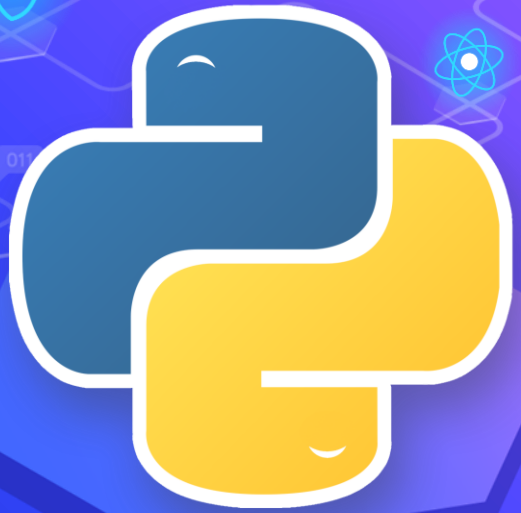
Introduction To Python Programming



Introduction to Python programming Course Outline

- Intro to Computer Science
- Environment Setup (Anaconda)
- Command Line
- Conda & pip package managers
- Jupyter Notebook
- Input & Output
- Variables
- Data types
- Numbers & Math
- Boolean & Comparison and Logic
- If Conditions
- For & While Loops
- Strings
- **Lists**
- Tuples
- Sets
- Dictionaries
- Advanced If and Loops
- List Comprehensions
- Dictionary Comprehensions
- Exceptions
- File Handling
- Functions
- Built-in functions & Operators (zip, enumerate, range, …)
- Map, Filter, Reduce
- Lambda Expressions

Lists

Lists are the most common data structure in Python

You can store multiple values (elements) inside a single variable

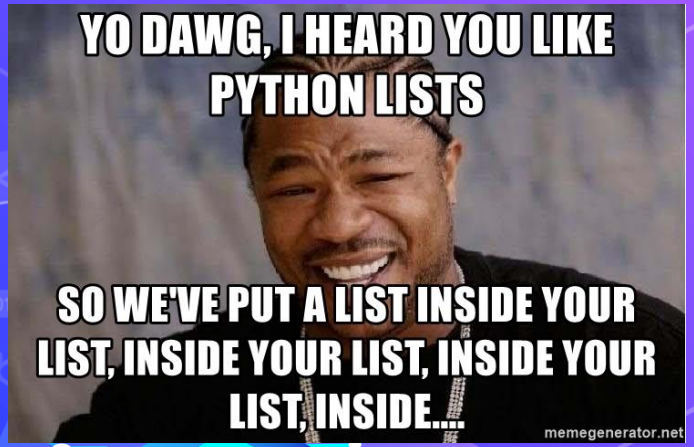
Unlike other programming languages, Python lists can have elements of different types

```python
my_list = ['A string', 23, 100.232 , 'p', True]

print(my_list[0])   # 'A string'
print(my_list[1])   # 23
print(my_list[2])   # 100.232
print(my_list[3])   # 'p'
print(my_list[4])   # True
```

Lists

List elements can be lists too!

```
my_list = [[1,2,3], [4,5,6], [7,[8,9]]]

print(my_list[0])        # [1,2,3]
print(my_list[0][1])     # 2
print(my_list[2][1][1]) # 9
```



Introduction to Python programming Course Outline

- Intro to Computer Science
- Environment Setup (Anaconda)
- Command Line
- Conda & pip package managers
- Jupyter Notebook
- Input & Output
- Variables
- Data types
- Numbers & Math
- Boolean & Comparison and Logic
- If Conditions
- For & While Loops
- Strings
- Lists
- **Tuples**
- Sets
- Dictionaries
- Advanced If and Loops
- List Comprehensions
- Dictionary Comprehensions
- Exceptions
- File Handling
- Functions
- Built-in functions & Operators (zip, enumerate, range, …)
- Map, Filter, Reduce
- Lambda Expressions

Tuples (faster and immutable lists)

Used when you have immutable values and need faster processing on them

```
my_tuple = ('A string', 23, 100.232 , 'p', True)

print(my_tuple[0])  # 'A string'
print(my_tuple[1])  # 23
print(my_tuple[2])  # 100.232
print(my_tuple[3])  # 'p'
print(my_tuple[4])  # True
```

