LIST, Tuple, Set,Dictionary

|  |  |
| --- | --- |
| **Python List**   * In Python, a list is a sequence of items in an order. We create lists by placing items inside square brackets [] separated by commas.   # a list of integers  numbers = [1, 5, 6, -4]  print(numbers)  # mixed list  random\_list = [2.5, "Hello", -5, 2.5]  print(random\_list)  # empty list  list1 = []  print(list1) | OUTPUT  [1, 5, 6, -4]  [2.5, 'Hello', -5, 2.5]  [] |
| # a list of integers  numbers = [1, 5, 6, -4]  print(len(numbers))  # empty list  list1 = []  print(len(list1)) | 4  0 |
| languages = ["Python", "JavaScript", "C++", "Kotlin"]  print(languages) | ['Python', 'JavaScript', 'C++', 'Kotlin'] |
| languages = ["Python", "JavaScript", "C++", "Kotlin"]  # first item  print(languages[0])  # third item  print(languages[2]) | Python  C++ |
| **Negative indexes:**   * In Python, we can also use negative indexing for sequences like lists. Using a negative index gives us items from the last**.**   languages = ["Python", "JavaScript", "C++", "Kotlin"]  # last element  print(languages[-1])  # third to last element  print(languages[-3]) | Kotlin  JavaScript |
| **Slicing of a list**   * It is also possible to access a whole section of items from the list, not just a single item.   languages = ["Python", "JavaScript", "C++", "Kotlin"]  # first three items -> 0, 1, 2  print(languages[0:3])  # second to last items -> 1, 2, 3  print(languages[1:4])  While using slicing, the starting index is inclusive but the ending index is exclusive. | ['Python', 'JavaScript', 'C++']  ['JavaScript', 'C++', 'Kotlin'] |
| languages = ["Python", "JavaScript", "C++", "Kotlin"]  # first to third element  print(languages[:3])  # second to last element  print(languages[1:]) | ['Python', 'JavaScript', 'C++']  ['JavaScript', 'C++', 'Kotlin'] |
| **Change Items of a List**  languages = ["Python", "JavaScript", "C++", "Kotlin"]  # modifying the second item  languages[1] = "Ruby"  print(languages) | ['Python', 'Ruby', 'C++', 'Kotlin'] |
| languages = ["Python", "JavaScript", "C++", "Kotlin"]  # modifying the second item to third item  languages[1:3] = ["Ruby", "Dart"]  print(languages) | ['Python', 'Ruby', 'Dart', 'Kotlin'] |
| **in keyword**  languages = ["Python", "JavaScript", "C++", "Kotlin"]  print("Python" in languages)  print("Rust" in languages) | True  False |
| **Iterating through a List**  languages = ["Python", "JavaScript", "C++", "Kotlin"]  for language in languages:  print(language) | Python  JavaScript  C++  Kotlin |
| **List Methods**  append() method  To add items to end of the list, we can use the append() method.  languages = ["Python", "JavaScript", "C++", "Kotlin"]  languages.append("Rust")  print(languages) | ['Python', 'JavaScript', 'C++', 'Kotlin', 'Rust'] |
| insert() method  To add items to a specified index in the list, we can use the insert() method.  languages = ["Python", "JavaScript", "C++", "Kotlin"]  languages.insert(1, "Rust")  print(languages) | ['Python', 'Rust', 'JavaScript', 'C++', 'Kotlin'] |
| remove() method  To remove specified item from a list, we can use the remove() method.  languages = ["Python", "JavaScript", "C++", "Kotlin"]  languages.remove("C++")  print(languages) | ['Python', 'JavaScript', 'Kotlin'] |
| copy() method  To create a copy of a list, we can use the copy() method.  languages = ["Python", "JavaScript", "C++", "Kotlin"]  languages1 = languages.copy()  print(languages1) | ["Python", "JavaScript", "C++", "Kotlin"] |

**Tuples:**

A tuple in Python is similar to a list. It is also an ordered collection of items.

The only difference between the two is that lists are mutable (elements of a list can be changed), whereas tuples are immutable (elements of a tuple cannot be changed).

