Fresher Academy

Python for Data Science 2

FPT Software | FRESHER ACADEMY

What Will You Learn

- List
- Tuple
- Dictionary
- Functions
- Modules
- File I/O
- Exceptions

FRESHER ACADEMY

Python for Data Science 2

FPT Software | FRESHER ACADEMY

List

List

Declare a list

- Different comma-separated values between square brackets
- Zero-based sequences

```
subjects = ['physics', 'chemistry', 'math'];
years = [2018, 2019, 2020, 2021, 2022];
chars = ["a", "b", "c", "d"]
```

List

Access values

- Use the square brackets for slicing with a certain index or indices to retrieve value(s) of the list

```
subjects = ['physics', 'chemistry', 'math'];
print(subjects[0])  # physics
print(subjects[2])  # math
print(subjects[-1]) # math
print(subjects[0:2])# ['physics', 'chemistry']
print(subjects[0:3])# ['physics', 'chemistry', 'math']
print(subjects[:])  # ['physics', 'chemistry', 'math']
```

FRESHER ACADEMY

List

Update values

- Assign one or multiple list element(s) by slicing indices
- Use append() method to add new elements to lists

```
subjects = ['physics', 'chemistry', 'math'];
subjects[1] = 'english'
subjects.append('arts')
print(subjects[1:4])# ['english', 'math', 'arts']
```

List

Delete values

- The del statement or the remove() method to remove a list element

```
subjects = ['physics', 'chemistry', 'math'];
del subjects[1];
subjects.remove('math')
print(subjects)# ['physics']
```

List

Basic List Operators

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	TRUE	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

FRESHER ACADEMY

List

Indexing and Slicing

- codes = ['Fresher', 'Academy', 'AI']

Python Expression	**Results**	**Description**
`codes[2]`	AI	Offsets start at zero
`codes[-2]`	Academy	Negative: count from the right
`codes[1:]`	['Academy', 'AI']	Slicing fetches sections

List

Built-in List Functions

- cmp(list1, list2) Compares elements of both lists.
- len(list) Gives the total length of the list.
- max(list) Returns item from the list with max value.
- min(list) Returns item from the list with min value.
- list(seq) Converts a tuple into list.

List

Built-in List Methods

- list.append(obj) Appends object obj to list
- list.count(obj) Returns count of how many times obj occurs in list
- list.extend(seq) Appends the contents of seq to list
- list.index(obj) Returns the lowest index in list that obj appears
- list.insert(index, obj) Inserts object obj into list at offset index
- list.pop(obj=list[-1])Removes and returns last object or obj from list
- list.remove(obj) Removes object obj from list
- list.reverse() Reverses objects of list in place
- list.sort([func]) Sorts objects of list, use compare func if given

Python for
Data Science 2

Software

FRESHER
ACADEMY

Tuple

Tuple

Declare a tuple

- A tuple is a sequence of immutable Python objects.
- Different comma-separated values between square parentheses
- Zero-based sequences
- To declare a tuple with a single value, we must include a comma

```
subjects = ('physics', 'chemistry', 'math');
years = (2018, 2019, 2020, 2021, 2022);
chars = ("a", "b", "c", "d")
```

Tuple

Access values

- Use the square brackets for slicing with a certain index or indices to retrieve value(s) of the tuple

```
subjects = ('physics', 'chemistry', 'math');
print(subjects[0])  # physics
print(subjects[2])  # math
print(subjects[-1]) # math
print(subjects[0:2])# ('physics', 'chemistry')
print(subjects[0:3])# ('physics', 'chemistry', 'math')
print(subjects[:])  # ('physics', 'chemistry', 'math')
```

FRESHER ACADEMY

Tuple

Update values

- Tuples are immutable so we cannot update or change values of elements

Tuple

Delete values

- Removing tuple elements is impossible.

