

Fundamentals of Computer Programming



Submitted by

Rishav Rauniyar

Sir

Section E 8711

BSc (Computing)

Submitted to

Krishna

Introduction to Programming

Lab Worksheet

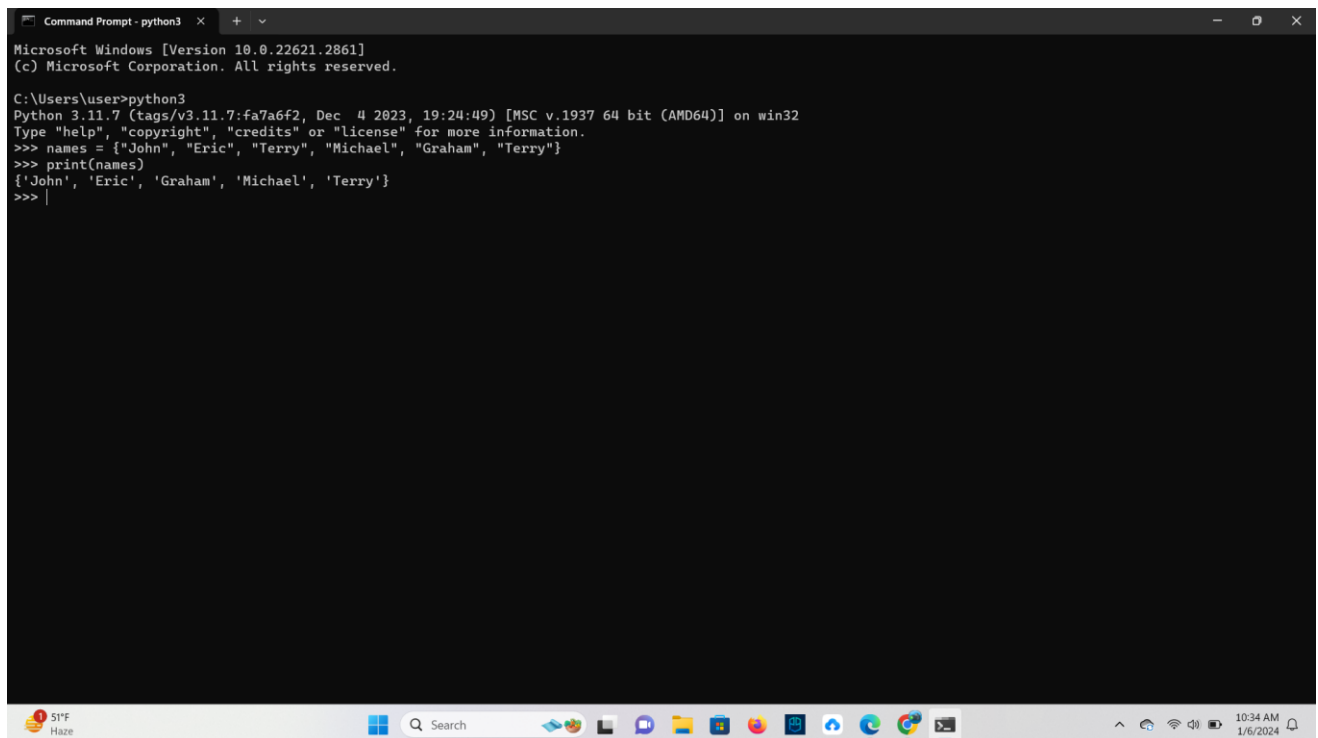
Week 7

Introducing Sets

TASK: Try creating a set by entering the code below. Then use the print () function to display the contents of the set. Notice how the output varies from the entered values.

```
names = {"John", "Eric", "Terry", "Michael", "Graham", "Terry"}
```

Ans

A screenshot of a Windows Command Prompt window titled "Command Prompt - python3". The window shows the execution of Python 3.11.7. The user enters the code to create a set named 'names' with the elements {"John", "Eric", "Terry", "Michael", "Graham", "Terry"} and then prints it. The output shows the set with the duplicate "Terry" removed: {'John', 'Eric', 'Graham', 'Michael', 'Terry'}. The Windows taskbar is visible at the bottom, showing the time as 10:34 AM on 1/6/2024.

