Python Bootcamp Session #1

The DA Programming Club

What is Python?

“Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.”

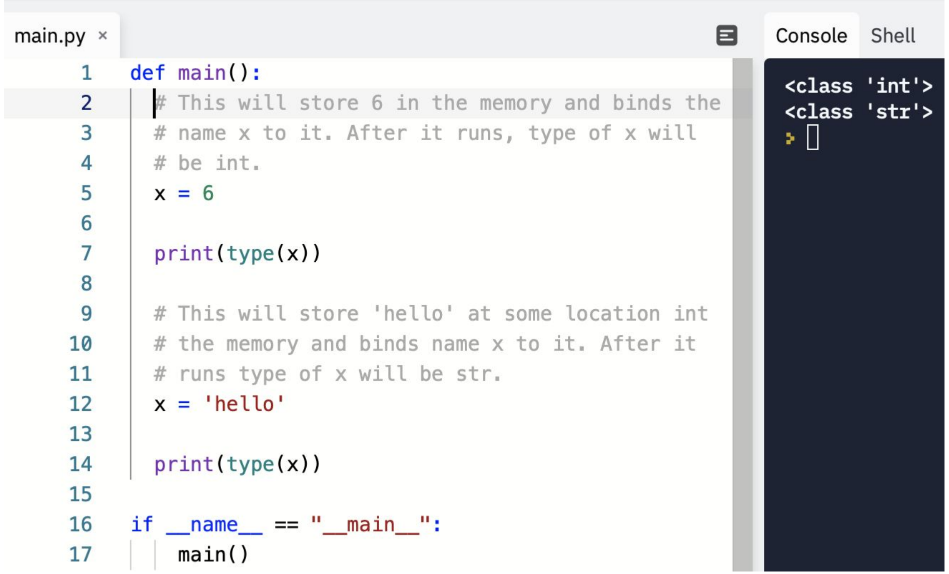
- Interpreted: Runs through the program line by line and executes each command, without any previous compilation of the code into machine-language instructions.
- Object-oriented: organized around classes and objects instead of functions and logic.
- Dynamic semantics: a variable’s type can change over its lifetime within a single program.

Dynamic Semantics

"When we declare a variable in C or alike languages, this sets aside an area of memory for holding values allowed by the data type of the variable. The memory allocated will be interpreted as the data type suggests. If it's an integer variable the memory allocated will be read as an integer and so on. When we assign or initialize it with some value, that value will get stored at that memory location. At compile time, initial value or assigned value will be checked. So we cannot mix types. Example: initializing a string value to an int variable is not allowed and the program will not compile.

But Python is a dynamically typed language. It doesn't know about the type of the variable until the code is run. So declaration is of no use. What it does is, It stores that value at some memory location and then binds that variable name to that memory container. And makes the contents of the container accessible through that variable name. So the data type does not matter. As it will get to know the type of the value at run-time."

https://www.geeksforgeeks.org/why-python-is-called-dynamically-typed/



```python
def main():
    # This will store 6 in the memory and binds the
    # name x to it. After it runs, type of x will
    # be int.
    x = 6

    print(type(x))

    # This will store 'hello' at some location int
    # the memory and binds name x to it. After it
    # runs type of x will be str.
    x = 'hello'

    print(type(x))

if __name__ == "__main__":
    main()
```

```
<class 'int'>
<class 'str'>
>
```