Tuple

Basic Tuple Operators

Python Expression	Results	Description
`len((1, 2, 3))`	3	Length
`(1, 2, 3) + (4, 5, 6)`	(1, 2, 3, 4, 5, 6)	Concatenation
`('Hi!',) * 4`	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
`3 in (1, 2, 3)`	TRUE	Membership

`for x in (1, 2, 3):`
`     print(x)`
	1 2 3	Iteration

Tuple

Indexing and Slicing

- codes = ('Fresher', 'Academy', 'AI')

Python Expression	**Results**	**Description**
codes[2]	AI	Offsets start at zero
codes[-2]	Academy	Negative: count from the right
codes[1:]	['Academy', 'AI']	Slicing fetches sections

Tuple

Built-in List Functions & Methods

- cmp(tuple1, tuple2) Compares elements of both tuples.
- len(tuple) Gives the total length of the tuple.
- max(tuple) Returns item from the tuple with max value.
- min(tuple) Returns item from the tuple with min value.
- tuple(seq) Converts a list into tuple.

FRESHER ACADEMY

Python for
Data Science 2

Software

FRESHER
ACADEMY

Dictionary

Dictionary

Declare a dictionary

- Key-value storage, separated values by commas between two curly braces
- Each key is separated from its value by a colon (:)
- Keys must be unique and immutable (strings, numbers or tuples)
- Values can be any type

```
registration = {'Name': 'Python', 'Year': 2018, 'Class': 'AI'}
languages    = {1: 'Python', 2: 'Java', 3: 'C++'}
```

Dictionary

Access values

- Use the square brackets with the key to retrieve its value
- Retrieving value of a non-existing key will lead to KeyError

```
registration = {'Name': 'Python', 'Year': 2018, 'Class': 'AI'}
print(registration ['Name'])   # Python
print(registration ['Year'])   # 2018
```

Dictionary

Update values

- Could add new entries with new key-value pairs
- Could modify existing items by assigning new values

```
registration = {'Name': 'Python', 'Year': 2018, 'Class': 'AI'}
registration['Name'] = 'Data'
registration['School'] = 'FA'
print(registration ['Name'])   # Data
print(registration ['School']) # FA
```

FRESHER ACADEMY

Dictionary

Delete values

- Use the del statement to remove items
- Use the clear() method to all entries

```
registration = {'Name': 'Python', 'Year': 2018, 'Class': 'AI'}
del registration ['Name']; # remove entry with key 'Name'
registration.clear();      # remove all entries in registration
del dict ;                 # delete entire dictionary
```

Dictionary

Properties of Dictionary Keys

- No duplicate key
- Keys must be immutable

Dictionary

Built-in List Functions

- cmp(dict1, dict2) Compares elements of both dict.
- len(dict) Gives the number of items of the dictionary.

Dictionary

Built-in List Methods

- dict.clear() Removes all elements of dictionary dict
- dict.copy() Returns a shallow copy of dictionary dict
- dict.fromkeys() Create a new dictionary with keys from seq and values set to value
- dict.get(key, default=None) Returns value or default if key not in dictionary
- dict.has_key(key) Returns true if key in dictionary dict, false otherwise
- dict.items() Returns a list of dict's (key, value) tuple pairs
- dict.keys() Returns list of dictionary dict's keys
- dict.update(dict2) Adds dictionary dict2's key-values pairs to dict
- dict.values() Returns list of dictionary dict's values

Python for Data Science 2

Functions

FPT Software | FRESHER ACADEMY

Functions

Define functions

```
def function_name(parameters):
        "function_doc"
        function_statements
        return [expression]
```

```
def sayHello(name):
        "To say hello to a name as a passed string"
        print("Hello " + name)
        return
def add(a, b):
        "To sum of absolute values of both numbers"
        return abs(a) + abs(b)
```

Functions

Call functions

- Call a function by using its name and necessary parameters

```
sayHello("Python") # Hello Python
add(-3, 4)         # 7
```

FRESHER ACADEMY

Functions

Pass by reference

- All parameters (arguments) in the Python language are passed by reference

```
def double(a, repeat):
        "To repeat input"
        a *= repeat
        return True
scores = [1, 2, 3]
double(scores, 2) # True
print(scores)     # [1, 2, 3, 1, 2, 3]
```

Functions

Function arguments

- Keyword arguments: the caller identifies the arguments by the parameter name.
- Default arguments: Default values are used when parameter values are not provided
- Variable-length arguments: An asterisk (*) is placed before a variable name holding values of all non-keyword variable arguments.

Functions

Return statements

- Return a specific value
- A return statement with no arguments ~ return None.

