3.1 String basics

Strings and string literals

©zyBooks 03/05/20 10:22 59141 Alexey Munishkin

A **string** is a sequence of characters, like the text MARY, that can be stored in a variable. A **string literal** is a string value specified in the source code of a program. A programmer creates a string literal by surrounding text with single or double quotes, such as 'MARY' or "MARY".

The string type is a special construct known as a **sequence type**: A type that specifies a collection of objects ordered from left to right. A string's characters are ordered from the string's first letter to the last. A character's position in a string is called the character's index, which starts at 0. Ex: In "Trish", T is at index 0, r at 1, etc.

PARTICIPATION ACTIVITY	3.1.1: String	ind	exing].					
Type a string to characters orderecters characters	lered by posit			0				ate	
Type a st (up to 6 c	ring characters)	0 T	1 r	2 i	3 s	4 h	5		
111511									

A programmer can assign a string just as with other types. Ex: str1 = 'Hello', or str1 = str2. The input() function can also be used to get strings from the user.

An empty string is a sequence type with 0 elements, created with two quotes. Ex:

my str = ''.

zyDE 3.1.1: A program with strings.

Try the 'mad libs' style game below.

Load default template...

		brother burritos macho
		Run
		ks 03/05/20 10:22 591419 Alexey Munishkin CSE20NawabWinter2020
2 3 4	#A 'Mad Libs' style game where user enters r #verbs, etc., and then a story using those v #Get user's words	
5 6	<pre>relative = input('Enter a type of relative: print()</pre>	
7 8	, , , , , , , , , , , , , , , , , , ,	>
12	print() 1 adjective <u>literal</u> s('Enter an adjective: ') print()	
13 Indicate which ite	neriod string literals. a time period: ') print()	
1) 'Hey' 17 18 O Yes 19	<pre># Tell the story print('My', relative, 'says eating', food) print('will make me more', adjective)</pre>	
O No 21	<pre>print('so now I eat it every', period)</pre>	
2) 'Hey there.'		
O Yes		
O No		
3) 674		
O Yes		ks 03/05/20 10:22 591419
O No		Alexey Munishkin CSE20NawabWinter2020
4) '674'		
O Yes		
O No		
5) "ok"		

O Yes	
O No	
6) "a"	
O Yes	
O No	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
PARTICIPATION 3.1.3: String basics.	
 Which creates a string variable first_name with a value 'Daniel'? 	
<pre>O Daniel = first_name</pre>	
<pre>O first_name = 'Daniel'</pre>	
O first_name = Daniel	
2) Which prints the value of the first_name variable?	
<pre>O print(first_name)</pre>	
O print('first_name')	
<pre>O print("first_name")</pre>	
3) Which assigns a string read from input to first_name?	
<pre>O first_name = input</pre>	
<pre>O input('Type your name:')</pre>	
<pre>O first_name = input('Type your name:')</pre>	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
4) Which answer assigns an empty string to first_name?	
O first_name =	
O first_name = ''	

```
O '' = first_name
```

String length and indexing

A common operation is to find the length, or the number of characters, in a string. The **len()** built-in function can be used to find the length of a string (and any other sequence type).

Alexey Munishkin

Figure 3.1.1: Using len() to get the length of a string.

The \ character after the string literal extends the string to the following line.

```
george_v = "His Majesty George V, by the Grace
of God, " \
           "of the United Kingdom of Great
Britain and " \
           "Ireland and of the British
Dominions beyond " \
                                                     185 characters is much too
           "the Seas, King, Defender of the
                                                     long of a name!
Faith, Emperor of India"
                                                     26 characters is better...
gandhi = 'Mohandas Karamchand Gandhi'
                                                     3 characters is short
john_f_kennedy = 'JFK'
                                                     enough.
print(len(george_v), 'characters is much too
long of a name!')
print(len(gandhi), 'characters is better...')
print(len(john_f_kennedy), 'characters is short
enough.')
```

PARTICIPATION ACTIVITY	3.1.4: Using len() to find the length of a string.	
1) What is the "Santa"?	e length of the string	
		©zyBooks 03/05/20 10:22 591419 Alexey Munishkin
Check	Show answer	UCSCCSE20NawabWinter2020
2) Print the le	ngth of the string st_name.	
<	>	

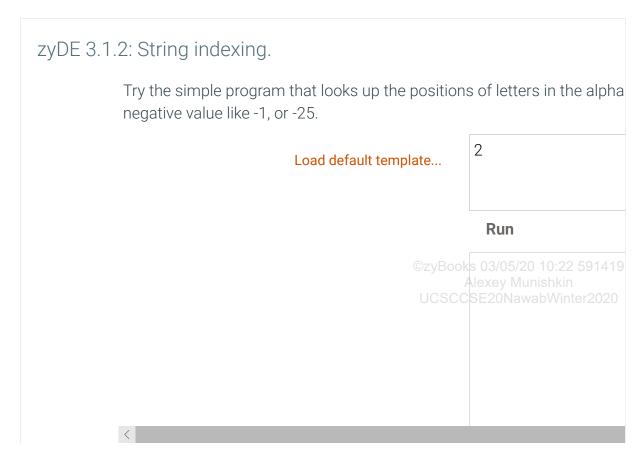
Check Show answer

Programs commonly access an individual character of a string. As a sequence type, every character in a string has an index, or position, starting at 0 from the leftmost character. For example, the 'A' in string 'ABC' is at index 0, 'B' is at index 1, and 'C' is at index 2. A programmer can access a character at a specific index by appending **brackets** [] containing the index:

Figure 3.1.2: Accessing individual characters of a string.

```
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
print(alphabet[0], alphabet[1], alphabet[25])
A B Z
```

Note that negative indices can be used to access characters starting from the rightmost character of the string, instead of the leftmost.

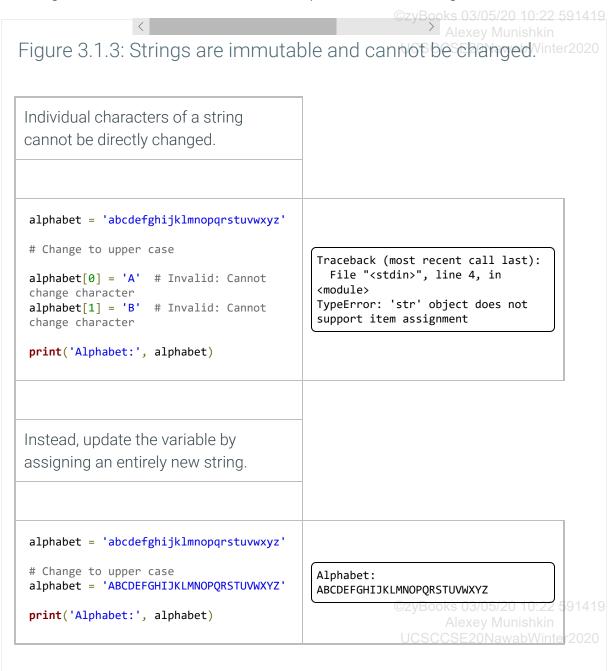


	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
<	>
PARTICIPATION 3.1.5: String indexing.	
1) What character is in index 2 of the string "America"?	
Check Show answer	
2) Write an expression that accesses the first character of the string my_country.	
Check Show answer	
3) Assign my_var with the last character in my_str. Use a negative index.	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2920
Check Show answer	

```
1 alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

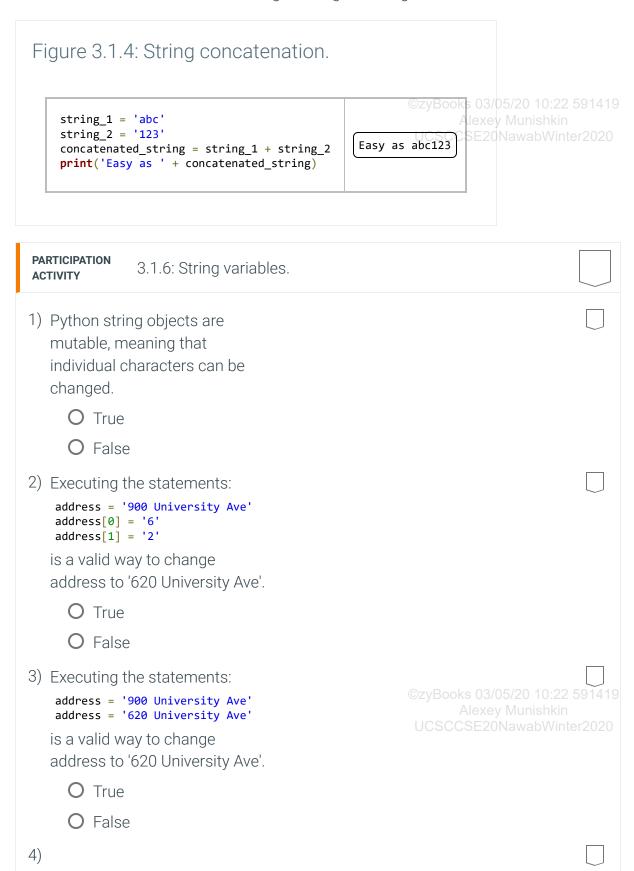
Changing string variables and concatenating strings
3 user_number = int(input('Enter number to use a
4 print()
```

Writing or altering individual characters of a string variable is not allowed. Strings are immutable objects, meaning that string values cannot change once created. Instead, an assignment statement must be used to update an entire string variable.



A program can add new characters to the end of a string in a process known as **string concatenation**. The expression "New" + "York" concatenates the strings New and York to create a new string NewYork. Most sequence types support concatenation.

String concatenation does not contradict the immutability of strings, because the result of concatenation is a new string; the original strings are not altered.