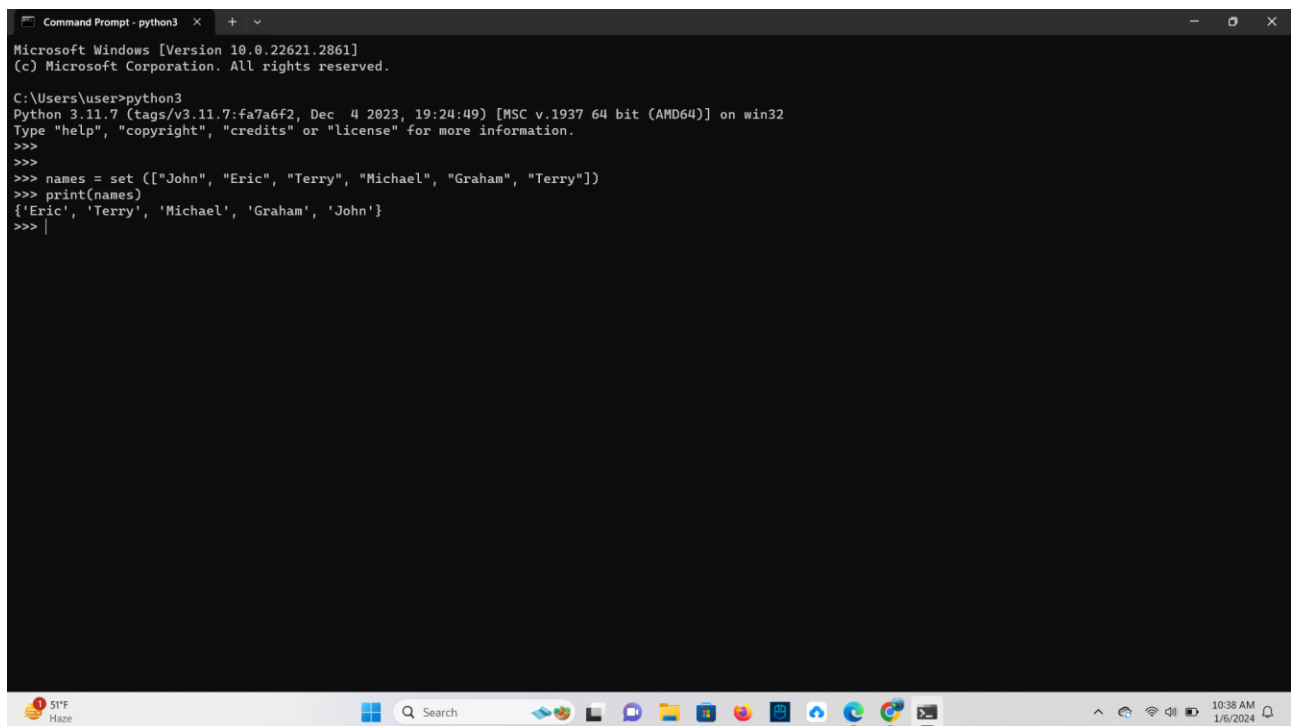
```
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> names = {"John", "Eric", "Terry", "Michael", "Graham", "Terry"}
>>> print(names)
{'John', 'Eric', 'Graham', 'Michael', 'Terry'}
>>> |
```

A set can also be created by calling the `set ()` constructor. A constructor is similar to a function but is used to initialize an object based on the named type. However, this constructor takes only a single parameter, which is iterated to extract the contents of the set. So, to create the above set using the `set ()` constructor the following code could be used:

```
names = set (["John", "Eric", "Terry", "Michael", "Graham", "Terry"])
```

Ans

A screenshot of a Windows Command Prompt window titled "Command Prompt - python3". The window shows the output of running Python 3.11.7. The user has entered the following code in the prompt:

```
>>> names = set (["John", "Eric", "Terry", "Michael", "Graham", "Terry"])
>>> print(names)
{'Eric', 'Terry', 'Michael', 'Graham', 'John'}
```

The output shows a set containing the names Eric, Terry, Michael, Graham, and John, with the duplicate "Terry" removed. The Windows taskbar is visible at the bottom, showing the date and time as 10:38 AM on 1/6/2024.

