ENDG 233 – Programming with Data

Simple and Compound data type

Week 5: Oct. 4th – 10th

Schedule for Week 5

UNIVERSITY OF CALGARY

- Review
- Examples on Compound data type
- portfolio project #1 solution
- zyLabs exercise based on last week's material

Review: definition

UNIVERSITY OF CALGARY

- **Container:** A container is a construct used to group related values together and contains references to other objects instead of data.

- Mutable & Immutable

Review: Common data type

- Numeric type (int, float)
- Sequence type
  - **String:** Used for text
  - **List:** A mutable container with ordered elements
  - **Tuple:** An immutable container with ordered elements
- Set type: A mutable container with unordered and unique elements
- Mapping type (Dict): A container with key-values associated elements

Review

UNIVERSITY OF CALGARY

- python is 0-indexed
  - 0,1, 2
- Build-in functions
  - Ex. len(), join(), split(), etc.
- A programmer can access a character at a specific index by appending brackets [ ] containing the index.
  - Ex. a[2]: return the third one

Review: string

- A string is a sequence of characters
- Strings are immutable and cannot be changed.
- When you use the input() function, the default type is string
- Difference between a = 7 vs. a = ‘7’

Review: string

- **string concatenation:** add new characters to the end of a string in a process

```
string_1 = 'abc'
string_2 = '123'
concatenated_string = string_1 + string_2
print(concatenated_string)
```

abc123

Review: string

Unicode: Python uses Unicode to represent every possible character as a unique number, known as a **code point**.

ord(): returns an encoded integer value for a string of length one.

chr(): returns a string of one character for an encoded integer.

Table 4.2.1: Encoded text values.

Decimal	Character	Decimal	Character	Decimal	Character
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g

Review: string

- **Common escape sequences**
  - **\\:** backslash (\)
  - **\`:** single quote(`)
  - **\”:** double quote(“)
  - **\n:** newline
  - **\t:** Tab (indent)

- Raw String

```
my_string = 'This is a \n \'normal\' string\n'
my_raw_string = r'This is a \n \'raw\' string'

print(my_string)
print(my_raw_string)
```

Output:

```
This is a
 'normal' string

This is a \n \'raw\' string
```

Review: string

UNIVERSITY OF CALGARY

String formatting

- Positional replacement
  - 'The {1} in the {0} is {2}.'.format('hat', 'cat', 'fat')
  - The cat in the hat is fat.
- Inferred positional replacement
  - 'The {} in the {} is {}.'.format('cat', 'hat', 'fat')
  - The cat in the hat is fat.
- Named replacement
  - 'The {animal} in the {headwear} is {shape}.'.format(animal='cat', headwear='hat', shape='fat')
  - The cat in the hat is fat.

Review: formatting

UNIVERSITY OF CALGARY

- Common formatting specification presentation types.

‘{:□}’.format{variable}

Type	Description	Example	Output
s	String (default presentation type - can be omitted)	`'{:s}'.format('Aiden')`	`Aiden`
d	Decimal (integer values only)	`'{:d}'.format(4)`	`4`
b	Binary (integer values only)	`'{:b}'.format(4)`	`100`
x, X	Hexadecimal in lowercase (x) and uppercase (X) (integer values only)	`'{:x}'.format(15)`	`f`
e	Exponent notation	`'{:e}'.format(44)`	`4.400000e+01`
f	Fixed-point notation (6 places of precision)	`'{:f}'.format(4)`	`4.000000`
.[precision]f	Fixed-point notation (programmer-defined precision)	`'{:.2f}'.format(4)`	`4.00`
0[precision]d	Leading 0 notation	`'{:03d}'.format(4)`	`004`

Tutorial 5.1 – Longest String

UNIVERSITY OF CALGARY

- **Task**: Write a program that takes in two strings and outputs the longest string
- If they are the same length then output the second string
- Ex. If the input is:

```
almond
pistachio
```

The output is:

```
pistachio
```

Tutorial 5.1 – Longest String

UNIVERSITY OF CALGARY

Solution:

```
# input string1 and string2
str1 = input()
str2 = input()
# compare the length of strings
if len(str1) > len(str2):
        print(str1)
else:
        print(str2)
```

Review: Lists

UNIVERSITY OF CALGARY

- A list is a built-in data structure that groups together variables of the same type or mixed variable types
- To define a list:
  - **list1** = **[**element1, element2, element3,…**]**
- List is **mutable**
- As with strings, there are many different built-in functions that you can use for lists

- **Reminder**: list index starts at 0, not 1

Review: List functions

- **Concatenate** (add two lists together)
  - list3 = list1 **+** list2
- **Append** (add one element to the end of the list)
  - list1.**append**(new_element)
- **Find** the **index** of a specific element
  - index1 = list1.**index**[element]
- **Replace** element in a list
  - list1**[index1]** = new_element
- **Insert** element at desired index
  - list1.**insert**(index1, element)

Review: List Functions (cont'd)