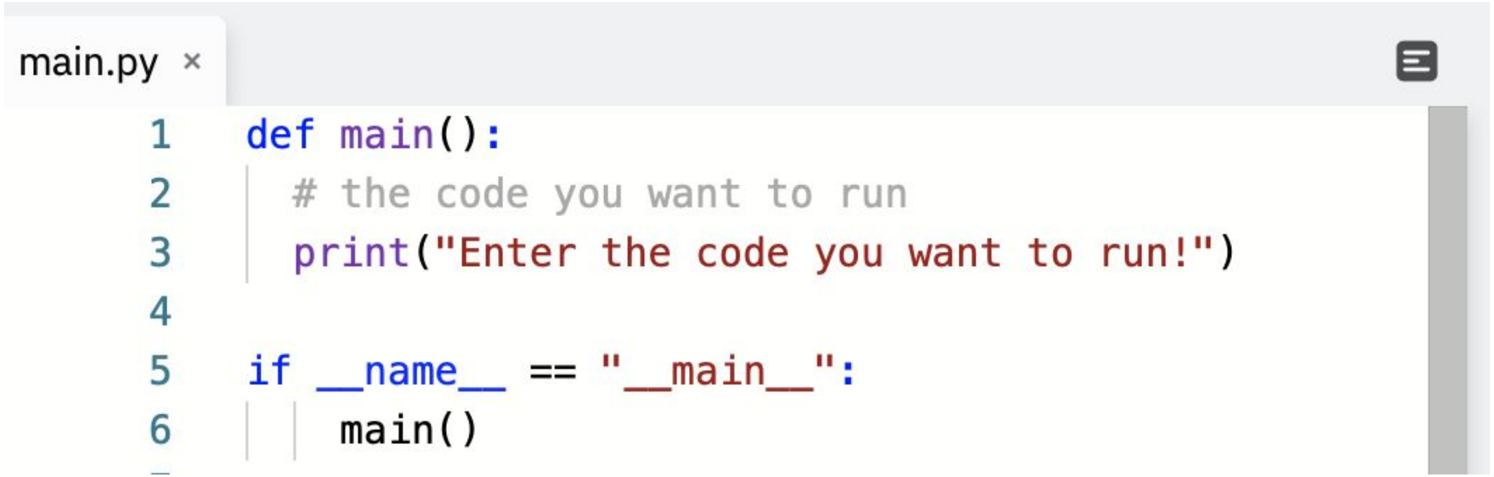
Structure of a Python Program

Many programming languages have a special function that is automatically executed when an operating system starts to run a program. This function is usually called main() and must have a specific return type and arguments according to the language standard. On the other hand, the Python interpreter executes scripts starting at the top of the file, and there is no specific function that Python automatically executes.

Nevertheless, having a defined starting point for the execution of a program is useful for understanding how a program works. Python programmers have come up with several conventions to define this starting point.

https://realpython.com/python-main-function/

Structure of a Python Program: Main


main.py ×

```python
def main():
  # the code you want to run
  print("Enter the code you want to run!")

if __name__ == "__main__":
    main()
```

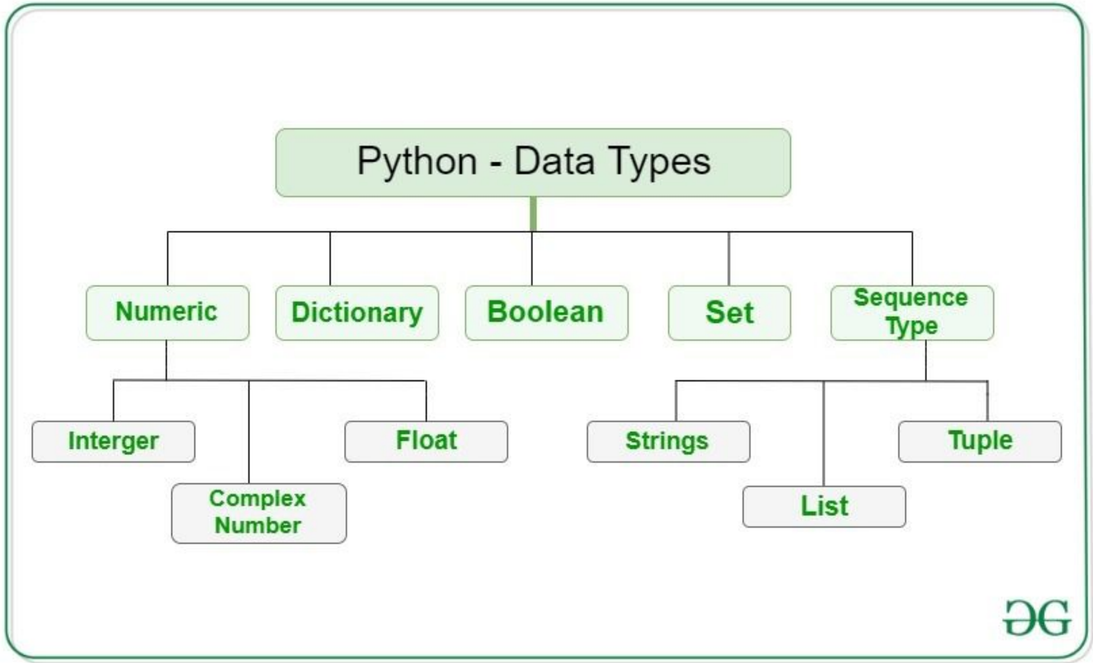
Try turning this structure into a “Hello World!” output of your own!