Notice how a List was created (using `[]`) and then passed as a single parameter. If multiple parameters had been passed, then an error would have been reported. Try entering the following and see the result -

```
names = set ("John", "Eric", "Terry", "Michael", "Graham", "Terry")
```

Ans

```
Command Prompt - python3 x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> names=set("John", "Eric", "Terry", "Michael", "Graham", "Terry")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: set expected at most 1 argument, got 6
>>> |
```

Creating a set using the constructor is convenient if the values already exist in some other iterable value, such as a List or Tuple. It is even possible to create a set of individual characters by passing a string type value. This means that the following statements both create the exactly the same set:

```
hex_letters = {"a", "b", "c", "d", "e", "f"}
```

```
hex_letters = set("abcdef")
```

Finally, an empty set must be created using the set () constructor rather than empty braces {}, since the latter creates an empty dictionary (see later).

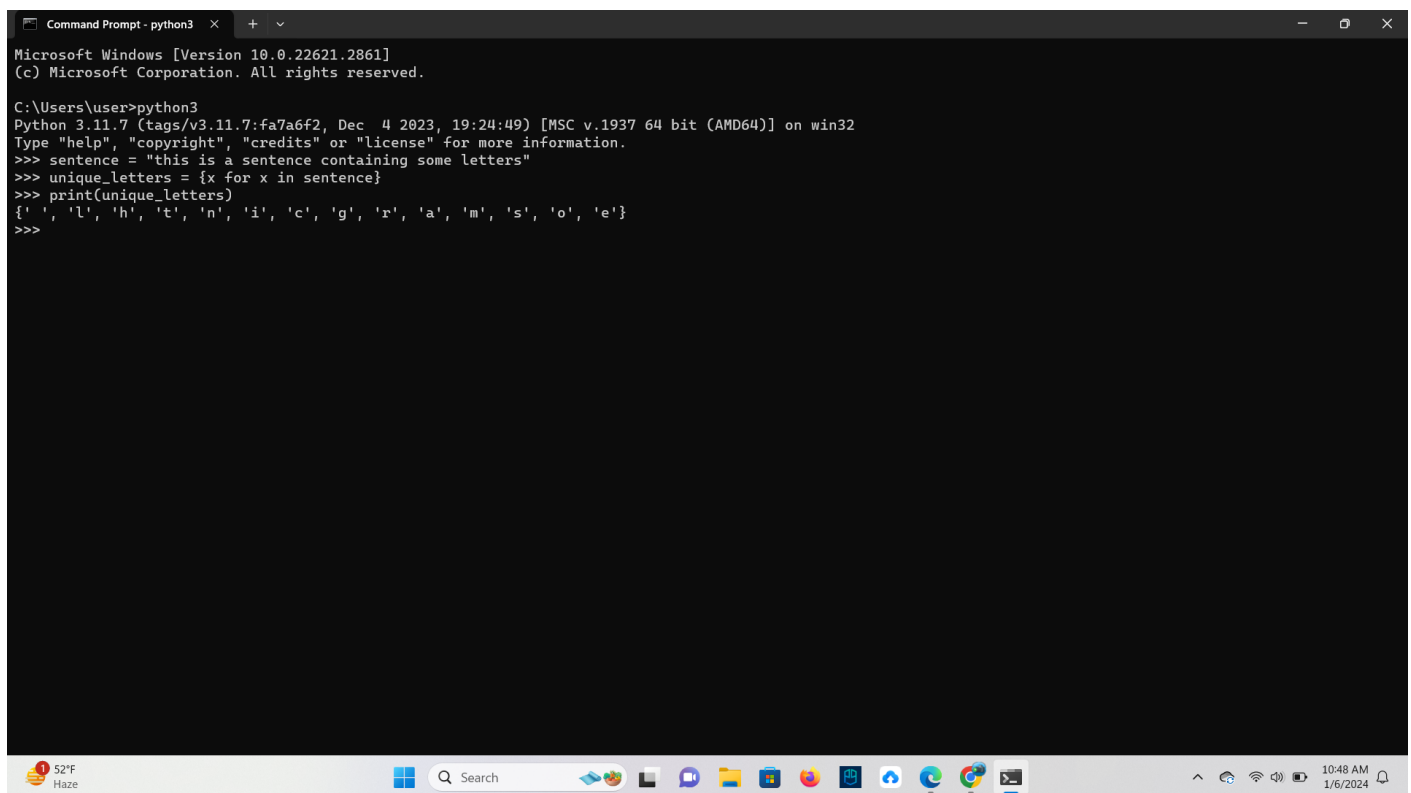
Set Comprehensions

TASK: Enter the code below, then make a call to the print () function to display the contents of the set.

```
sentence = "this is a sentence containing some letters"
```

```
unique_letters = {x for x in sentence}
```