After the following executes,
the value of address is '500
Floral Avenue'.

street_num = '500'
street = 'Floral Avenue'
address = street_num + ' ' +
street

O True

O False

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
UCSCCSE20NawabWinter2020

CHALLENGE ACTIVITY

3.1.1: Reading multiple data types.

Type two statements. The first reads user input into person_name. The second reads user input into person_age. Use the int() function to convert person_age into an integer. Below is a sample output for the given program if the user's input is: Amy 4

In 5 years Amy will be 9

Note: Do not write a prompt for the input values.

```
1 person_name = ''
2 person_age = 0
3
4 ''' Your solution goes here '''
5
6 print()'In 5 years', person_name, 'will be', person_age + 5)|
```

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
UCSCCSE20NawabWinter2020

Run

CHALLENGE **ACTIVITY**

3.1.2: Concatenating strings.

Write two statements to read in values for my_city followed by my_state. Do in not provide a prompt. Assign log_entry with current_time, my_city, and wabWinter2020 my_state. Values should be separated by a space. Sample output for given program if my_city is Houston and my_state is Texas:

2014-07-26 02:12:18: Houston Texas

Note: Do not write a prompt for the input values.

```
1 current_time = '2014-07-26 02:12:18:'
2 my_city = ''
3 my_state = ''
4 log_entry = ''
6 ''' Your solution goes here '''
8 print(log_entry)
```

Run

↓ Download student submissions

3.2 List basics

Creating a list

©zvBooks 03/05/20 10:22 591419

A **container** is a construct used to group related values together and contains references to other objects instead of data. A **list** is a container created by surrounding a sequence of variables or literals with brackets []. Ex: my_list = [10, 'abc'] creates a new list variable my_list that contains the two items: 10 and 'abc'. A list item is called an **element**.

A list is also a sequence, meaning the contained elements are ordered by position in the list, known as the element's **index**, starting with 0. my_list = []. creates an empty list.

The animation below shows how a list is created and managed by the interpreter. A list itself is an object, and its value is a sequence of references to the list's elements.

PARTICIPATION ACTIVITY

3.2.1: Creating lists.

Animation captions:

- 1. User creates a new list.
- 2. The interpreter creates new object for each list element.
- 3. 'prices' holds references to objects in list.

zyDE 3.2.1: Creating lists.

The following program prints a list of names. Try adding your name to th program again.

©zyBooks 03/05/20 10:22 591419

Alexey Munishkin Load default template CSCSE 200 Nawab Winter 2020

	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCC SE20NawabWinter2020
<	>
ACTIVITY 3.2.2: Creating lists.	
<pre>1) Write a statement that creates a list called my_nums, containing the elements 5, 10, and 20. 1 names = ['Daniel', 'Roxanna', 'Jean']</pre>	
2) Write a statement that creates a list called my_list with the elements -100 and the string 'lists are fun'.	
	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin
Check Show answer	UCSCCSE20NawabWinter2020
3) Write a statement that creates an empty list called class_grades.	

Check Show answer

Accessing list elements

©zvBooks 03/05/20 10:22 591419

Lists are useful for reducing the number of variables in a program. Instead of having a separate variable for the name of every student in a class, or for every word in an email, a single list can store an entire collection of related variables.

Individual list elements can be accessed using an indexing expression by using brackets as in my_list[i], where i is an integer. This allows a programmer to quickly find the i'th element in a list

Figure 3.2.1: Access list elements using an indexing expression.

```
# Some of the most expensive cars in the
lamborghini_veneno = 3900000 # $3.9 million!
bugatti veyron = 2400000 # $2.4 million!
aston_martin_one77 = 1850000 # $1.85
million!
                                                  Lamborghini Veneno: 3900000
                                                  dollars
prices = [lamborghini_veneno, bugatti_veyron,
                                                  Bugatti Veyron Super Sport:
aston_martin_one77]
                                                  2400000 dollars
                                                  Aston Martin One-77: 1850000
print('Lamborghini Veneno:', prices[0],
                                                  dollars
'dollars')
print('Bugatti Veyron Super Sport:',
prices[1], 'dollars')
print('Aston Martin One-77:', prices[2],
'dollars')
```

©zvBooks 03/05/20 10:22 591419

List elements can also be updated with new values by performing an assignment to a position in the list.

A list's index must be an integer. The index cannot be a floating-point type, even if the value is a whole number like 0.0 or 1.0. Using any type besides an integer will produce a runtime error and the program will terminate.

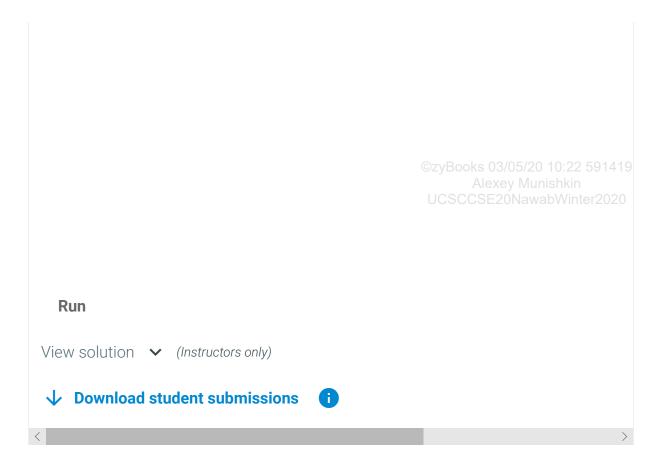
```
my_nums = [5, 12, 20]
print(my_nums)

# Modify a list element
my_nums[1] = -28
print (my_nums)

[5, 12, 20]
[5, -28, 20]
```

©zyBooks 03/05/20 10:22 591419

		UCSCCSE20NawabWinter2020
PARTICIPATION ACTIVITY	3.2.3: Accessing list	
	tement that assigns the the 3rd element of the 3rd	
	tement that assigns ement of my_towns it'.	
Check	Show answer	
CHALLENGE ACTIVITY	3.2.1: Initialize a list.	
	st short_names with st given program:	rings 'Gus', 'Bob', and 'Zoe'. Sample
Gus Bob Zoe		©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020



Adding and removing list elements

Lists are mutable, meaning that a programmer can use methods to add and remove elements from a list as needed. A **method** instructs an object to perform some action, and is executed by specifying the method name following a "." symbol and an object. The **append()** list method is used to add new elements to a list. Elements can be removed using the **pop()** or **remove()** methods. Methods are covered in greater detail in another section.

Adding elements to a list:

list.append(value): Adds value to the end of list. Ex: my_list.append('abc')

Removing elements from a list:

©zyBooks 03/05/20 10:22 591419

- list.pop(i): Removes the element at index i from list. Ex: my_list.pop(1)
- list.remove(v): Removes the first element whose value is v. Ex: my_list.remove('abc')

PARTICIPATION ACTIVITY

3.2.4: Adding and removing list elements.

Animation content: undefined **Animation captions:** 1. append() adds an element to the end of the list. ©zyBooks 03/05/20 10:22 591419 2. pop() removes the element at the given index from the list. bw', which is at index 1, is removed and 'abc' is now at index 1. 3. remove() removes the first element with a given value. 'abc' is removed and now the list only has one element. **PARTICIPATION** 3.2.5: List modification. **ACTIVITY** Write a statement that performs the desired action. Assume the list house_prices = ['\$140,000', '\$550,000', '\$480,000'] exists. 1) Update the price of the second item in house_prices to '\$175,000'. Check **Show answer** 2) Add a price to the end of the list with a value of '\$1,000,000'. Check **Show answer** 3) Remove the 1st element from ©zyBooks 03/05/20 10:22 59 house_prices, using the pop() method. Check Show answer 4) Remove '\$140,000' from house_prices,

using the remove() method.

Sequence-type methods and functions

©zyBooks 03/05/20 10:22 591419 Alexey Munishkin

Sequence-type functions are built-in functions that operate on sequences like lists and strings. **Sequence-type methods** are methods built into the class definitions of sequences like lists and strings. A subset of such functions and methods is provided below.

Table 3.2.1: Some of the functions and methods useful to lists.

Operation	Description
len(list)	Find the length of the list.
list1 + list2	Produce a new list by concatenating list2 to the end of list1.
min(list)	Find the element in list with the smallest value.
max(list)	Find the element in list with the largest value.
sum(list)	Find the sum of all elements of a list (numbers only).
list.index(val)	Find the index of the first element in list whose value matches val.
list.count(val)	Count the number of occurrences of the value val in list.

©zyBooks 03/05/20 10:22 591419 Alexey Munishkin

UCSCCSE20NawabWinter2020

Figure 3.2.3: Using sequence-type functions with lists.

```
# Concatenating lists
house_prices = [380000, 900000, 875000] + [225000]
print('There are', len(house_prices), 'prices in the
list.')

# Finding min, max
print('Cheapest house:', min(house_prices))
print('Most expensive house:', max(house_prices))

# ZyBooks 03/05/20 10:22 5914 9
Alexey Munishkin
UCSCCSE20NawabWinter2020
```

Note that lists can contain mixed types of objects. Ex: x = [1, 2.5, 'abc'] creates a new list x that contains an integer, a floating-point number, and a string. Later material explores lists in detail, including how lists can even contain other lists as elements.

zyDE 3.2.2: Student grade statistics.