Introduction to Python programming Course Outline

- Intro to Computer Science
- Environment Setup (Anaconda)
- Command Line
- Conda & pip package managers
- Jupyter Notebook
- Input & Output
- Variables
- Data types
- Numbers & Math
- Boolean & Comparison and Logic
- If Conditions
- For & While Loops
- Strings
- Lists
- Tuples
- **Sets**
- Dictionaries
- Advanced If and Loops
- List Comprehensions
- Dictionary Comprehensions
- Exceptions
- File Handling
- Functions
- Built-in functions & Operators (zip, enumerate, range, …)
- Map, Filter, Reduce
- Lambda Expressions

Sets (unique lists)

Used for intersections & union operations

```
my_list = [1,1,2,2,3,4,5,6,1,1]

my_set = set(my_list)
print(my_set)   # {1, 2, 3, 4, 5, 6}
```

Introduction to Python programming Course Outline

- Intro to Computer Science
- Environment Setup (Anaconda)
- Command Line
- Conda & pip package managers
- Jupyter Notebook
- Input & Output
- Variables
- Data types
- Numbers & Math
- Boolean & Comparison and Logic
- If Conditions
- For & While Loops
- Strings
- Lists
- Tuples
- Sets
- **Dictionaries**
- Advanced If and Loops
- List Comprehensions
- Dictionary Comprehensions
- Exceptions
- File Handling
- Functions
- Built-in functions & Operators (zip, enumerate, range, …)
- Map, Filter, Reduce
- Lambda Expressions

Dictionaries

Just like a human dictionary, Python dictionary are data structures that store data in key – value pairs

```python
store = {'apples': 10, 'oranges': 20}

print(store['apples'])  # result is 10
print(store['oranges']) # result is 20
```

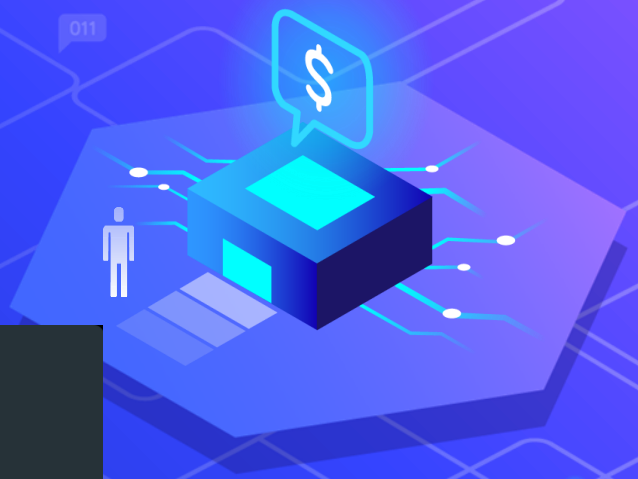
Dictionaries

Some of the useful dictionary functions:

dict.get('key') – looks for the key in the dictionary and returns value if found, returns default value if not found

dict.keys() – returns dictionary keys
dict.values() – returns dictionary values

```
store = {'apples':10, 'oranges':20}

print(store['grapes'])      # KeyError
print(store.get('grapes'))  # '' - default value
print(list(store.keys()))   # ['apples', 'oranges']
print(list(store.values())) # [10, 20]
```



More functions:
https://www.w3schools.com/python/python_ref_dictionary.asp

Quiz Time!

Q1. list1 = ['physics', 'chemistry', 1997, 2000]
print(list1[1][-1])

- A. p
- B. c
- C. y
- D. Error

Q2. list1 = [1, 2, 3, [1, 2], (1, 2, 3)]
print(len(list1))

- A. 8
- B. 5
- C. 6

Q3. list1 = [1998, 2002]
list2 = [2014, 2016]
print(list2 + list1)

- A. [4012, 4018]
- B. [2014, 2016, 1998, 2002]
- C. [1998, 2002, 2014, 2016]

Q4. name = “Data Science”
print(name[:4] + “Analysis”)

- A. “Data Analysis”
- B. “Data Snalysis”
- C. “DataAnalysis”

Practice #1

- Ask the user to enter five numbers then print the numbers that are divisible by 5.

Solution practice #1

- Ask the user to enter five numbers then add them to a list and print the numbers that are divisible by 5.

```
numbers = list()
for i in range(5):
    numbers.append(float(input("Enter your employee name: ")))
for num in numbers:
    if num % 5 ==0:
        print(num,"is divisble by 5")
```

Practice #2

- Reverse the following list using:
  - [10, 20,30,40,50]
  1. Built in function
  2. While loop
  3. For loop