Ans

A screenshot of a Windows Command Prompt window titled "Command Prompt - python3". The window shows the execution of Python code. The output of the print statement is a set of unique characters from the sentence: {' ', 'l', 'h', 't', 'n', 'i', 'c', 'g', 'r', 'a', 'm', 's', 'o', 'e'}. The Windows taskbar is visible at the bottom, showing the Start button, Search bar, and various application icons. The system tray on the right indicates the temperature is 52°F, the weather is Hazy, and the time is 10:48 AM on 1/6/2024.

```
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> sentence = "this is a sentence containing some letters"
>>> unique_letters = {x for x in sentence}
>>> print(unique_letters)
{' ', 'l', 'h', 't', 'n', 'i', 'c', 'g', 'r', 'a', 'm', 's', 'o', 'e'}
>>>
```

Set Operations

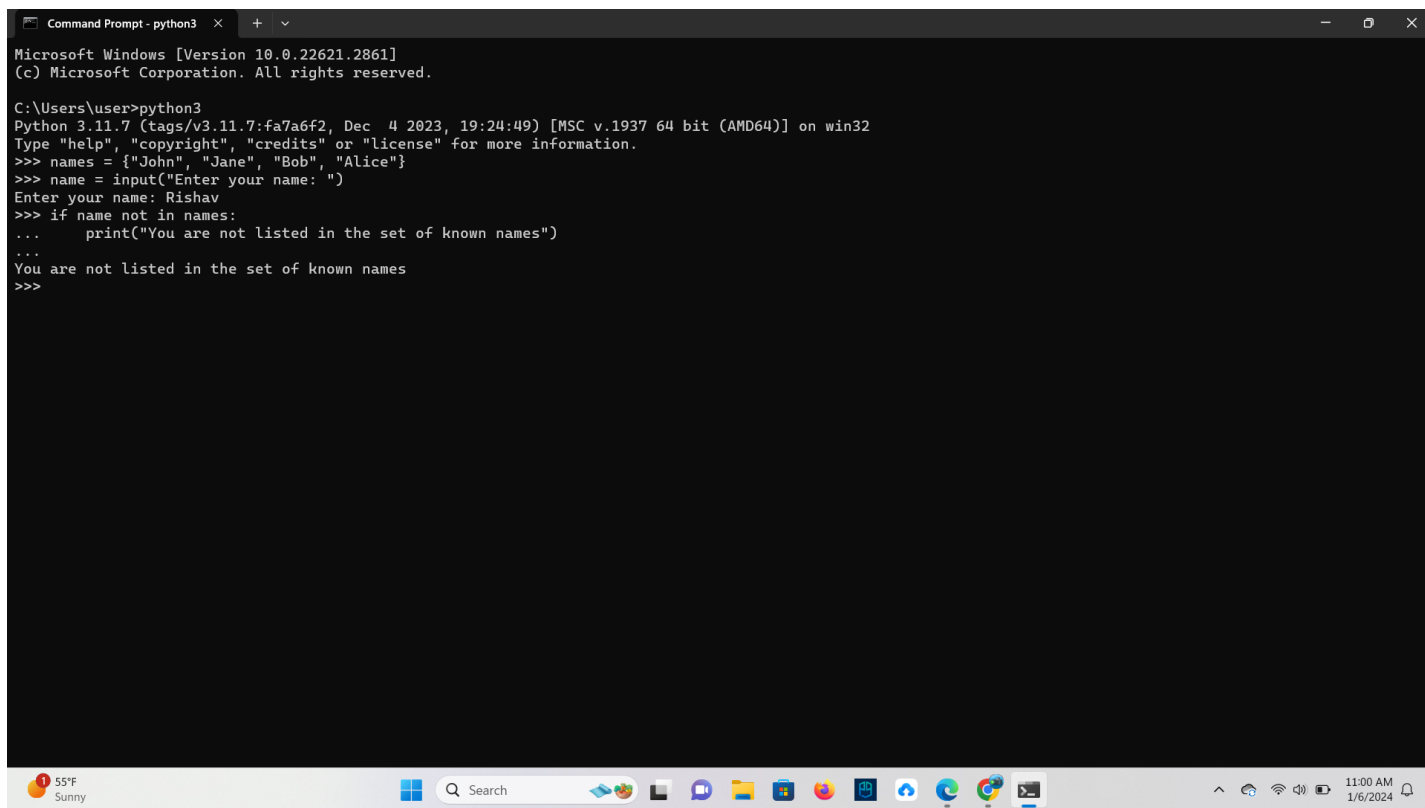
```
name = input("Enter your name: ")

if name in names:

    print("You are listed in the set of known names")
```