(replit.com is a good online IDE for Python)

Variables/Data Types

Important note: everything in Python is an object. So data types are classes and variables are instances of those classes (objects).

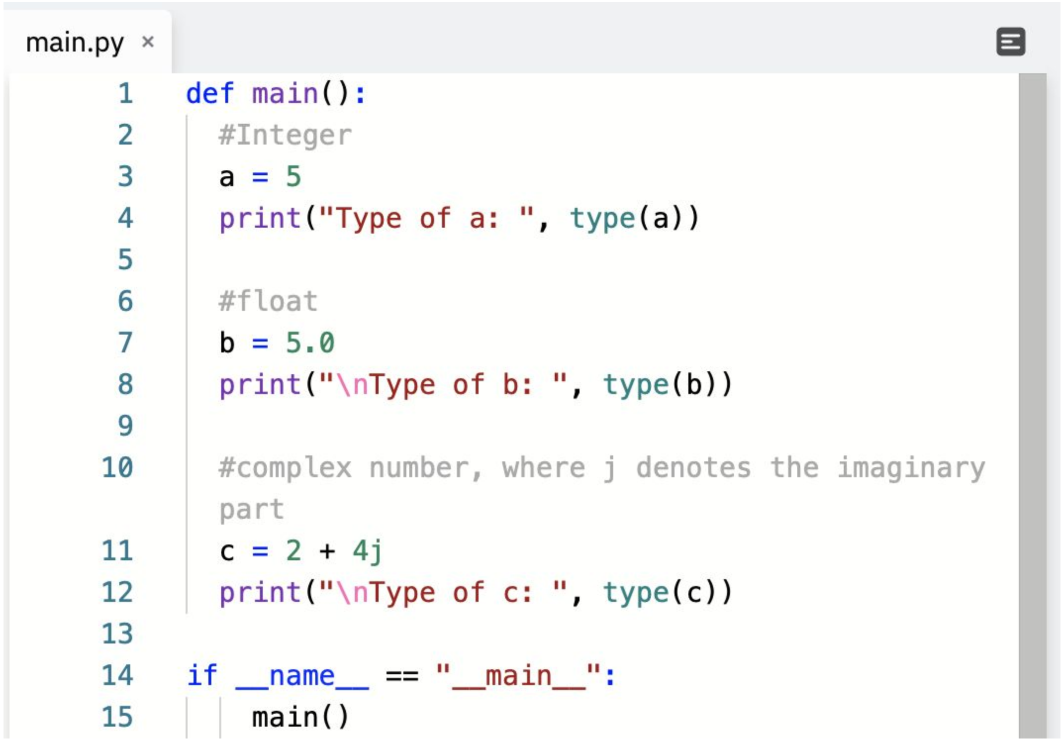
More in-depth resource: https://realpython.com/python-data-types/