The following program calculates some information regarding final and renhancing the program by calculating the average midterm and final scc

```
Lc
 1 #Program to calculate statistics from student test scores.
 2 midterm_scores = [99.5, 78.25, 76, 58.5, 100, 87.5, 91, 68, 100]
 3 final_scores = [55, 62, 100, 98.75, 80, 76.5, 85.25]
 5 #Combine the scores into a single list
 6 all_scores = midterm_scores + final_scores
 8 num midterm scores = len(midterm_scores)
 9 num final scores = len(final scores)
10
11 print(num_midterm_scores, 'students took the midterm.')
12 print(num_final_scores, 'students took the final.')
14 #Calculate the number of students that took the midterm but not the fi
15 dropped_students = num_midterm_scores - num_final_scores
16 print(dropped students, 'students must have dropped the class.')
18 lowest final = min(final scores)
19 highest final = max(final scores)
                                       lowest final 'to' highest final)
21 nrint('\nFinal scores ranged from'
Run
```

PARTICIPATION ACTIVITY	3.2.6: Using sequence-type function	ons.
concatena	xpression that tes the list to the end of . Show answer	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
	xpression that finds um value in the list	
the averag	Show answer atement that assigns be of the elements of the variable avg_price.	
Check CHALLENGE ACTIVITY	Show answer 3.2.2: List functions and methods.	
		©zvBooks 03/05/20 10:22 591/119

Tuples

©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020

A **tuple**, usually pronounced "tuhple" or "toople", behaves similar to a list but is immutable – once created the tuple's elements cannot be changed. A tuple is also a sequence type, supporting len(), indexing, and other sequence type functions. A new tuple is generated by creating a list of comma-separated values, such as **5**, **15**, **20**.

Typically, tuples are surrounded with parentheses, as in (5, 15, 20). Note that printing a tuple always displays surrounding parentheses.

A tuple is not as common as a list in practical usage, but can be useful when a programmer wants to ensure that values do not change. Tuples are typically used when element position, and not just the relative ordering of elements, is important. Ex: A tuple might store the latitude and longitude of a landmark because a programmer knows that the first element should be the latitude, the second element should be the longitude, and the landmark will never move from those coordinates. Nawab Winter 2020

Figure 3.2.4: Using tuples. white house coordinates = (38.8977, 77.0366) Coordinates: (38.8977, 77.0366) print('Coordinates:', Tuple length: 2 white_house_coordinates) print('Tuple length:', Latitude: 38.8977 north len(white house coordinates)) Longitude: 77.0366 west # Access tuples via index Traceback (most recent call print('\nLatitude:', last): white house coordinates[0], 'north') File "<stdin>", line 10, in print('Longitude:', <module> white house coordinates[1], 'west\n') TypeError: 'tuple' object does not support item assignment # Error. Tuples are immutable white_house_coordinates[1] = 50

PARTICIPATION 3.2.7: Tuples.	
 Create a new variable point that is a tuple containing the strings 'X string' and 'Y string'. 	
Check Show answer	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
<pre>2) If the value of variable friends is the tuple ('Cleopatra', 'Marc', 'Seneca'), then</pre>	

what is the result of len(friends)? Check **Show answer**

CHALLENGE ACTIVITY

3.2.3: Initialize a tuple.

UCSCCSE20NawabWinter20

Initialize the tuple team_names with the strings 'Rockets', 'Raptors', 'Warriors', and 'Celtics' (The top-4 2018 NBA teams at the end of the regular season in order). Sample output for the given program:

Rockets Raptors Warriors Celtics

```
1 team_names = ''' Your solution goes here '''
3 print(team_names[0])
4 print(team_names[1])
5 print(team_names[2])
6 print(team_names[3])
```

©zyBooks 03/05/20 10:22 591419 UCSCCSE20NawabWinter2020

Run



↓ Download student submissions

Named tuples

A program commonly captures collections of data; for example, a car could be be described using a series of variables describing the make, model, retail price, however, and number of seats. A **named tuple** allows the programmer to define a new simple data type that consists of named attributes. A **Car** named tuple with fields like **Car.price** and **Car.horsepower** would more clearly represent a car object than a list with index positions correlating to some attributes.

The *namedtuple* package must be imported to create a new named tuple. Once the package is imported, the named tuple should be created like in the example below, where the name and attribute names of the named tuple are provided as arguments to the namedtuple constructor. Note that the fields to include in the named tuple are found in a list, but may also be a single string with space or comma separated values.

```
Figure 3.2.5: Creating named tuples.

from collections import namedtuple
Car = namedtuple('Car', ['make','model','price','horsepower','seats']) #
Create the named tuple
chevy_blazer = Car('Chevrolet', 'Blazer', 32000, 275, 8) # Use the named
tuple to describe a car
chevy_impala = Car('Chevrolet', 'Impala', 37495, 305, 5) # Use the named
tuple to describe a different car

print(chevy_blazer)
print(chevy_impala)

Car(make='Chevrolet', model='Blazer', price=32000, horsepower=275, seats=8)
Car(make='Chevrolet', model='Impala', price=37495, horsepower=305, seats=5)
:22 591419

Alexey Munishkin
UCSCCSE20NawabWinter2020
```

namedtuple() only creates the new simple data type, and does not create new data
objects. Above, a new data object is not created until Car() is called with appropriate
values. A data object's attributes can be accessed using dot notation, as in
chevy_blazer.price. This "named" attribute is simpler to read than if using a list or
tuple referenced via index like chevy_blazer[2].

Like normal tuples, named tuples are immutable. A programmer wishing to edit a named tuple would replace the named tuple with a new object.

PARTICIPATION ACTIVITY	3.2.8: Named tuples.	
	edtuple has been imported. Use a lis) constructor where applicable.	st of strings in the 105/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
tuple defini house. House =		
2) Create a ne breed, and Check	ew named tuple Dog that has the att color. Show answer	ributes name ,
'country'	ss = namedtuple('Address', [']). Create a new address object horeet is "221B Baker Street", house.	buse where
Check	Show answer	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin
<pre>tuple Car namedtup] ['make', 'price',</pre>	le('Car',	UCSCCSE20NawabWinter2020

car1 and car2, write an expression that computes the sum of the price of both cars. Check **Show answer** CHALLENGE 3.2.4: Creating a named tuple **ACTIVITY** Define a named tuple **Player** that describes an athlete on a sports team. Include the fields name, number, position, and team. 1 from collections import namedtuple 3 Player = ''' Your solution goes here ''' 5 cam = Player('Cam Newton', '1', 'Quarterback', 'Carolina Panthers') 6 lebron = Player('Lebron James', '23', 'Small forward', 'Los Angeles Lakers') 8 print(cam.name + '(#' + cam.number + ')' + ' is a ' + cam.position + ' for the ' + 9 print(lebron.name + '(#' + lebron.number + ')' + ' is a ' + lebron.position + ' for Run **↓** Download student submissions



3.3 Set basics

Set basics

©zvBooks 03/05/20 10:22 591419

A **set** is an unordered collection of unique elements. Sets have the following hkin properties:

- Elements are unordered: Elements in the set do not have a position or index.
- Elements are unique: No elements in the set share the same value.

A set can be created using the **set()** function, which accepts a sequence-type iterable object (list, tuple, string, etc.) whose elements are inserted into the set. A **set literal** can be written using curly braces { } with commas separating set elements. Note that an empty set can only be created using **set()**.

```
# Create a set using the set() function.
nums1 = set([1, 2, 3])
# Create a set using a set literal.
nums2 = { 7, 8, 9 }
# Print the contents of the sets.
print(nums1)
print(nums2)
[1, 2, 3]
{7, 8, 9}
```

Because the elements of a set are unordered and have no meaningful position in the collection, the index operator is not valid. Attempting to access the element of a set by position, for example nums1[2] to access the element at index 2 is invalid and will 1419 produce a runtime error.

Alexey Munishkin

A set is often used to reduce a list of items that potentially contains duplicates into a collection of unique values. Simply passing a list into **set()** will cause any duplicates to be omitted in the created set.

```
Run
                                      Load default template...
               1 # Initial list contains some duplicate values
               2 first_names = [ 'Harry', 'Hermione', 'Ron', 'H
               4 # Creating a set removes any duplicate values
               5 names_set = set(first_names)
                 print(names set)
                                                            >
PARTICIPATION
                3.3.1: Basic sets.
ACTIVITY
1) What's the result of set(['A',
   'Z'])?
     O A set that contains 'A'
         and 'Z'
      A list with the following
         elements: ['A', 'Z'].
     O Error: invalid syntax.
2) What's the result of set(10,
   20, 25)?

    A list with the following

         elements: [10, 20,
         25].
      • A set that contains 10,
         20, and 25.
     O Error: invalid syntax.
```

3)

What's the result of set([100, 200, 100, 200, 300])?

O A list with the following elements: [100, 200, 100, 200, 300].

O A set that contains 100, 200, and 300.

O A set that contains 100, 200, 300, another 100, and another 200.

O Error: invalid syntax.

Modifying sets

Sets are mutable – elements can be added or removed using set methods. The **add()** method places a new element into the set if the set does not contain an element with the provided value. The **remove()** and **pop()** methods remove an element from the set.

Additionally, sets support the len() function to return the number of elements in a set. To check if a specific value exists in a set, a membership test such as value in set (discussed in another section) can be used.