TASK: Rewrite the previous code so that it checks that the input name is NOT within the set of known names. Hint: use the not in operator.

Ans

A screenshot of a Windows Command Prompt window titled "Command Prompt - python3". The window shows the execution of a Python script. The script defines a set of known names and checks if an input name is not in that set. The user enters "Rishav", and the program outputs "You are not listed in the set of known names".

```
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> names = {"John", "Jane", "Bob", "Alice"}
>>> name = input("Enter your name: ")
Enter your name: Rishav
>>> if name not in names:
...     print("You are not listed in the set of known names")
...
You are not listed in the set of known names
>>>
```

TASK: Use the built-in help () function to view all the methods available on the set type.

Ans

```
Command Prompt - python3 x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> help(set)
Help on class set in module builtins:

class set(object)
|   set() -> new empty set object
|   set(iterable) -> new set object
|
|   Build an unordered collection of unique elements.
|
|   Methods defined here:
|
|   __and__(self, value, /)
|       Return self&value.
|
|   __contains__(...)
|       x.__contains__(y) <==> y in x.
|
|   __eq__(self, value, /)
|       Return self==value.
|
|   __ge__(self, value, /)
|       Return self>=value.
|
|   __getattr__(self, name, /)
|       Return getattr(self, name).
|
|   __gt__(self, value, /)
|       Return self>value.
|
|   __iand__(self, value, /)
-- More --
```

Breaking news
Get caught up

Search

TASK: Create the two initial sets, staff and directors as shown in the first example above. Perform the four mathematical set operations shown, but use the equivalent method calls to achieve the same results. For example:

```
staff = staff | {"Mark", "Ringo"}
```

becomes ...

```
staff = staff.union({"Mark", "Ringo"})
```

Ans

```
Command Prompt - python3 x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> staff = {"Pete", "Kelly", "Jon", "Paul", "Sally", "Sue"}
>>> directors = {"Kelly", "Rupert", "Cyril", "Jon"}
>>> a=staff.union(directors)
>>> b=staff.intersection(directors)
>>> c=staff.difference(directors)
>>> d=staff.symmetric_difference(directors)
>>> print("Union:",a)
Union: {'Kelly', 'Sue', 'Jon', 'Cyril', 'Paul', 'Rupert', 'Pete', 'Sally'}
>>> print("Intersection:",b)
Intersection: {'Kelly', 'Jon'}
>>> print("Difference:",c)
Difference: {'Sally', 'Pete', 'Paul', 'Sue'}
>>> print("Symmetric Difference:",d)
Symmetric Difference: {'Sue', 'Cyril', 'Paul', 'Rupert', 'Pete', 'Sally'}
>>> |
```


TASK: Create the set shown below then use the mutator version of the union method, which is `update ()` to add the two missing vowels to the set.

```
vowels = set ({"a", "e", "i"})
```

Ans

```
Command Prompt - python3  x  +  v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> vowels={"a","e","i"}
>>> missing_vowels={"o","u"}
>>> vowels.update(missing_vowels)
>>> print("Update Set:",vowels)
Update Set: {'e', 'u', 'a', 'i', 'o'}
>>>
```

Set Comparison Operations

TASK: Write code based on the previous two examples, but use the equivalent method calls to achieve the same results.

