Python_Overview_technical

May 3, 2024

Benson Nyota

1 Python Programming Language

Reach out if you need the jupyter file

1.0.1 Introduction

Python is an open source language which is very powerful but very easy to learn. It's elegant syntax and dynamic typing together with its interpreted nature makes it an ideal language to use.

[5]:
```
#print function
print("Hello World!!")
```

```
Hello World!!
```

[6]:
```
#Keywords in python
print(help('keywords'))
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not

None
```

[1]:
```
#Variables
x =  5
y = 8

#Expressions
```

```
x+y
x*y
y/x
5+8
```

[1]: 13

[7]:
```
#input function
input("Enter your name: ")
```

```
Enter your name:  Benson Nyota
```

[7]: 'Benson Nyota'

1.0.2 Modules, Classes and Packages(Libraries)

Module is a file containing a python code that can be reused severally through importing.

[12]:
```
#Modules
import os #operating system function
import sys #access to variables and functions
import math #mathematical functions
import random #pseudo-random numbers
import collections #counter -- additional data structures
import datetime
import urllib.request #fetching data from urls
```

[8]:
```
import os
print(os.getcwd())  # Prints the current working directory
```

```
C:\Users\bency\OneDrive\Desktop\Techsisters
```

Package/Library is a Collection of multiple modules

[ ]:
```
#Libraries
import numpy as np #numericals
import pandas as pd #dataframe
import matplotlib #visualization
import sklearn #machine learning
import tensorflow as tf #deep learning
import nltk #Natural Language Processing
import sqlalchemy #databases connection
```

[8]:
```
# Variable sizes
import sys
a = ['Love', 'Cold', 'Hot', 'Python']
b = {'Love', 'Cold', 'Hot', 'Python'}
c = ('Love', 'Cold', 'Hot', 'Python')
print(f'The memory size of a list is '
```

```
f'{sys.getsizeof(a)} ')
print(f'The memory size of a set is '
f'{sys.getsizeof(b)} ')
print(f'The memory size of a tuple is '
f'{sys.getsizeof(c)} ')
```

```
The memory size of a list is 88
The memory size of a set is 216
The memory size of a tuple is 72
```

1.1 Lists

[3]:
```
#sample of a list
#int
squares = [1, 4, 9, 16, 4, 25]
print(squares)
#strings
fruits = ['mango', 'banana', 'apple', 'kiwi']
print(fruits)
#floats
dimensions = [3.6, 6.87, 6.23]
print(dimensions)
```

```
[1, 4, 9, 16, 4, 25]
['mango', 'banana', 'apple', 'kiwi']
[3.6, 6.87, 6.23]
```

[5]:
```
fruits
```

[5]:
```
['mango', 'banana', 'apple', 'kiwi']
```

Indexing The aspect of retrieveing an elemnent in the list using it's index position

[4]:
```
#the first element
squares[0]
```

[4]:
```
1
```

[8]:
```
#the last element
fruits[-1]
```

[8]:
```
'kiwi'
```

[5]:
```
#the last element
dimensions[2]
```

[5]:
```
6.23
```

Append Adding new item(s) at the end of the list, by using the *list.append()*

[6]:
```
squares.append(897)
print(squares)
```

```
[1, 4, 9, 16, 4, 25, 897]
```

[7]:
```
fruits.append('pawpaw')
fruits
```

[7]:
```
['mango', 'banana', 'apple', 'kiwi', 'pawpaw']
```

Slicing getting a subset

[8]:
```
fruits[1:3]
```

[8]:
```
['banana', 'apple']
```

[9]:
```
#first two elements
fruits[:2]
```

[9]:
```
['mango', 'banana']
```

[10]:
```
#last two element
fruits[-2:]
```

[10]:
```
['kiwi', 'pawpaw']
```

[12]:
```
#finding the index of an element
fruits.index('apple')
```

[12]:
```
2
```

remove

[13]:
```
fruits.remove('mango')
fruits
```

[13]:
```
['banana', 'apple', 'kiwi', 'pawpaw']
```

Replace

[18]:
```
squares[-1] = 36
```

[19]:
```
squares
```

[19]:
```
[1, 4, 9, 16, 4, 25, 36]
```

Insert Insert an item at a given position. The first argument is the index of the element before which to insert, so *a.insert(0, x)* inserts at the front of the list, and *a.insert(len(a), x)* is equivalent to *a.append(x)*.