Adding elements to a set:

set.add(value): Add value into the set. Ex: my_set.add('abc')

Remove elements from a set:

- set.remove(value): Remove the element with given value from the set. Raises KeyError if value is not found. Ex: my_set.remove('abc')
- my_set.pop(): Remove a random element from the set. Ex: my_set.pop()

Table 3.3.1: Some of the methods useful to sets. Alexey Munishkin UCSCCSE20NawabWinter2020

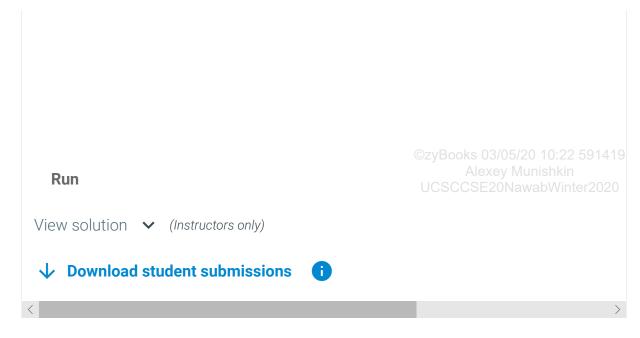
Operation	Description
len(set)	Find the length (number of elements) of the set.
set1.update(set2)	Adds the elements in set2 to set1.

set.add(value)	Adds value into the set.
set.remove(value)	Removes value from the set. Raises KeyError if value is not found.
set.pop()	Removes a random element from the set.
set.clear()	Clears all elements from the set. ©zyBooks 03/05/20 10:22 5914 9
	UCSCCSE20NawabWinter2020

PARTICIPATION ACTIVITY	3.3.2: Modifying sets.		
Animation c	ontent:		
Animation c	aptions:		
2. The add(3. The upda4. The remainstance) method adds a single ele ate() method adds the eler ove() method removes a s r() method removes all ele	ments of one set to another set.	
PARTICIPATION ACTIVITY	3.3.3: Modifying sets.		
Write a line of	code to complete the follo	wing operations.	
1) Add the lite set names.	ral ' Ryder' to the		
Check	Show answer	©zyBooks 03/05/20 10:2 Alexey Munishk UCSCCSE20NawabWi	
2) Add all of the into set mor	ne elements of set gobli n s ters .	S	

<

©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
e Oliver, Declan, and Henry
set by removing a name and
which the names in



Set operations

Python set objects support typical set theory operations like intersections and unions. A brief overview of common set operations supported in Python are provided below:

Table 3.3.2: Common set theory operations.

Operation	Description
set.intersection(set_a, set_b, set_c)	Returns a new set containing only the elements in common between set and all provided sets.
set.union(set_a, set_b, set_c)	Returns a new set containing all of the unique elements in all sets.
set.difference(set_a, set_b, set_c)	Returns a set containing only the 10:22 5914 elements of set that are not found in any of the provided sets.
set_a.symmetric_difference(set_b)	Returns a set containing only elements that appear in exactly one of set_a or set_b

PARTICIPATION ACTIVITY	3.3.4: Set theory operation	ons.	
Animation (content:		
Animation (•	Alexey	5/20 10:22 591419 Munishkin lawabWinter2020 s from names1
and nan 2. The inte	nes2. 'Corrin' only appears		
	erence() method builds a s et that are not in names4.	et that contains elements on	ly found in
PARTICIPATION ACTIVITY	3.3.5: Set theory operation	ons.	
Assume that:			
• trolls	rs = {'Gorgon', 'Medu = {'William', 'Bert' = {'Gorgon', 'Bert',	, 'Tom'}	
Fill in the code	e to complete the line that	would produce the given set.	
1) {'Gorgon 'Medusa'	', 'Bert', 'Tom', , 'William'}		
monsters (trolls)			
Check	Show answer		
			/20 40.22 504440

2) {'Gorgon'}

monsters. (horde)

Check

Show answer

3) {'Medusa', 'Bert', 'Tom'}

```
monsters.symmetric_difference(

Check Show answer
```

CHALLENGE ACTIVITY

3.3.2: Set theory methods.



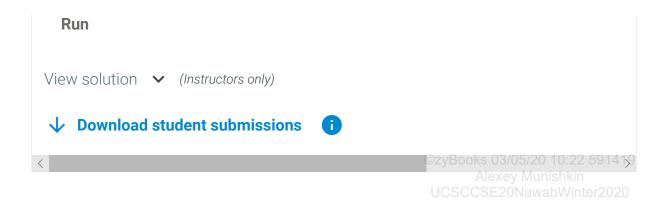
The following program includes fictional sets of the top 10 male and female baby names for the current year. Write a program that creates:

- 1. A set all_names that contains all of the top 10 male and all of the top 10 female names.
- 2. A set neutral_names that contains only names found in both male_names and female_names.
- 3. A set **specific_names** that contains only gender specific names.

Sample output for all names:

```
{'Michael', 'Henry', 'Jayden', 'Bailey', 'Lucas', 'Chuck',
'Aiden', 'Khloe', 'Elizabeth', 'Maria', 'Veronica',
'Meghan', 'John', 'Samuel', 'Britney', 'Charlie', 'Kim'}
```

NOTE: Because sets are unordered, they are printed using the sorted() function here for comparison.



3.4 Dictionary basics

Creating a dictionary

Consider a normal English language dictionary – a reader looks up the word "cat" and finds the definition, "A small, domesticated carnivore." The relationship between "cat" and its definition is *associative*, i.e., "cat" is associated with some words describing "cat."

A *dictionary* is a Python container used to describe associative relationships. A dictionary is represented by the *dict* object type. A dictionary associates (or "maps") keys with values. A *key* is a term that can be located in a dictionary, such as the word "cat" in the English dictionary. A *value* describes some data associated with a key, such as a definition. A key can be any immutable type, such as a number, string, or tuple; a value can be any type.

A dict object is created using **curly braces** { } to surround the **key:value pairs** that comprise the dictionary contents. Ex:

players = {'Lionel Messi': 10, 'Cristiano Ronaldo': 7} creates a dictionary called players with two keys: 'Lionel Messi' and 'Cristiano Ronaldo', associated with the values 10 and 7 (their respective jersey numbers). An empty dictionary is created with the expression players = { } PzyBooks 03/05/20 10:22 591419 Alexey Munishkin

Dictionaries are typically used in place of lists when an associative relationship exists. Ex: If a program contains a collection of anonymous student test scores, those scores should be stored in a list. However, if each score is associated with a student name, a dictionary could be used to associate student names to their score. Other examples of associative relationships include last names and addresses, car models and price, or student ID number and university email address.

Figure 3.4.1: Creating a dictionary.

```
players = {
    'Lionel Messi': 10,
    'Cristiano Ronaldo': 7
}

print(players)

{'Lionel Messi': 10, 'Cristiano Ronaldo'; 7}

Alexey Munishkin
    UCSCCSE20NawabWinter2020
```

Note that formatting list or dictionary entries like in the above example, where elements appear on consecutive lines, helps to improve the readability of the code. The behavior of the code is not changed.

zyDE 3.4.1: Creating dictionaries.

Run the program below that displays the caffeine content in milligrams f some popular foods. The indentation and spacing of the caffeine_conter simply provides more readability. Note that order is *not* maintained in the

Try adding new items into the dictionary, using this U.S. federal governm content.

Run

```
Load default template...
1 caffeine_content_mg = {
       'Mr. Goodbar chocolate': 122,
2
 3
       'Red Bull': 33,
       'Monster Hitman Sniper energy drink': 270
 4
 5
       'Lipton Brisk iced tea - lemon flavor': 2
6
       'dark chocolate coated coffee beans': 869
7
       'Regular drip or percolated coffee': 60,
8
       'Buzz Bites Chocolate Chews': 1639
9 }
10
11 print(caffeine_content_mg)
12
```

<

PARTICIPATION ACTIVITY	3.4.1: Create a dictionary.	
maps the 'Frank' to t respective	s to create a called ages that names 'Bob' and heir ages, 27 and 75, ly. For this exercise, ' the first entry in the	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
ages = Check	Show answer	

Accessing dictionary entries

Unlike a list, which orders elements according to a left-to-right positioning in a sequence, a dictionary's entries maintain no such ordering. To access an entry, the key is specified in brackets []. If no entry with a matching key exists in the dictionary, then a **KeyError** runtime error occurs and the program is terminated.

```
Figure 3.4.2: Accessing dictionary entries.
  prices = {'apples': 1.99, 'oranges': 1.49}
                                                   The price of apples is 1.99
                                                   Traceback (most recent call
  print('The price of apples is',
  prices['apples'])
                                                     File "<stdin>", line 3, in
  print('\nThe price of lemons is',
                                                   <module>
  prices['lemons'])
                                                   KeyError: 'lemons'
                                                        ©zyBooks 03/05/20 10:22 591419
PARTICIPATION
                3.4.2: Accessing dictionary entries.
ACTIVITY
1) Dictionary entries are ordered
   by position.
```

True	
O False	
2) A dictionary entry is accessedby placing a key in curly braces{}.	
O True	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin
O False	UCSCCSE20NawabWinter2020

Adding, modifying, and removing dictionary entries

A dictionary is mutable, so entries can be added, modified, and deleted as necessary by a programmer. A new dictionary entry is added by using brackets to specify the key: prices['banana'] = 1.49. A dictionary key is unique — attempting to create a new entry with a key that already exists in the dictionary replaces the existing entry. The del keyword is used to remove entries from a dictionary: del prices['papaya'] removes the entry whose key is 'papaya'. If the requested key to delete does not exist then a KeyError occurs.