Numeric

Three types:

1. Integer:
   a. positive or negative whole numbers. There is no limit to how long an integer value can be in Python.
2. Float:
   a. a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
3. Complex Number:
   a. specified as (real part) + (imaginary part)j. For example – 2+3j



main.py ×

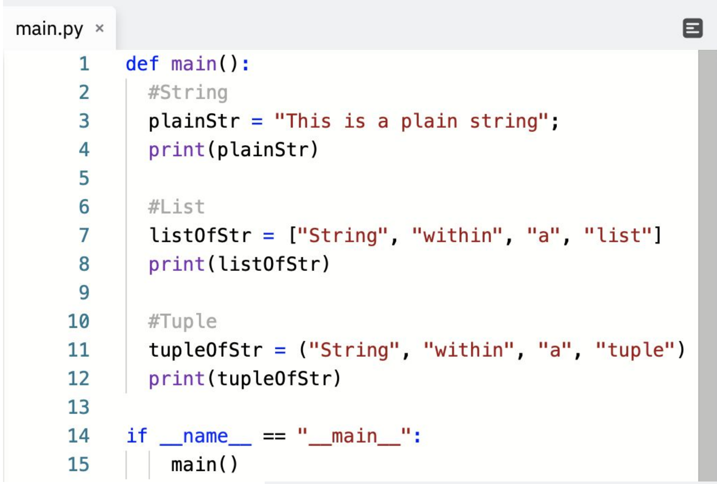
```python
def main():
    #Integer
    a = 5
    print("Type of a: ", type(a))

    #float
    b = 5.0
    print("\nType of b: ", type(b))

    #complex number, where j denotes the imaginary part
    c = 2 + 4j
    print("\nType of c: ", type(c))

if __name__ == "__main__":
    main()
```

Sequence Type

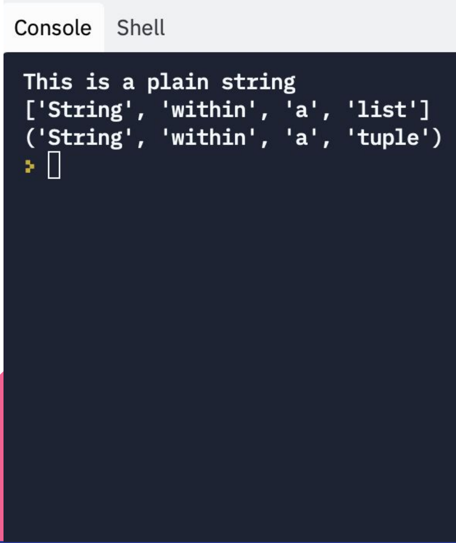
Three types:

1. String
    a. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by str class.
2. List
    a. An ordered collection of data. They are very flexible as the items in a list do not need to be of the same type.
3. Tuple
    a. Like a list, a tuple is an ordered collection of Python objects. The only difference between tuple and list is that tuples are immutable i.e. tuples cannot be modified after they are created.

Lists and tuples can both be accessed using square brackets and the index of choice.



```python
def main():
  #String
  plainStr = "This is a plain string";
  print(plainStr)

  #List
  listOfStr = ["String", "within", "a", "list"]
  print(listOfStr)

  #Tuple
  tupleOfStr = ("String", "within", "a", "tuple")
  print(tupleOfStr)

if __name__ == "__main__":
    main()
```