UNIVERSITY OF CALGARY

- **Sort** list elements
  - list1.**sort**()
- **Remove** specific element
  - list1.**remove**(element)
- **Remove** element at desired index
  - **del** list1[index1]
  - list1.**pop**(index1)
- Length of list = **len**(list1)
- Number of a specific element = list1.**count**(element)

Review: List Functions (cont'd)

UNIVERSITY OF CALGARY

- Find the element in list with the **smallest value**
  - **min**(list)
- Find the element in list with the **largest value**
  - **max**(list)
- Find the **sum** of all elements of a list (numbers only)
  - **sum**(list)

Tutorial 5.2 – List Basics

UNIVERSITY OF CALGARY

- Given the user inputs, complete a program that does the following tasks:
  - Define a list, **my_list**, containing the user inputs: **my_flower1**, **my_flower2**, and **my_flower3** in the same order
  - Define a list, **your_list**, containing the user inputs, **your_flower1** and **your_flower2**, in the same order
  - Define a list, **our_list**, by concatenating **my_list** and **your_list**
  - Append the user input, **their_flower**, to the end of **our_list**
  - Replace **my_flower2** in **our_list** with **their_flower**
  - Remove the first occurrence of **their_flower** from **our_list** without using index()
  - Remove the second element of **our_list**
- Observe the output of each print statement carefully to understand what was done by each task of the program

Tutorial 5.2 – List Basics

UNIVERSITY OF CALGARY

- Ex: If the input is:

```
rose peony lily rose daisy aster
```

- The output is:

```
['rose', 'peony', 'lily', 'rose', 'daisy’]
['rose', 'peony', 'lily', 'rose', 'daisy', 'aster’]
['rose', 'aster', 'lily', 'rose', 'daisy', 'aster’]
['rose', 'lily', 'rose', 'daisy', 'aster’]
['rose', 'rose', 'daisy', 'aster']
```

Tutorial 5.2 – List Basics

UNIVERSITY OF CALGARY

Solution:

```
# input the list values
my_flower1 = input()
my_flower2 = input()
my_flower3 = input()
your_flower1 = input()
your_flower2 = input()
their_flower = input()

# Define my_list containing my_flower1, my_flower2, and my_flower3 in that order
my_list = [my_flower1, my_flower2, my_flower3]

# Define your_list containing your_flower1 and your_flower2 in that order
your_list =  [your_flower1, your_flower2]
```

Tutorial 5.2 – List Basics

UNIVERSITY OF CALGARY

Solution:

```
# Define our_list by concatenating my_list and your_list
our_list = my_list + your_list
print(our_list)

# Append their_flower to the end of our_list
our_list.append(their_flower)
print(our_list)

# Replace my_flower2 in our_list with their_flower
index = our_list.index(my_flower2)
our_list[index] = their_flower
print(our_list)
```

Tutorial 5.2 – List Basics

UNIVERSITY OF CALGARY

Solution:

```
# Remove the first occurrence of their_flower from our_list without using index()
our_list.remove(their_flower)
print(our_list)

# Remove the second element of our_list
our_list.pop(1)
print(our_list)                    # Alternatively, use: del our_list[1]
```

Review: Data Types

- **List** is a collection which is ordered and changeable. Allows for duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows for duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members allowed.
- **Dictionary** is a collection which is ordered and changeable. No duplicate members allowed.

Review: Lists vs. Sets

- A list can have duplicate elements, whereas a set doesn't
- The elements in a set are unordered and unindexed

Ex: (from https://towardsdatascience.com/)

```
text = "Hello World!"
print(list(text))     # Will contain duplicates, in order of text
['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!']
print(set(text))      # Removes duplicates, no specific order
{'H', 'W', 'o', ' ', 'l', 'r', '!', 'e', 'd'}
```

Review: Lists vs. Sets vs. Tuples

UNIVERSITY OF CALGARY

- Tuples are not changeable, unlike lists and sets
- Tuples are ordered, unlike sets
- Can use typecasting to switch between the three types
- Can add two lists or tuples using “+”, but can’t add two sets using the same method
- To add two sets together use: set1 = set1.union(set2) or set1.update(set2)
- To append an item to a set, use the add() function

Review: Lists vs. Sets vs. Tuples (cont'd)

UNIVERSITY OF CALGARY

- Can use the **remove**() function for sets
  - When using pop(), removes random element from set
- Can use: list1 = **sorted**(tuple1) to sort a tuple into a separate list variable as tuples can't be changed
- Can use the **len**() function for all three
- Can use **count**() function for lists and tuples, since duplicate elements are allowed
  - Returns 1 if used for sets as all values are unique

Review: Lists vs. Sets vs. Tuples

UNIVERSITY OF CALGARY

	Lists	Sets	Tuples
Mutable	Yes	Yes	No
Ordered	Yes	No	Yes
duplication	Yes	No	Yes
Indexed	Yes	No	Yes
Adding two	+	set1.union(set2)	+
append	.append()	.add()	N/A
Len	Yes	Yes	Yes
Count	Yes	No	Yes