Adding new entries to a dictionary:

dict[k] = v: Adds the new key-value pair k-v, if dict[k] does not already exist.
 Example: students['John'] = 'A+'

Modifying existing entries in a dictionary:

dict[k] = v: Updates the existing entry dict[k], if dict[k] already exists.
 Example: students['Jessica'] = 'A+'

Removing entries from a dictionary:

del dict[k]: Deletes the entry dict[k].
 Example: del students['Rachel']

```
Figure 3.4.3: Adding and editing dictionary UCSCCSE20NawabWinter2020 entries.

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

```
prices['banana'] = 1.49 # Add new entry
    print(prices)
    prices['banana'] = 1.69 # Modify entry
    print(prices)
    del prices['banana'] # Remove entry
    print(prices)
                                                        ©zyBooks 03/05/20 10:22 591419
                                                                Alexey Munishkin
PARTICIPATION
                3.4.3: Modifying dictionaries.
ACTIVITY
1) Which statement adds 'pears'
   to the following dictionary?
    prices = {'apples': 1.99,
    'oranges': 1.49, 'kiwi': 0.79}
     O prices['pears'] = 1.79
     O prices['pears': 1.79]
2) Executing the following
   statements produces a
   KeyError:
    prices = {'apples': 1.99,
    'oranges': 1.49, 'kiwi': 0.79}
    del prices['limes']
      O True
      O False
3) Executing the following
   statements adds a new entry to
   the dictionary:
    prices = {'apples': 1.99,
    'oranges': 1.49, 'kiwi': 0.79}
    prices['oranges'] = 1.29
      O True
      O False
CHALLENGE
             3.4.1: Modify and add to dictionary.
ACTIVITY
```

prices = {} # Create empty dictionary

Write a statement to add the key Tesla with value USA to car_makers. Modify the car maker of Fiat to Italy. Sample output for the given program: Acura made in Japan Fiat made in Italy Tesla made in USA 1 car_makers = {'Acura': 'Japan', 'Fiat': 'Egypt'} 3 # Add the key Tesla with value USA to car_makers 4 # Modify the car maker of Fiat to Italy 6 ''' Your solution goes here ''' 8 print('Acura made in', car_makers['Acura']) 9 print('Fiat made in', car_makers['Fiat'])
10 print('Tesla made in', car_makers['Tesla']) Run **↓** Download student submissions

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
UCSCCSE20NawabWinter2020

3.5 Common data types summary

The most common Python types are presented below.

Common data types

Numeric types int and float represent the most common types used to store data. All numeric types support the normal mathematical operations such as addition, subtraction, multiplication, and division, among others.

Table 3.5.1: Common data types.

©zyBooks 03/05/20 10:22 591419

Туре	Notes UCSC	CSE20NawabWinter2020
int	Numeric type: Used for variable-width integers.	
float	Numeric type: Used for floating-point numbers.	

Sequence types string, list, and tuple are all containers for collections of objects ordered by position in the sequence, where the first object has an index of 0 and subsequent elements have indices 1, 2, etc. A list and a tuple are very similar, except that a list is mutable and individual elements may be edited or removed. Conversely, a tuple is immutable and individual elements may not be edited or removed. Lists and tuples can contain any type, whereas a string contains only single-characters. Sequence-type functions such as len() and element indexing using brackets [] can be applied to any sequence type.

The only **mapping type** in Python is the dict type. Like a sequence type, a dict serves as a container. However, each element of a dict is independent, having no special ordering or relation to other elements. A dictionary uses key-value pairs to associate a key with a value.

Table 3.5.2: Containers: sequence and mapping types.

Туре	Notes		
string	Sequence type: Used for text.	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020	
list	Sequence type: A mutable container with ordered elements.		
tuple	Sequence type: An immutable container with ordered elements.		
dict	Mapping type: A container with key-values associated elements.		

_		
PARTICIPATION ACTIVITY	3.5.1: Common data types.	
invalid bed	a', 'b', 3] is cause the list contains crings and integers.	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
O Fals	e	
	at types can always xact same values.	
O True		
O Fals	е	
	ollection of integers t be contained in a	
O True		
O Fals	е	

Choosing a container type

New programmers often struggle with choosing the types that best fit their needs, such as choosing whether to store particular data using a list, tuple, or dict. In general, a programmer might use a list when data has an order, such as lines of text on a page. A programmer might use a tuple instead of a list if the contained data should not change. If order is not important, a programmer might use a dictionary to capture relationships between elements, such as student names and grades.

PARTICIPATION ACTIVITY	3.5.2: Choosing among different container types.03/05/20 10:22 59141 Alexey Munishkin
	UCSCCSE20NawabWinter2020
Choose the co	ontainer that best fits the described data.
	st scores that may ljusted, ordered from orst.
O list	

O tuple	
O dict	
2) Student names and their current grades	
O list	
O tuple	©zyBooks 03/05/20 10:22 59141 Alexey Munishkin
O dict	UCSCCSE20NawabWinter2020
3) The final number of As, E Ds, and Fs in the class	Bs, Cs,
O list	
O tuple	
O dict	
PARTICIPATION ACTIVITY 3.5.3: Finding Click on the error.	g errors in container code.
ACTIVITY 3.5.3: FINGIN	
Click on the error. 1) # Student grade program. students = [']o', grades = {} # Get student name, gradename = input('name:') grade = input('grade:')#	'Bob',
Click on the error. 1) # Student grade program. students = ['Jo',	Bob', le Assign ©zyBooks 03/05/20 10:22 59141 Alexey Munishkin LICSCCSE20NawabWinter2020

```
# Remove Amy from workers

del workers[1]

# Print workers

print('Jo:', workers[0])

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
UCSCCSE20NawabWinter2020
```

3.6 Additional practice: Grade calculation

The following program calculates an overall grade in a course based on three equally weighted exams.

```
zyDE 3.6.1: Grade calculator: Average score on three exams.
```

```
Load
```

```
1 exam1_grade = float(input('Enter score on Exam 1 (out of 100):\n'))
2 exam2_grade = float(input('Enter score on Exam 2 (out of 100):\n'))
3 exam3_grade = float(input('Enter score on Exam 3 (out of 100):\n'))
4
5 overall_grade = (exam1_grade + exam2_grade + exam3_grade) / 3
6
7 print('Your overall grade is:', overall_grade)
8
```

©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020

