at the given position	`item = my_list.pop(2)`
`clear()`	Removes all items from the list	`my_list.clear()`
`index()`	Returns the index of the first item with the specified value	`index = my_list.index(5)`
`count()`	Returns the number of items with the specified value	`count = my_list.count(4)`
`sort()`	Sorts the list in ascending order	`my_list.sort()`
`reverse()`	Reverses the elements of the list	`my_list.reverse()`
`copy()`	Returns a shallow copy of the list	`new_list = my_list.copy()`

Lecture 5 – part 4

Dictionaries

Dictionaries

- Dictionaries are unordered mappings for storing objects.
- Dictionaries use a key-value pairing instead.
- This key-value pair allows users to quickly grab objects without needing to know an index location.
- Dictionaries use curly braces and colons to signify the keys and their associated values.

{'key1':'value1','key2':'value2'}

Method	Description	Example
`get()`	Returns the value for a specified key	`value = my_dict.get(key)`
`update()`	Updates the dictionary with elements from another dictionary or iterable	`my_dict.update(other_dict)`
`keys()`	Returns a view object of the dictionary's keys	`keys = my_dict.keys()`
`values()`	Returns a view object of the dictionary's values	`values = my_dict.values()`
`items()`	Returns a view object of the dictionary's key-value pairs	`items = my_dict.items()`
`pop()`	Removes the specified key and returns its value	`value = my_dict.pop(key)`
`popitem()`	Removes and returns the last inserted key-value pair	`key, value = my_dict.popitem()`
`setdefault()`	Returns the value of a key. If the key does not exist, inserts the key with a specified value	`value = my_dict.setdefault(key, default_value)`
`copy()`	Returns a shallow copy of the dictionary	`new_dict = my_dict.copy()`
`clear()`	Removes all items from the dictionary	`my_dict.clear()`

Lecture 5 – part 5

Tuples, sets

Tuples

- Tuples are very similar to lists.
- However, they have one key difference - **immutability**.
- Once an element is inside a tuple, it can not be reassigned.
- Tuples use parenthesis: (1,2,3)

Sets

- Sets are unordered collections of unique elements.
- Meaning there can only be one representative of the same object.

{1,2,3,4,5,’anything’}