Review: Dictionaries

UNIVERSITY OF CALGARY

- **Dictionaries** are similar to sets, except that dictionaries are used to store values in **key:value** pairs, and are ordered and changeable
- Ex:

```
car_dict = {
  "brand":"Ford",
  "model":"Mustang",
  "year":1964
}
```

Review: Dictionaries (cont'd)

UNIVERSITY OF CALGARY

- **Add** a new dictionary **key:value** pair:
  - car_dict["color"] = "Blue"
- **Replace** a value:
  - car_dict["model"] = "Focus"
- To delete an element, use **del** car_dict["color"]
- If you want to print the keys in the entered order, use **list**(car_dict)
- If you want to print the sorted list of keys, use **sorted**(car_dict)

Review: Dictionaries (cont'd)

EX

```
prices = {}  # Create empty dictionary
prices['banana'] = 1.49  # Add new entry
print(prices)

prices['banana'] = 1.69  # Modify entry
print(prices)

del prices['banana']  # Remove entry
print(prices)
```

Output:

```
{'banana': 1.49}
{'banana': 1.69}
{}
```

Review: Dictionaries (cont'd)

```
service_dict = {
    "oil change": 35,
    "tire rotation": 19,
    "car wash": 7,
    }

# menu on screen
print('Welcome to ENDG_AutoServices. We are happy to provide
the following services:\n')
print(f'1. Oil Change\t\t$ {service_dict["oil change"]}')
print(f'2. Tire Rotation\t$ {service_dict["tire rotation"]}')
print(f'3. Car Wash\t\t$ {service_dict["car wash"]}')
print()
```

Review: Dictionaries (cont'd)

```
#input service
service = input('Please enter requested service :\n')
# condition check
if service in service_dict.keys():
    print(f'Service requested:\t{service}')
    print(f'Service cost:\t\t${service_dict[service]}')
    print(f'Total cost:\t\t${service_dict[service]*1.05:.2f}')
else:
    print('Service not available')
```

Review

- Membership operators: **in and not in**
  - Ex. X in Y True or False
- Identity operator: **is**
  - Ex: a is b True or False

Review

- **Type conversions:** A type conversion is a conversion of one type to another, such as an int to a float.

- **implicit conversion:** An implicit conversion is a type conversion automatically made by the interpreter, usually between numeric types.

Function	Notes	Can convert:
int()	Creates integers	int, float, strings w/ integers only
float()	Creates floats	int, float, strings w/ integers or fractions
str()	Creates strings	Any

Tutorial 5.3 – Set Basics

UNIVERSITY OF CALGARY

- Given the user inputs, complete a program that does the following tasks:
  - Define a set, **fruits**, containing the user inputs: **my_fruit1**, **my_fruit2**, and **my_fruit3**
  - Add the user inputs, **your_fruit1** and **your_fruit2**, to **fruits**
  - Add the user input, **their_fruit**, to **fruits**
  - Add **your_fruit1** to **fruits**
  - Remove **my_fruit1** from **fruits**
- Observe the output of each print statement carefully to understand what was done by each task of the program
- Note: For testing purposes, sets are printed using **sorted()** for comparison, as in the book's examples

Tutorial 5.3 – Set Basics

UNIVERSITY OF CALGARY

- Ex: If the input is:

```
apple peach lemon apple pear plum
```

- the output is:

```
['apple', 'lemon', 'peach']
['apple', 'lemon', 'peach', 'pear']
['apple', 'lemon', 'peach', 'pear', 'plum']
['apple', 'lemon', 'peach', 'pear', 'plum']
['lemon', 'peach', 'pear', 'plum']
```

Tutorial 5.3 – Set Basics

UNIVERSITY OF CALGARY

Solution:

```
# input the values
my_fruit1 = input()
my_fruit2 = input()
my_fruit3 = input()
your_fruit1 = input()
your_fruit2 = input()
their_fruit = input()

# Define a set, fruits, containing my_fruit1, my_fruit2, and my_fruit3
fruits = {my_fruit1, my_fruit2, my_fruit3}
print(sorted(fruits))
# Add your_fruit1 and your_fruit2 to fruits
fruits.update({your_fruit1, your_fruit2})
print(sorted(fruits))
```

Tutorial 5.3 – Set Basics

UNIVERSITY OF CALGARY

Solution:

```
# Add their_fruit to fruits
fruits.add(their_fruit)
print(sorted(fruits))

# Add your_fruit1 to fruits
fruits.add(your_fruit1)
print(sorted(fruits))

# Remove my_fruit1 from fruits
fruits.remove(my_fruit1)
print(sorted(fruits))
```

Exercise 5.4: Car Wash Costs

UNIVERSITY OF CALGARY

Task: Write a program to calculate the total price for car wash services. A base car wash is $10.

A dictionary with each additional service and the corresponding cost has been provided. Two additional services can be selected.

A '-' signifies an additional service was not selected.

Output all selected services along with the corresponding costs and then the total price for all car wash services.

Exercise 5.4: Car Wash Costs

UNIVERSITY OF CALGARY

Ex. If input is

Tire shine
Wax

Output :

ZyCar Wash
Base car wash -- $10
Tire shine -- $2
Wax -- $3

Total price: $15