FRESHER ACADEMY

Functions

Anonymous functions

- Use the lambda keyword instead of the def keyword
- Can take any number of arguments
- Return only one value in the form of an expression.
- Cannot be a direct call to print because lambda requires an expression
- Can access only variables in the parameter list and the global namespace.

```
sum = lambda x, y: x + y;
print(sum(3, 4))  # 7
print(sum(-3, 4)) # 1
```

FRESHER ACADEMY

Python for
Data Science 2

Modules

Modules

The import statement

- Use an import statement to import/add any Python source file as a module
- A module is loaded only once when it is imported several times

```
# Import module training
import training
# Can call defined function startClass in training
training.startClass("AI")
```

FRESHER ACADEMY

Modules

The from…import statement

- Use from...import to import specific attributes from a module

```
# Import module training
from training import startClass
# Can call defined function startClass in training
startClass("AI")
```

```
# Import all functions in the module training
from training import *
```

FRESHER ACADEMY

Python for
Data Science 2

FRESHER ACADEMY

File I/O

File I/O

The input statement

- To read one line from standard input and returns it as a string

```
name = input("Enter your name: ");
print("Hi, ", name)
```

File I/O

The open statement

- fileObject = open(file_name [, access_mode][, buffering])

```
# Open a file
letter = open("letter.txt", "w")
```

FRESHER ACADEMY

File I/O

The access modes to open files

- r Opens a file for reading only.
- rb Opens a file for reading only in binary format.
- r+ Opens a file for both reading and writing.
- rb+ Opens a file for both reading and writing in binary format.
- w Opens a file for writing only.
- wb Opens a file for writing only in binary format.
- w+ Opens a file for both writing and reading.
- wb+ Opens a file for both writing and reading in binary format.
- a Opens a file for appending.
- ab Opens a file for appending in binary format.
- a+ Opens a file for both appending and reading.
- ab+ Opens a file for both appending and reading in binary format.

File I/O

The access modes to open files

- r, rb, r+ , rb+ The file pointer is placed at the beginning of the file.
- w, wb, w+, wb+ Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
- a, ab, a+, ab+ The file pointer is at the end of the file if the file exists. If the file does not exist, it creates a new file for writing.

File I/O

The file Object Attributes

- file.closed Returns true if file is closed, false otherwise.
- file.mode Returns access mode with which file was opened.
- file.name Returns name of the file.
- file.softspace Returns false if space explicitly required with print, true otherwise.

File I/O

The close() method

- fileObject.close();

```
# Close opened file
letter.close()
```

File I/O

Write files

- fileObject.write(contents);

```
# Open a file
letter = open("letter.txt", "w")
letter.write("Python is a great language.\nWelcome!!\n");
# Close opened file
letter.close()
```

File I/O

Read files

- fileObject.read();

```
# Open a file
letter = open("letter.txt", "r")
message = letter.read();
# Close opened file
letter.close()
```

File I/O

The os module

- os.rename(current_file_name, new_file_name)
- os.remove(file_name)
- os.mkdir("newdir"): to create directories in the current directory
- os.chdir("newdir"): to change the current directory
- os.getcwd(): to display the current working directory
- os.rmdir('dirname'): to delete the directory

File I/O

Rename files

```
import os
# Rename a file from letter.txt to message.txt
os.rename("letter.txt", "message.txt")
```

File I/O

Delete files

```python
import os
# Rename a file from letter.txt to message.txt
os.rename("letter.txt", "message.txt")
```

FRESHER ACADEMY

Python for Data Science 2

Exceptions

Exceptions

Assertions

- Often place assertions at the start of a function to check for valid input, and after a function call to check for valid output.
- assert Expression[, Arguments]

```
def area(x, y):
        assert (x > 0 and y > 0),"Input must be positive!"
        return x * y
```

Exceptions

Exception Handling

- An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.
- Handle exceptions by using try…except block statements

```
try:
        letter = open("letter.txt", "w")
        letter.write("Python is a great language.\nWelcome!!\n");
except IOError:
        print("Error: can\'t find file to write data")
else:
        print("Written content in the file successfully")
```

Fresher Academy

FPT Software | FRESHER ACADEMY

Happy Analyzing!