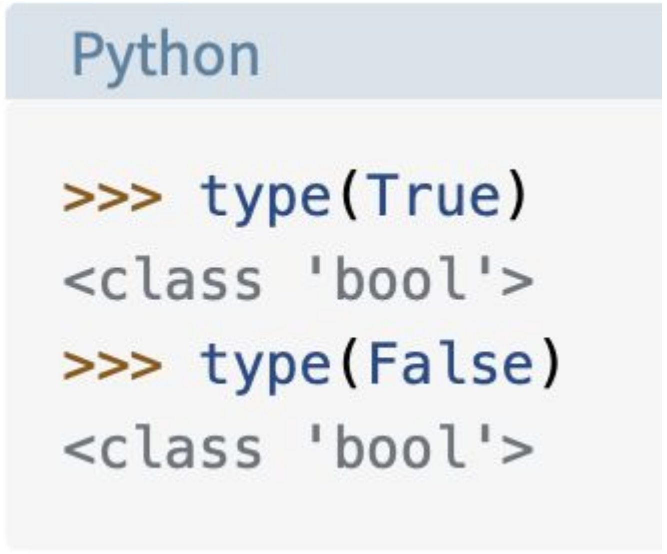


```
This is a plain string
['String', 'within', 'a', 'list']
('String', 'within', 'a', 'tuple')
> 
```

Boolean

A true or false value.

Expressions in Python are often evaluated in Boolean context, meaning they are interpreted to represent truth or falsehood. A value that is true in Boolean context is sometimes said to be “truthy,” and one that is false in Boolean context is said to be “falsy.” (You may also see “falsy” spelled “falsey.”)