Ans

```
Command Prompt - python3 x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> staff = {"Pete", "Kelly", "Jon", "Paul", "Sally", "Sue"}
>>> managers = {"Kelly", "Jon", "Paul", "Sally", "Sue"}
>>> if managers.issubset(staff):
...     print("All Manager are staff Members")
...
All Manager are staff Members
>>> if staff.issuperset(managers):
...     print("All Manager are staff Members")
...
All Manager are staff Members
>>>
```

Creating an Immutable Set

TASK: Use the built-in help () function to view all the methods available on the frozen set type.

Ans

```
Command Prompt - python3 x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> help(frozenset)
Help on class frozenset in module builtins:

class frozenset(object)
|   frozenset() -> empty frozenset object
|   frozenset(iterable) -> frozenset object
|
|   Build an immutable unordered collection of unique elements.
|
|   Methods defined here:
|
|   __and__(self, value, /)
|       Return self&value.
|
|   __contains__(...)
|       x.__contains__(y) <==> y in x.
|
|   __eq__(self, value, /)
|       Return self==value.
|
|   __ge__(self, value, /)
|       Return self>=value.
|
|   __getattr__(self, name, /)
|       Return getattr(self, name).
|
|   __gt__(self, value, /)
|       Return self>value.
|
|   __hash__(self, /)
|
-- More --
```

Introducing Dictionaries

The Dictionary data-type stores multiple values like the other collection types. However, what is distinct about a dictionary is that it stores elements as pairs, often called a key: value pair.

Dictionaries are ordered and mutable and can have key: value pairs added and removed after initial creation.

Each key is unique and is associated with a single value. i.e. a key maps to a value just like a word in a conventional language dictionary maps to a definition. The set of keys must be unique - each key can appear at most once. However, the values in the dictionary do not need to be unique, therefore different keys can be mapped to the same value.

Dictionaries are useful when a value needs to be located quickly given a known key. For example, a dictionary may store a collection of customer records (values), and the key to these is the unique customer number.

Since the Dictionary type is built directly into the Python language there is a specific syntax associated with creation and manipulation. This is similar to what we have already seen with a Set. The main difference however is that operations involving Dictionaries usually involve pairs of values. To create a Dictionary directly, braces (curly brackets) can be used:

```
stock = {"apple":10, "banana":15, "orange":11}
```

This is similar to the notation for a Set, but the contents of the brackets show that it is a dictionary being created. Notice that a pair is specified for each entry separated by a colon ':', in the form key: value.

The key of the first element within the example is the string "apple", and the value associated with this is the number 10. Unlike a set the insertion ordering of the elements is maintained (from Python version 3.7 at least).

A dictionary can also be created by calling the dict() constructor. This can be called with various types of arguments and is somewhat more flexible than the set () constructor. To create the above dictionary, any of the following variations could be used:

```
# create by passing a dictionary stock = dict({"apple":10, "banana":15, "orange":11})
```

```
# create by passing keywords (only possible if keys are strings) stock = dict(apple=10, banana=15, orange=11)
```

```
# create by passing list of tuples stock = dict([("apple",10), ("banana",15), ("orange",11)])
```

Finally, an empty dictionary can be created using either empty braces {} or the dict() constructor with no arguments. This explains why, as mentioned earlier, an empty Set can ONLY be created using the set () constructor with no arguments.