To create a tuple, we need to wrap items inside parentheses () and separate items by a comma.

|  |  |
| --- | --- |
| numbers = (21, -5, 6, 9)  print(numbers) | (21, -5, 6, 9) |
| numbers = (1, 5, 6, 3)  print(numbers[2]) | 6 |
| numbers = (1, 5, 6, 3)  print(numbers[1:4]) | (5, 6, 3) |
| numbers = (1, 5, 6, 3)  for number in numbers:  print(number) | 1  5  6  3 |

**Python Dictionary**

* A dictionary is a collection of key/value pairs. It is similar to associative arrays in other programming languages.
* To create a dictionary, we put the key/value pairs separated by a colon : inside the curly braces {}.

|  |  |
| --- | --- |
| person1 = {"name": "Linus", "age": 21}  print(person1) | {"name": "Linus", "age": 21} |
| **Access Dictionary Elements**   * Dictionaries are optimized to get values when the key is known. * Similar to numbered indexes like 0, 1, 2 to get elements from sequences like lists and tuples, keys are used as indices for dictionaries.   person1 = {"name": "Linus", "age": 21}  print(person1["name"])  print(person1["age"]) | Linus  21 |
| person1 = {"name": "Linus", "age": 21}  print(person1["hobbies"]) | Traceback (most recent call last):  File "<string>", line 2, in <module>  KeyError: 'hobbies' |
| person1 = {"name": "Linus", "age": 21}  print(person1.get("namr"))  print(person1.get("hobbies")) | Linus  None |
| None which denotes empty or no value. This value can be used with if statement to make different decision as per the need.  We can also pass a second default argument to the get() method that will be returned instead of None if the key is not found.  person1 = {"name": "Linus", "age": 21}  print(person1.get("hobbies", ["dancing", "fishing"])) | ["dancing", "fishing"] |
| **Add and Change Dictionary Elements:**  person1 = {"name": "Linus", "age": 21}  # changing existing keys  person1["name"] = "Dennis"  print(person1)  # adding new keys  person1["hobbies"] = ["dancing", "fishing"]  print(person1) | {'name': 'Dennis', 'age': 21}  {'name': 'Dennis', 'age': 21, 'hobbies': ['dancing', 'fishing']} |
| **Remove Elements From a Dictionary**  To remove an item from the dictionary, we can use the dictionary's pop() method.  The pop() method also returns the value of the removed key.  person1 = {"name": "Linus", "age": 21}  print(person1.pop("name"))  print(person1) | Linus  {"age": 21} |
| **Iterating Through a Dictionary**   * Similar to sequences, we can easily iterate through items of a dictionary by using a for loop. We get one key in every iteration:   person1 = {"name": "Linus", "age": 21}  for key in person1:  print(key)  print(person1[key]) | name  Linus  age  21 |
| synonyms = {"mountain": "peak", "forest": "jungle"}  print("1.", synonyms["mountain"])  synonyms["terrain"] = "land"  print("2.", synonyms)  synonyms.pop("forest")  print("3.", synonyms) | 1. peak  2. {'mountain': 'peak', 'forest': 'jungle', 'terrain': 'land'}  3. {'mountain': 'peak', 'terrain': 'land'} |

Do yourself

|  |  |
| --- | --- |
| 1. Program to Find Even Numbers From a List | list1 = [2, 3, 7, 5, 10, 17, 12, 4, 1, 13]  for i in list1:  if i % 2 == 0:  print(i) |
| 1. Access value 20 from the following tuple.   aTuple = ("Orange", [10, 20, 30], (5, 15, 25)) | aTuple = ("Orange", [10, 20, 30], (5, 15, 25))  print(aTuple[1][1]) |
| 1. Copy element 44 and 55 from the following tuple into a new tuple   tuple1 = (11, 22, 33, 44, 55, 66) | tuple1 = (11, 22, 33, 44, 55, 66)  tuple2 = tuple1[3:-1]  print(tuple2) |
| 1. Write a Python program to check if a value 6 is present in a set or not. | nums = {1, 3, 5, 7, 9, 11}  print("Original sets(nums): ",nums,"\n")  print("Test if 6 exists in nums:")  print(6 in nums) |
| 1. Write a Python script to concatenate following dictionaries to create a new one.   Sample Dictionary :  dic1={1:10, 2:20}  dic2={3:30, 4:40}  dic3={5:50,6:60} | dic1={1:10, 2:20}  dic2={3:30, 4:40}  dic3={5:50,6:60}  dic4 = {}  for d in (dic1, dic2, dic3): dic4.update(d)  print(dic4)Top of Form |
| 1. WAP to read a set of numbers in an array & to find the largest of them. | alist=[-45,0,3,10,90,5,-2,4,18,45,100,1,-266,706]  largest=alist[0]  for large in alist:  if large > largest:  largest=large  print(largest) |

**Summary**

* A list is a collection of ordered items.
* To access individual elements of a list, we use indices.
* To access a portion of a list, we use the slicing operator.
* Python has several useful methods that can make it easier to add, change, insert and remove list elements.
* A tuple is similar to a list except tuples are immutable ; we can’t change elements of a tuple.