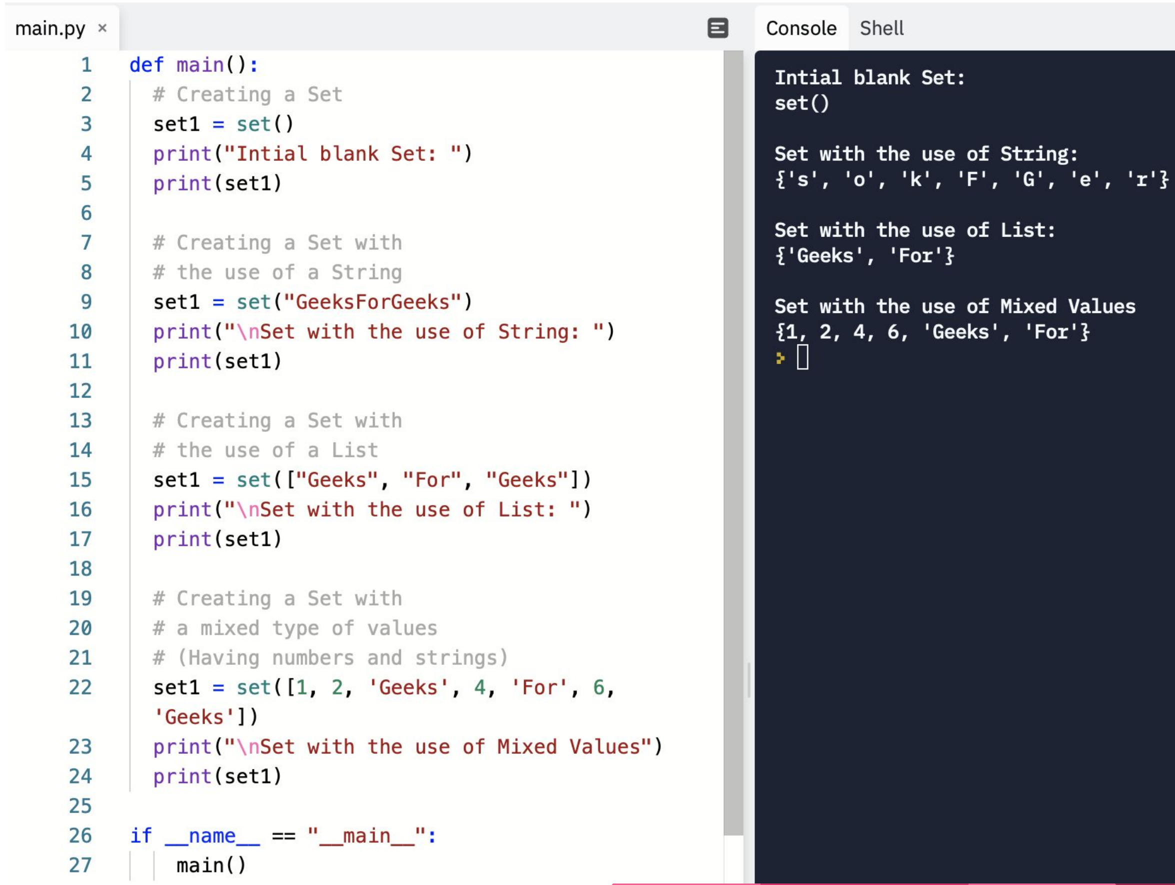

Python

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

Set

An unordered collection of data type that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

*Set items cannot be accessed by referring to an index, since sets are unordered the items has no index. But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.



main.py ×

```python
def main():
    # Creating a Set
    set1 = set()
    print("Intial blank Set: ")
    print(set1)

    # Creating a Set with
    # the use of a String
    set1 = set("GeeksForGeeks")
    print("\nSet with the use of String: ")
    print(set1)

    # Creating a Set with
    # the use of a List
    set1 = set(["Geeks", "For", "Geeks"])
    print("\nSet with the use of List: ")
    print(set1)

    # Creating a Set with
    # a mixed type of values
    # (Having numbers and strings)
    set1 = set([1, 2, 'Geeks', 4, 'For', 6,
'Geeks'])
    print("\nSet with the use of Mixed Values")
    print(set1)

if __name__ == "__main__":
    main()
```

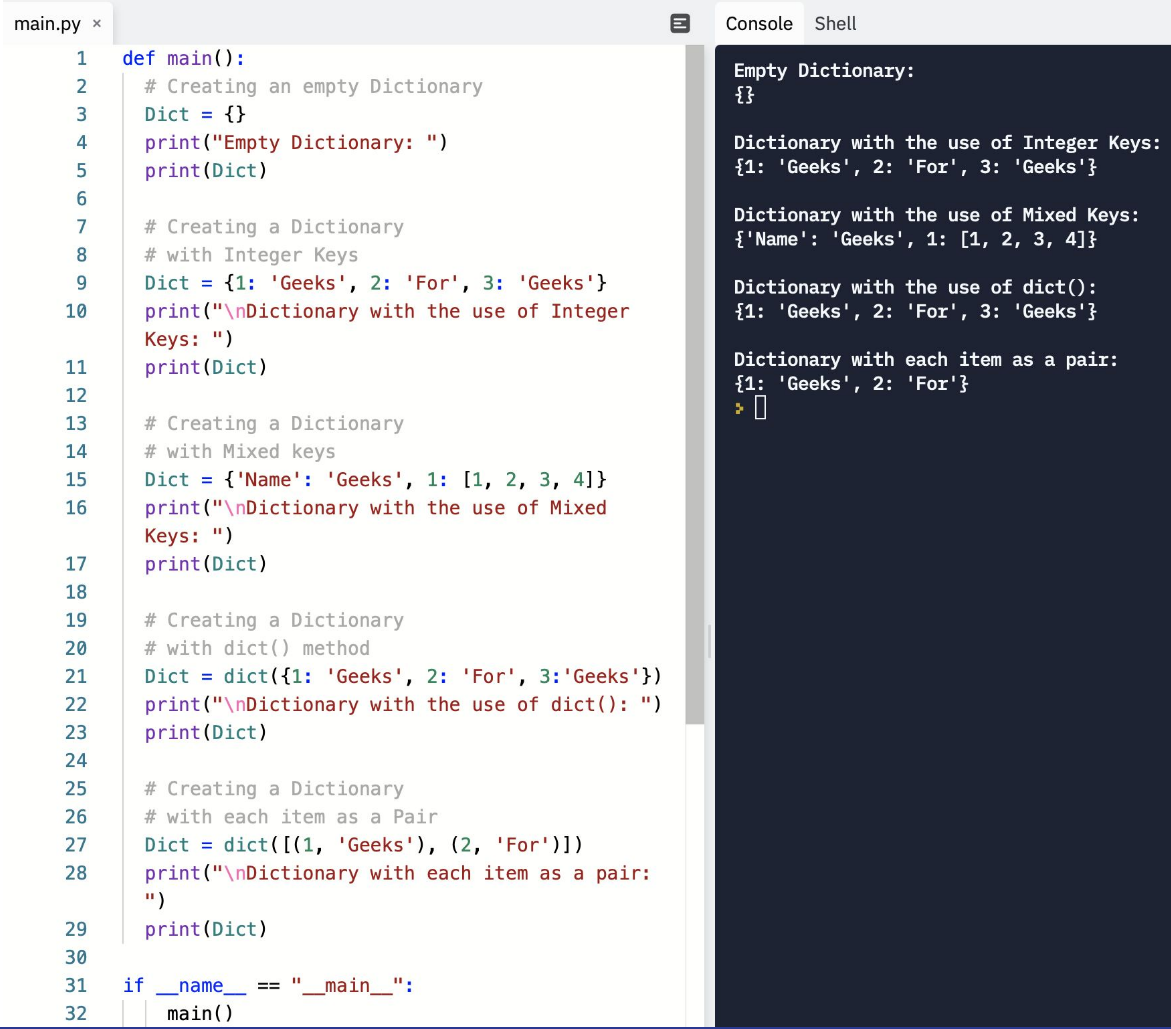
Console Shell

```
Intial blank Set:
set()

Set with the use of String:
{'s', 'o', 'k', 'F', 'G', 'e', 'r'}

Set with the use of List:
{'Geeks', 'For'}

Set with the use of Mixed Values
{1, 2, 4, 6, 'Geeks', 'For'}
>
```