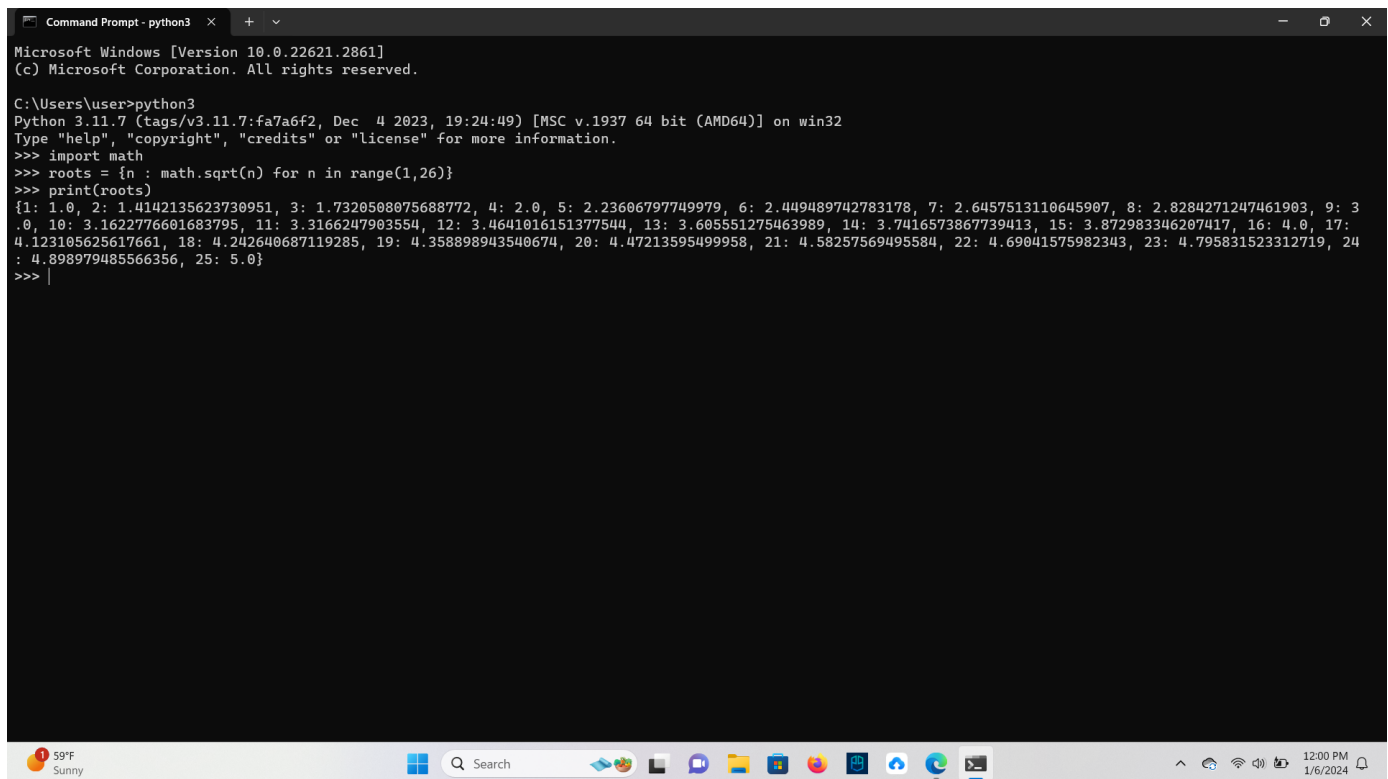
Dictionary Comprehensions

TASK: Enter the code below, then make a call to the print () function to display the contents of the dictionary.

```
import math

roots = {n: math.sqrt(n) for n in range (1,26)}
```

Ans



```
Command Prompt - python3
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import math
>>> roots = {n : math.sqrt(n) for n in range(1,26)}
>>> print(roots)
{1: 1.0, 2: 1.4142135623730951, 3: 1.7320508075688772, 4: 2.0, 5: 2.23606797749979, 6: 2.449489742783178, 7: 2.6457513110645907, 8: 2.8284271247461903, 9: 3.0, 10: 3.1622776601683795, 11: 3.3166247903554, 12: 3.4641016151377544, 13: 3.605551275463989, 14: 3.7416573867739413, 15: 3.872983346207417, 16: 4.0, 17: 4.123105625617661, 18: 4.242640687119285, 19: 4.358898943540674, 20: 4.47213595499958, 21: 4.58257569495584, 22: 4.69041575982343, 23: 4.795831523312719, 24: 4.898979485566356, 25: 5.0}
>>>
```

Manipulating Dictionaries

TASK: Write some code which adds a new fruit called "kiwi" to the stock dictionary, with an initial stock level of 10.

Ans

```
Command Prompt - python3
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> stock = {
...     "apple": 20,
...     "banana": 15,
...     "orange": 25
... }
>>> a="kiwi"
>>> initial_stock_level = 10
>>> stock[a]=initial_stock_level
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'initial_stock_level' is not defined. Did you mean: 'initial_stock_level'?
>>> stock[a]=initial_stock_level
>>> print("Updated Stock Dictionary:")
Updated Stock Dictionary:
>>> print(stock)
{'apple': 20, 'banana': 15, 'orange': 25, 'kiwi': 10}
>>>
```

Dictionary Methods

TASK: Use the built-in help () function to view all the methods available on the dict type.