```
70
75
91
```

	Run	
	<	
<		©zyBooks 03/05/20 10:22 59141\$

Create a different version of the program that:

Alexey Munishkin UCSCCSE20NawabWinter2020

- 1. Calculates the overall grade for four equally weighted programming assignments, where each assignment is graded out of 50 points. Hint: First calculate the percentage for each assignment (e.g., score / 50), then calculate the overall grade percentage (be sure to multiply the result by 100).
- 2. Calculates the overall grade for four equally weighted programming assignments, where assignments 1 and 2 are graded out of 50 points and assignments 3 and 4 are graded out of 75 points.
- 3. Calculates the overall grade for a course with three equally weighted exams (graded out of 100) that account for 60% of the overall grade and four equally weighted programming assignments (graded out of 50) that account for 40% of the overall graded. Hint: The overall grade can be calculated as 0.6 * averageExamScore + 0.4 * averageProgScore.
- 4. Extend the program to support the grading scheme for one (or all) of the courses.

3.7 Type conversions

Type conversions

©zvBooks 03/05/20 10:22 591419

A calculation sometimes must mix integer and floating-point numbers. For example, given that about 50.4% of human births are males, then 0.504 * num_births ter calculates the number of expected males in num_births births. If num_births is an integer type, then the expression combines a floating-point and integer.

A **type conversion** is a conversion of one type to another, such as an int to a float. An **implicit conversion** is a type conversion automatically made by the interpreter, usually

between numeric types. For example, the result of an arithmetic operation like + or * will be a float only if either operand of the operation is a float.

- 1 + 2 returns an integer type.
- 1 + 2.0 returns a float type.
- 1.0 + 2.0 returns a float type.

int-to-float conversion is straightforward: 25 becomes 25.0. Alexey Munishkin UCSCCSE20NawabWinter2020 float-to-int conversion just drops the fraction: 4.9 becomes 4.

PARTICIPATION ACTIVITY	3.7.1: Implicit co	onversions between float and int.
Type the value	held in the variat	ble after the assignment statement, given:
num_iteritem_we		
For any floatin	g-point answer, ty	ype answer to tenths. Ex: 8.0, 6.5, or 0.1
1) num_items	+ num_items.	
Check	Show answer	
2) item_weigh	nt * num_items.	
Check	Show answer	
3) (num_item: item_weigh	s + num_items) * it	
Check	Show answer	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020

Conversion methods

Sometimes a programmer needs to explicitly convert an item's type. Conversion can be explicitly performed using the below conversion methods:

Table 3.7.1: Conversion methods for some common types.

Function	Notes	©zyBooks 03/05/20 10:22 5 Can convert _{Alexey} Munishkin	
int()	Creates integers	int, float, strings w/ integers only	er2020
float()	Creates floats	int, float, strings w/ integers or fractions	
str()	Creates strings	Any	

Converting a float to an int will truncate the floating-point number's fraction. For example, the variable temperature might have a value of 18.75232, but can be converted to an integer expression **int(temperature)**. The result would have the value 18, with the fractional part removed.

Conversion of types is very common. In fact, all user input obtained using input() is initially a string and a programmer must explicitly convert the input to a numeric type.

Strings can also be converted to numeric types, if the strings follow the correct formatting, i.e. using only numbers and possibly a decimal point. For example, int('500') yields an integer with a value of 500, and float('1.75') yields the floating-point value 1.75.

zyDE 3.7.1: Simple example of converting float and int types.

Run the below program. Observe how the type conversion affects the en the input to 18.552 and run the program again.

Load default template...

Load default template...

Run

	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
<	
<	>
PARTICIPATION ACTIVITY 3.7.2: Type conversions.	
What is the result of each expression?	
1) int(1.55)	
O 1.55	
0 1	
O '1.55' 1 input_text = input('Enter of the control	a number:\n') ut text)
3 int_variable = int(float_v 2) float("7 99")4	ariable)
7 0 6 print('input text converte	d to a float:', floa
7 print('float variable conv	erted to an int:', i
O 7.99	
3) str(99)	©zyBooks 03/05/20 10:22 591419
O 99	Alexey Munishkin UCSCCSE20NawabWinter2020
O 99.0	
O '99'	
•	
CHALLENGE 3.7.1: Type casting: Computir	ig average owls per zoo.

Assign avg_owls with the average owls per zoo. Print avg_owls as an integer. Sample output for inputs: 1 2 4 Average owls per zoo: 2 1 avg_owls = 0.03 num owls zooA = int(input()) 4 num_owls_zooB = int(input()) 5 num_owls_zooC = int(input()) 7 ''' Your solution goes here ''' 9 print(()'Average owls per zoo:', int(avg_owls)) Run **↓** Download student submissions CHALLENGE 3.7.2: Type casting: Reading and adding values. ACTIVITY Assign total_owls with the sum of num_owls_A and num_owls_B=20NawabWinter2020 Sample output with inputs: 3 4 Number of owls: 7

3.8 Binary numbers

Binary numbers

©zyBooks 03/05/20 10:22 591419

Normally, a programmer can think in terms of base ten numbers. However, a computer must allocate some finite quantity of bits (e.g., 32 bits) for a variable, and that quantity of bits limits the range of numbers that the variable can represent. Python allocates additional memory to accommodate numbers of very large sizes (past a typical 32 or 64 bit size), and a Python programmer need not think of such low level details. However, binary base computation is a common and important part of computer

science, so some background on how the quantity of bits influences a variable's number range is helpful.

Because each memory location is composed of bits (0s and 1s), a processor stores a number using base 2, known as a **binary number**.

For a number in the more familiar base 10, known as a **decimal number**, each digit must be 0-9 and each digit's place is weighed by increasing powers of 100 10:22 591419

UCSCCSE20NawabWinter2020

Table 3.8.1: Decimal numbers use weighed powers of 10.

Decimal number with 3 digits	R	epresentation	٦
212	$= 2 \cdot 10^{2}$ $= 2 \cdot 100$ $= 200$ $= 212$	$+ 1 \cdot 10^{1}$ $+ 1 \cdot 10$ $+ 10$	$+ 2 \cdot 10^{0}$ $+ 2 \cdot 1$ + 2

In **base 2**, each digit must be 0-1 and each digit's place is weighed by increasing powers of 2.

Table 3.8.2: Binary numbers use weighed powers of 2.

Binary number with 4 bits		Represe	entation
1101	$= 1 \cdot 2^{3}$ $= 1 \cdot 8$ $= 8$ $= 13$	$+1 \cdot 2^{2}$ $+1 \cdot 4$ $+4$	ՖՁշնեs 03/ԽՁ/2(2 ¹ 0:22 591։ + 0 ⋅ 2 Alexe¥ իկսղishkin UCSCCSE20NawabWinter20 + 0 + 1

PARTICIPATION

15 (he largest in	teger that the 8 bits can repre	esent.
			©zyBooks 03/05/20 10:22 591 Alexey Munishkin
	RTICIPATION	3.8.2: Binary numbers.	UCSCCSE20NawabWinter 20
		binary number o a decimal number.	
	Check	Show answer	
		binary number o a decimal number.	
	Check	Show answer	
,		decimal number 17 inary number.	
	Check	Show answer	
/		decimal number 51 inary number.	
	Check	Show answer	©zyBooks 03/05/20 10:22 591 Alexey Munishkin UCSCCSE20NawabWinter20

3.9 String formatting

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
UCSCCSF20NawabWinter2020

The format() function

Program output commonly includes variables as a part of the text. The string **format()** function allows a programmer to create a string with placeholders that are replaced by values or variable values at execution. A placeholder surrounded by curly braces { } is called a **replacement field**. Values inside the **format()** parentheses are inserted into the replacement fields in the string.

PARTICIPATION ACTIVITY	3.9.1: String formatting.
Animation of	content:
undefined	
Animation of	captions:
format()	replacement field {} in the string is replaced with the first value in the parentheses. t replacement field uses the next value, and so on.

The three ways to provide values to replacements fields include:

Table 3.9.1: Th		s 03/05/20 10:22 591419 Jexey Munishkin
Replacement definition	Example	Formatted string result
Positional replacement	'The {1} in the {0} is {2}.'.format('hat', 'cat', 'fat')	The cat in the hat is fat.

p	nferred positional replacement	'The {} in the {} is {}.'.format('cat', 'hat', 'fat')	The cat in the hat is fat.
	Named replacement	<pre>'The {animal} in the {headwear} is {shape}.'.format(animal='cat', headwear='hat', shape='fat')</pre>	The cat in the hat is fat.
			lexey Munishkin SE20NawabWinter2020

Named replacement allows a programmer to create a **keyword argument** that defines a name and value in the **format()** parentheses. The name can then be placed into a replacement field. Ex: **animal='cat'** is a keyword argument that can be used in a replacement field like **{animal}** to insert the word "cat". <u>Good practice</u> is to use named replacement when formatting strings with many replacement fields to make the code more readable.

Note: The positional and inferred positional replacement types cannot be combined. Ex: '{} + {1} is {2}'.format(2, 2, 4) is not allowed. However, named and either positional replacement type can be combined. Ex:

Double braces {{ }} can be used to place an actual curly brace into a string. Ex: '{0} {{Bezos}}'.format('Amazon') produces the string "Amazon {Bezos}".

ACTIVITY 5.9.2

PARTICIPATION

3.9.2: Positional and named replacement in format strings.

Animation content:

Animation captions:

- 1. Empty replacement fields infer their position based on the order of values in format().
- 2. Numbers in replacement fields indicate the position of the value in format().
- 3. Names in replacement fields indicate a named keyword from format() in

UCSCCSE20NawabWinter2020

PARTICIPATION ACTIVITY	3.9.3: string.format() usage.	
Determine the	output of the following code snippets.	1

1)	<pre>print('April {}, {}'.format(22, 2020))</pre>	
	Check Show answer	
2)	<pre>date = 'April {}, {}' print(date.format(22, 2020))</pre>	©zyBooks 03/05/20 10:22 5914/19 Alexey Munishkin UCSCCSE20NawabWinter2020
	Check Show answer	
3)	<pre>date = 'April {}, {}' print(date.format(22, 2020)) print(date.format(23, 2024))</pre>	
	Check Show answer	
4)	<pre>print('{0}:{1}'.format(9, 43))</pre>	
	Check Show answer	
5)	<pre>print('{0}:{0}'.format(9, 43))</pre>	
	Check Show answer	
6)	<pre>print('Hi {{{0}}}!'.format('Bilbo'))</pre>	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020
	Check Show answer	
7)		

<pre>month = 'Apr day = 22 print('Today {0}'.format(</pre>		
Check	Show answer	©zyBooks 03/05/20 10:22 591419 Alexey Munishkin
		UCSCCSF20NawahWinter2020

Format specifications

A **format specification** inside of a replacement field allows a value's formatting in the string to be customized. Ex: Using a format specification, a variable with the integer value 4 can be output as a floating-point number (4.0) or with leading zeros (004).

A common format specification is to provide a **presentation type** for the value, such as integer (4), floating point (4.0), fixed precision decimal (4.000), percentage (4%), binary (100), etc. A presentation type can be set in a replacement field by inserting a colon: and providing one of the presentation type characters described below.

Table 3.9.2: Common formatting specification presentation types.

Туре	Description	Example	Output	
S	String (default presentation type - can be omitted)	'{:s}'.format('Aiden')	Aiden	1
d	Decimal (integer values only)	'{:d}'.format(4)	4	П
b	Binary (integer values only)	'{:b}'.format(4) ©zyBoo	Alexey Munishkii	h
x, X	Hexadecimal in lowercase (x) and uppercase (X) (integer values only)	'{:x}'.format(15)	f	nter2020
е	Exponent notation		4.400000e+01	V

		'{:e}'.format(44)	
f	Fixed-point notation (6 places of precision)	'{:f}'.format(4)	4.000000
[precision]f	Fixed-point notation (programmer-defined precision)	©zyBook '{:.2f}'.format(4) UCSCC	ks 03/05/20 10:22 591419 A 4.00 Munishkin S E 200 awabWinter2020

PARTICIPATION ACTIVITY

3.9.4: Format specifications and presentation types.

Enter the most appropriate format specification to produce the desired output.

1) The value of **num** as a decimal (base 10) integer: 31

```
num = 31
print('{:
}'.format(num))
```

Check Show answer

2) The value of **num** as a hexadecimal (base 16) integer:

```
num = 31
print('{:
}'.format(num))
```

1f

Check Show answer

3) The value of **num** as a binary (base 2) integer: 11111

```
num = 31
print('{:
}'.format(num))
```

©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSF20NawabWinter2020

Referencing format() values correctly

The colon: in the replacement field separates the "what" on the left from the "how" on the right. The left "what" side references a value in the format() parentheses. The left side may be omitted (inferred positional replacement), a number (positional winter2020 replacement), or a name (named replacement). The right "how" side determines how to show the value, such as a presentation type. More advanced format specifications, like fill and alignment, are provided in a later section.

Table 3.9.3: Referencing the correct format() values in replacement fields.

Replacement type	Example	Output
Inferred positional replacement	'{:s} \${:.2f} tacos is \${:.2f} total'.format('Three', 1.50, 4.50)	Three \$1.50 tacos is \$4.50 total
Positional replacement	'{0:s} \${2:.2f} tacos is \${1:.2f} total'.format('Three', 4.50, 1.50)	Three \$1.50 tacos is \$4.50 total
Named replacement	'{cnt:s} \${cost:.2f} tacos is \${sum:.2f} total'.format(cnt = 'Three', cost = 1.50, sum = 4.50)	Three \$1.50 tacos is \$4.50 total

PARTICIPATION ACTIVITY

3.9.5: Matching code blocks to formatted strings 03/05/20 10:2

Alexey Munishkin

Match each code block to the code output. If the code would generate an error, mark as "Error".

25 + 50 = 75

50 + 25 = 75

25.0 + 50.0 €₹5.00ks 03/05/20 10:22 591419

UCSCCSE20NawabWinter2020 50 + 25 = 75.01

Error

Reset

CHALLENGE ACTIVITY

3.9.1: Printing a string.

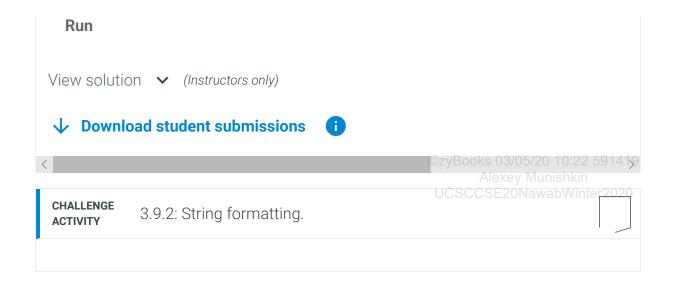
Write a *single* statement to print: user_word,user_number. Note that there is no space between the comma and user_number.

Sample output with inputs: 'Amy' 5

Amy,5