Dictionary

An unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a 'comma'.

*.get() is a good way to access keys since it allows you to specify a default value if the key is missing dictionary.get("bogus", default_value)



main.py ×

```python
def main():
    # Creating an empty Dictionary
    Dict = {}
    print("Empty Dictionary: ")
    print(Dict)

    # Creating a Dictionary
    # with Integer Keys
    Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
    print("\nDictionary with the use of Integer Keys: ")
    print(Dict)

    # Creating a Dictionary
    # with Mixed keys
    Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}
    print("\nDictionary with the use of Mixed Keys: ")
    print(Dict)

    # Creating a Dictionary
    # with dict() method
    Dict = dict({1: 'Geeks', 2: 'For', 3:'Geeks'})
    print("\nDictionary with the use of dict(): ")
    print(Dict)

    # Creating a Dictionary
    # with each item as a Pair
    Dict = dict([(1, 'Geeks'), (2, 'For')])
    print("\nDictionary with each item as a pair: ")
    print(Dict)

if __name__ == "__main__":
    main()
```

Console Shell

```
Empty Dictionary:
{}

Dictionary with the use of Integer Keys:
{1: 'Geeks', 2: 'For', 3: 'Geeks'}

Dictionary with the use of Mixed Keys:
{'Name': 'Geeks', 1: [1, 2, 3, 4]}

Dictionary with the use of dict():
{1: 'Geeks', 2: 'For', 3: 'Geeks'}

Dictionary with each item as a pair:
{1: 'Geeks', 2: 'For'}
>
```

Downloading/Selecting an IDE

Pycharm https://www.jetbrains.com/pycharm/download/#section=mac

and for your student license

https://www.jetbrains.com/community/education/#students

Google Colab https://colab.research.google.com

Replit replit.com

More Information about Data Types

A personal document updated irregularly by Wil:
https://docs.google.com/document/d/1Oz5ARGsI1PN1sRY1v_LxwWa2JrNYM5-24CceSn9c0rk/edit#

or

shorturl.at/ntCW2