Then write some code that uses the `popitem()` method to remove some key: value pairs from the stock dictionary.

Ans

```
Command Prompt - python3
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> help(dict)
Help on class dict in module builtins:

class dict(object)
| dict() -> new empty dictionary
| dict(mapping) -> new dictionary initialized from a mapping object's
|   (key, value) pairs
| dict(iterable) -> new dictionary initialized as if via:
|   d = {}
|   for k, v in iterable:
|       d[k] = v
| dict(**kwargs) -> new dictionary initialized with the name=value pairs
|   in the keyword argument list. For example: dict(one=1, two=2)
|
| Methods defined here:
|
| __contains__(self, key, /)
|     True if the dictionary has the specified key, else False.
|
| __delitem__(self, key, /)
|     Delete self[key].
|
| __eq__(self, value, /)
|     Return self==value.
|
| __ge__(self, value, /)
|     Return self>=value.
|
| __getattr__(self, name, /)
|     Return getattr(self, name).
-- More --
```

```
Command Prompt - python3
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> stock = {"apple":10, "banana":15, "orange":11}
>>> stock.popitem()
('orange', 11)
>>> print(thisdict)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'thisdict' is not defined
>>> print(stock)
{'apple': 10, 'banana': 15}
>>>
```

59°F
Partly sunny



Search



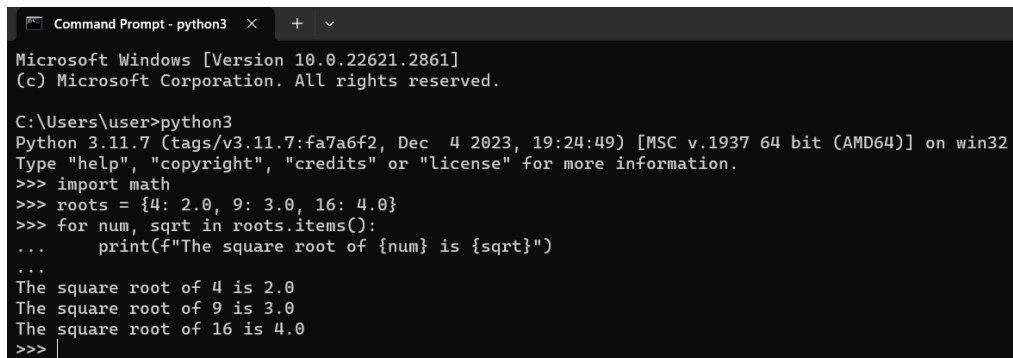
Iterating over Dictionaries

TASK: Write some code that iterates over the contents of the roots dictionary created within an earlier task. For each entry, print the message -

“The square root of is”

Where shows the number and shows the square root of that number.

Ans



```
Command Prompt - python3
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python3
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> roots = {4: 2.0, 9: 3.0, 16: 4.0}
>>> for num, sqrt in roots.items():
...     print(f"The square root of {num} is {sqrt}")
...
The square root of 4 is 2.0
The square root of 9 is 3.0
The square root of 16 is 4.0
>>> |
```

Key Terminology

TASK: Look at each of the phrases below and ensure you understand what each of these means. For any that you do not understand, do a little research to find a definition of each term. This research may involve looking back over these notes, or the associated lecture notes. It may also involve searching for these terms on the Internet.

- Set

Ans It is a 4 built in data type in python used to store several items in a single variable using curly bracket.

- Set operations

Ans Python Set provides different built-in methods to perform mathematical set operations like union, intersection, subtraction, and symmetric difference.

- Set comprehension

Ans Set comprehension may be a concise and capable way to make a new set based on an existing iterable object, such as a list, a tuple, or a run.

- Dictionary

Ans It is the data type in python which is used to store values in key: value pair using comma and curly bracket in a single variable.

- key: value pair

Ans Values which is found inside the data type of python dictionary and that include two pieces of data that is keys and their values.