```
1 user_word = str(input())
2 user_number = int(input())
3
4 ''' Your solution goes here '''
5 |
```

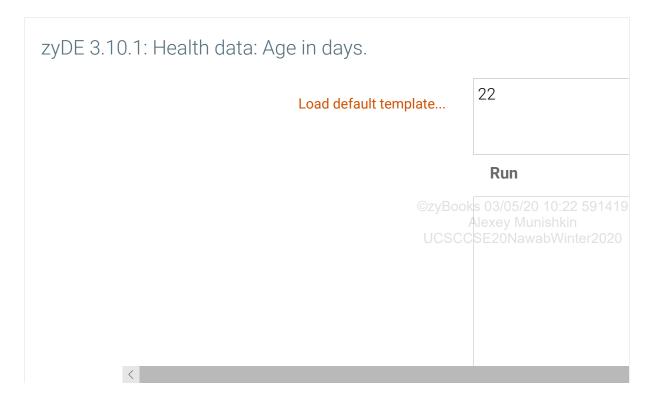
©zyBooks 03/05/20 10:22 591419 Alexey Munishkin UCSCCSE20NawabWinter2020

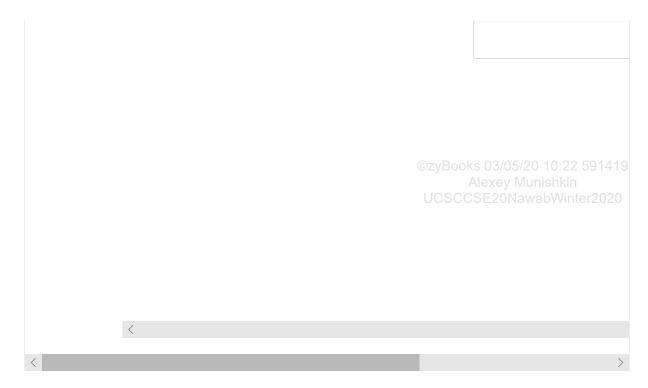


3.10 Additional practice: Health data

The following is a sample programming lab activity; not all classes using a zyBook require students to fully complete this activity. No auto-checking is performed. Users planning to fully complete this program may consider first developing their code in a separate programming environment.

The following calculates a user's age in days based on the user's age in years.





Create a different version of the program that:

- 1. Calculates the user's age in minutes and seconds.
- 2. Estimates the approximate number of times the user's heart has beat in his/her lifetime using an average heart rate of 72 beats per minute.
- 3. Estimates the number of times the person has sneezed in his/her lifetime.
- 4. Estimates the number of calories that the person has expended in his/her lifetime (research on the Internet to obtain a daily estimate). Also calculate the number of sandwiches (or other common food item) that equals that number of calories.
- 5. Be creative: Pick several other interesting health-related statistics. Try searching the Internet to determine how to calculate that data, and create a program to perform that calculation. The program can ask the user to enter any information needed to perform the calculation.

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
UCSCCSF20NawabWinter2020

3.11 LAB: Input and formatted output: Right-facing arrow

Given input characters for an arrowhead and arrow body, print a right-facing arrow.

Then the output is:

LAB ACTIVITY

3.11.1: LAB: Input and formatted output: Right-facing arrow 0 / 10

main.py

Load default template...

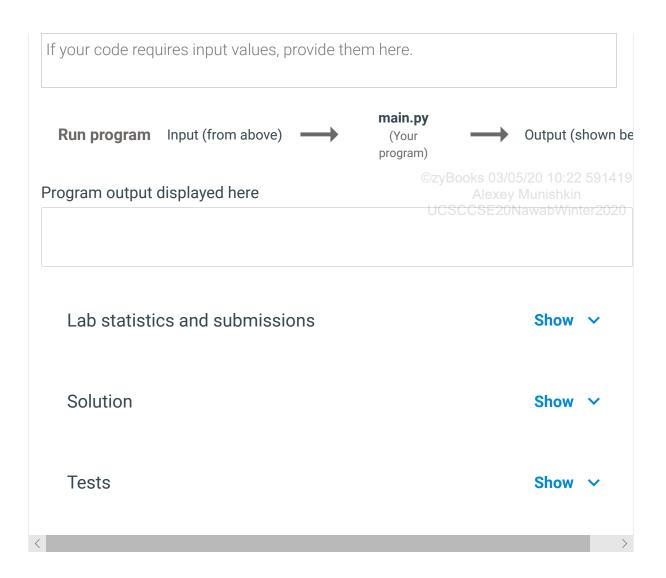
```
1 base_char = input()
2 head_char = input()
3
4 row1 = ' ' + head_char
5 ''' Type your code here. '''
6
7 print(row1)
8 print(row2)
9 print(row3)
10 print(row2)
11 print(row1)
12
13 |
```

Develop mode

Submit mode

Run your program as often as you'd like, 419 before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)



3.12 LAB: Phone number breakdown

Given an integer representing a 10-digit phone number, output the area code, prefix, and line number using the format (800) 555-1212.

Ex: If the input is:

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
UCSCCSE20NawabWinter2020

the output is:

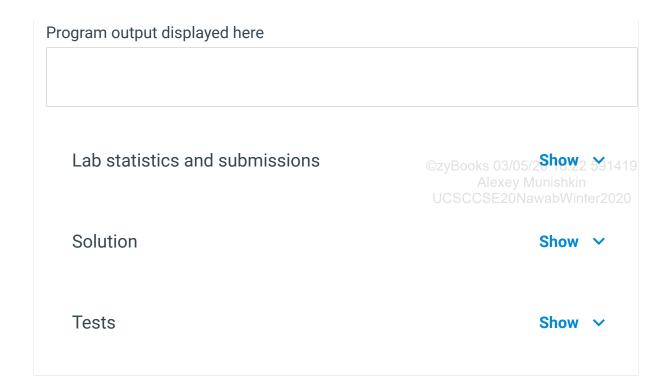
(800) 555-1212

Hint: Use % to get the desired rightmost digits. Ex: The rightmost 2 digits of 572 is gotten by 572 % 100, which is 72.

Hint: Use // to shift right by the desired amount. Ex: Shifting 572 right by 2 digits is done by 572 // 100, which yields 5. (Recall integer division discards the fraction).

For simplicity, assume any part starts with a non-zero digit. So 0119998888 is not allowed. ©zyBooks 03/05/20 10:22 591

UCSCCSE20NawabWinter LAB 3.12.1: LAB: Phone number breakdown 0/10 **ACTIVITY** main.py Load default template... 1 phone_number = int(input()) ''' Type your code here. ''' 3 5 Run your program as often as you'd like, **Develop mode Submit mode** before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box. Enter program input (optional) If your code requires input values, provide them here. UCSCCSE20NawabWinter2020 main.py **Run program** Input (from above) Output (shown be (Your program)



3.13 LAB: Input and formatted output: Caffeine levels

A half-life is the amount of time it takes for a substance or entity to fall to half its original value. Caffeine has a half-life of about 6 hours in humans. Given caffeine amount (in mg) as input, output the caffeine level after 6, 12, and 24 hours. Use a string formatting expression with conversion specifiers to output the caffeine amount as floating-point numbers.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:

```
print('{:.2f}'.format(your_value))

Ex: If the input is:

©zyBooks 03/05/20 10:22 591419

Alexey Munishkin

UCSCCSE20NawabWinter2020
```

100

the output is:

```
After 6 hours: 50.00 mg
After 12 hours: 25.00 mg
After 24 hours: 6.25 mg
```

Note: A cup of coffee has about 100 mg. A soda has about 40 mg. An "energy" drink (a misnomer) has between 100 mg and 200 mg.

©zyBooks 03/05/20 10:22 591419

Alexey Munishkin
3.13.1: LAB: Input and formatted output: Caffeine levels Naw 0 / 10e 2020

main.py

Load default template...

1 caffeine_mg = float(input())
2
3 ''' Type your code here. '''
4
5 |

Develop mode

Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

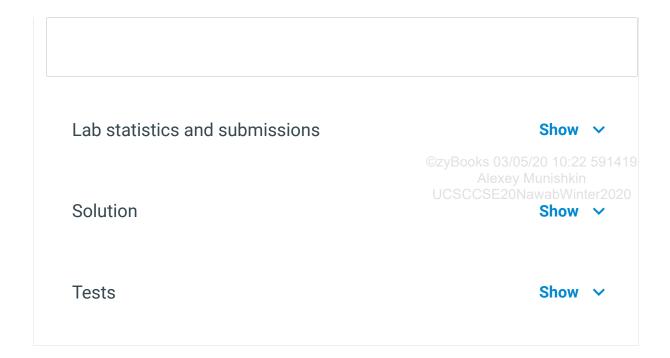
Enter program input (optional)

If your code requires input values, provide them here.

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
UCSCCSE20NawabWinter2020

Run program Input (from above) (Your program) Output (shown be

Program output displayed here



3.14 LAB: Input and formatted output: House real estate summary

Sites like Zillow get input about house prices from a database and provide nice summaries for readers. Write a program with two inputs, current price and last month's price (both integers). Then, output a summary listing the price, the change since last month, and the estimated monthly mortgage computed as (current_price * 0.051) / 12.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:

```
print('{:.2f}'.format(your_value))
```

Ex: If the input is:

	©zyBooks 03/05/20 10:22 591419
200000	Alexey Munishkin UCSCCSE20NawabWinter2020
210000	OCSCCSEZUNAWADWIIItei 2020

the output is:

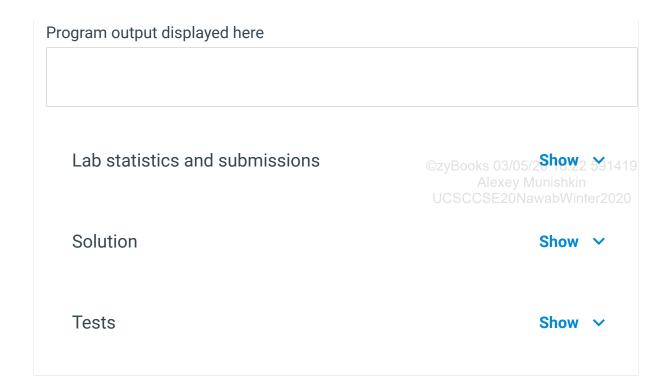
This house is \$200000. The change is \$-10000 since last month.

The estimated monthly mortgage is \$850.00.

Note: Getting the precise spacing, punctuation, and newlines exactly right is a key point

of this assignment. Such precision is an important part of programming. 3.14.1: LAB: Input and formatted output: House real estate LAB **ACTIVITY** summary 10 main.py Load default template... 1 current_price = int(input()) 2 last_months_price = int(input()) 4 ''' Type your code here. ''' Run your program as often as you'd like, **Submit mode** Develop mode before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box. Enter program input (optional) If your code requires input values, provide them here. main.py **Run program** Input (from above) Output (shown be (Your

program)



3.15 LAB: Simple statistics

Given 4 floating-point numbers. Use a string formatting expression with conversion specifiers to output their product and their average as integers (rounded), then as floating-point numbers.

Output each rounded integer using the following:

```
print('{:.0f}'.format(your_value))
```

Output each floating-point value with three digits after the decimal point, which can be achieved as follows:

```
print('{:.3f}'.format(your_value))
```

Ex: If the input is:

9zyBooks 03/05/20 10:22 591419

```
8.3 UCSCCSE20NawabWinter2020
10.4
5.0
4.8
```

the output is:

```
2072 7
2071.680 7.125
```

LAB 3.15.1: LAB: Simple statistics ACTIVITY main.py Load default template... UCSCCSE20NawabWinter2020

1 num1 = float(input()) 2 num2 = float(input()) 3 num3 = float(input()) 5 ''' Type your code here. '''

Develop mode Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

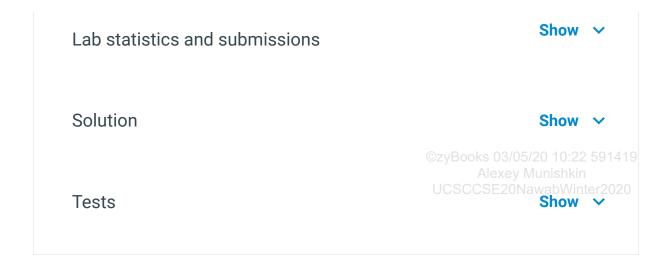
0/10

Enter program input (optional)

If your code requires input values, provide them here.

main.pvzyBooks 03/05/20 10:22 591419 (Your Output (shown be program) UCSCCSE20NawabWinter2020 **Run program** Input (from above)

Program output displayed here



3.16 LAB: Warm up: Creating passwords

(1) Prompt the user to enter two words and a number, storing each into separate variables. Then, output those three values on a single line separated by a space. (Submit for 1 point)

```
Enter favorite color:
yellow
Enter pet's name:
Daisy
Enter a number:
6
You entered: yellow Daisy 6
```

(2) Output two passwords using a combination of the user input. Format the passwords as shown below. (Submit for 2 points, so 3 points total).

```
Enter favorite color:

yellow
Enter pet's name:
Daisy
Enter a number:
6
You entered: yellow Daisy 6
```

```
First password: yellow_Daisy
Second password: 6yellow6
```

(3) Output the length of each password (the number of characters in the strings).

(Submit for 2 points, so 5 points total).

©zyBooks 03/05/20 10:22 59141

```
Enter favorite color:
yellow
Enter pet's name:
Daisy
Enter a number:
6
You entered: yellow Daisy 6

First password: yellow_Daisy
Second password: 6yellow6

Number of characters in yellow_Daisy: 12
Number of characters in 6yellow6: 8
```

LAB ACTIVITY

3.16.1: LAB: Warm up: Creating passwords

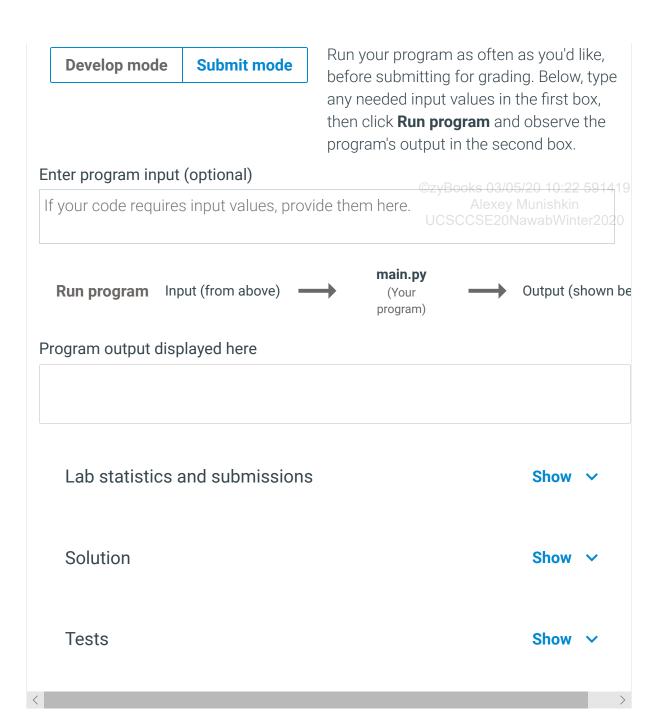
0/5

5

main.py

Load default template...

```
1 # FIXME (1): Finish reading another word and an integer into variables.
2 # Output all the values on a single line
3 favorite_color = input('Enter favorite color:\n')
4
5
6 # FIXME (2): Output two password options
7 password1 = favorite_color
8 print('\nFirst password:')
9
0
©zyBooks 03/05/20 10:22 591419
11 # FIXME (3): Output the length of the two password options Alexey Munishkin
12 |
```



©zyBooks 03/05/20 10:22 591419 Alexey Munishkin

3.17 LAB*: Program: Painting a wall wab Winter 2020

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:

print('{:.2f}'.format(your_value))

(1) Prompt the user to input a wall's height and width. Calculate and output the wall's area (integer). (Submit for 2 points).

```
Enter wall height (feet):

12
Enter wall width (feet):

15

Wall area: 180 square feet

CzyBooks 03/05/20 10:22 591419

Alexey Munishkin
UCSCCSE20NawabWinter2020
```

(2) Extend to also calculate and output the amount of paint in gallons needed to paint the wall (floating point). Assume a gallon of paint covers 350 square feet. Store this value in a variable. Output the amount of paint needed using the %f conversion specifier. (Submit for 2 points, so 4 points total).

```
Enter wall height (feet):

12
Enter wall width (feet):

15
Wall area: 180 square feet
Paint needed: 0.51 gallons
```

(3) Extend to also calculate and output the number of 1 gallon cans needed to paint the wall. Hint: Use a math function to round up to the nearest gallon. (Submit for 2 points, so 6 points total).

```
Enter wall height (feet):

12
Enter wall width (feet):

15
Wall area: 180 square feet
Paint needed: 0.51 gallons
Cans needed: 1 can(s)

©zyBooks 03/05/20 10:22 591419
Alexey Munishkin
```

(4) Extend by prompting the user for a color they want to paint the walls. Calculate and output the total cost of the paint cans depending on which color is chosen. Hint: Use a dictionary to associate each paint color with its respective cost. Red paint costs \$35 per gallon can, blue paint costs \$25 per gallon can, and green paint costs \$23 per gallon can. (Submit for 2 points, so 8 points total).

```
Enter wall height (feet):
12
Enter wall width (feet):
15
Wall area: 180 square feet
Paint needed: 0.51 gallons
Cans needed: 1 can(s)
Choose a color to paint the wall:
red
Cost of purchasing red paint: $35
```

LAB **ACTIVITY**

3.17.1: LAB*: Program: Painting a wall

0/8

main.py

Load default template...

```
1 import math
3 # Dictionary of paint colors and cost per gallon
4 paint colors = {
5
      'red': 35,
      'blue': 25,
6
7
      'green': 23
8 }
10 # FIXME (1): Prompt user to input wall's width
11 # Calculate and output wall area
12 wall_height = int(input('Enter wall height (feet):\n'))
13 print('Wall area:')
14
15 # FIXME (2): Calculate and output the amount of paint in gallons needed to paint th
17 # FIXME (3): Calculate and output the number of 1 gallon cans needed to paint the w
19 # FIXME (4): Calculate and output the total cost of paint can needed depending on c
20
```

Develop mode

Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above)	main.py (Your program)	Output (s	hown be
Program output displayed here			
		3/05/20 10:22 ey Munishkin !0NawabWint	
Lab statistics and submissions		Show	~
Solution		Show	~
Tests		Show	~
<			>