[20]:
```
squares.insert(2, 'nine')
squares
```

[20]:
```
[1, 4, 'nine', 9, 16, 4, 25, 36]
```

Pop Remove the item at the given position in the list, and return it. If no index is specified, *a.pop()* removes and returns the last item in the list. It raises an *IndexError* if the list is empty or the index is outside the list range.

[21]:
```
squares.pop()
```

[21]:
```
36
```

Deletion Deleting an element changes the indexies of elements in the list

[25]:
```
del squares[0]
squares
```

[25]:
```
[4, 'nine', 9, 16, 4, 25]
```

clear Remove all items from the list

[18]:
```
squares.clear()
squares
```

[18]:
```
[]
```

count return the number of times a specific element appears

[26]:
```
squares.count(4)
```

[26]:
```
2
```

Sort sort the items of the list in place

[30]:
```
squares.sort()
squares
```

[30]:
```
[4, 4, 9, 16, 25]
```

reverse

[31]:
```
squares.sort(reverse=True)
squares
```

[31]: [25, 16, 9, 4, 4]

concatenation

[32]:
```
fruits + squares
```

[32]: ['banana', 'apple', 'kiwi', 'pawpaw', 25, 16, 9, 4, 4]

Extend

[33]:
```
fruits.extend(['grapes','berries'])
fruits
```

[33]: ['banana', 'apple', 'kiwi', 'pawpaw', 'grapes', 'berries']

1.2 Tuples

They are immutable and are always enclosed by parentheses ().

They are the default sequence type in Python.

[34]:
```
# example of a tuple
t = (667, 78, 98, 23)

#calling out the tuple with variable name t
t
```

[34]: (667, 78, 98, 23)

Index

[35]:
```
#calling out element using their position indexes

t[0]
```

[35]: 667

More about Tuples

[36]:
```
#Tuples can be returned by a function
def square_info(x):
    A = x**2
    P = 4*x
    print("Area and Perimeter")
    return A,P

square_info(3)
```

```
Area and Perimeter
```

[36]: (9, 12)

[37]:
```
#Tuples can be used to extract info
name, age = 'Peter,24'.split(',')
print(name)
print(age)
```

```
Peter
24
```

1.3 Sets

A set is an unordered collection with no duplicate elements.

Set objects also support mathematical operations like union, intersection difference, and symmetric difference.

Objects are enclosed by parentheses

[39]:
```
#example
basket_1 = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
basket_1 # note that duplicates are removed
```

[39]:
```
{'apple', 'banana', 'orange', 'pear'}
```

membership testing

[40]:
```
'orange' in basket_1
```

[40]:
```
True
```

[41]:
```
'lemon' in basket_1
```

[41]:
```
False
```

Sequences

[43]:
```
#Sequence
basket_2 = ('apple', 'orange', 'apple', 'pear', 'orange', 'banana')
basket_2 #note: duplicates remain
```

[43]:
```
('apple', 'orange', 'apple', 'pear', 'orange', 'banana')
```

1.4 Dictionaries

Key:value pair

[44]:
```
# An example of a dictionary
animals = {'ki':'cat', 'k2':'dog', 'k3':'chicken', 'k4':'rabbit'}
animals
```

[44]:
```
{'ki': 'cat', 'k2': 'dog', 'k3': 'chicken', 'k4': 'rabbit'}
```

dict() constructor One can use dict() constructor method to create a dictionary

[46]:
```
company_1 = dict([('sales', 34), ('finance', 13), ['marketing', 56]])
company_1
```

[46]:
```
{'sales': 34, 'finance': 13, 'marketing': 56}
```

[47]:
```
# When the keys are simple strings, it is sometimes easier to specify pairs␣
↪using keyword arguments:
company_2 = dict(sales=34, finance=13, marketing=56)
company_2
```

[47]:
```
{'sales': 34, 'finance': 13, 'marketing': 56}
```

[48]:
```
# Use key to get the value
animals['k3']
```

[48]:
```
'chicken'
```

keys

[49]:
```
#listing the keys
animals.keys()
```

[49]:
```
dict_keys(['ki', 'k2', 'k3', 'k4'])
```

values

[50]:
```
#listing the values
animals.values()
```

[50]:
```
dict_values(['cat', 'dog', 'chicken', 'rabbit'])
```

Add an item

[51]:
```
animals['k5']='parrot'
```

[52]:
```
animals
```

[52]:
```
{'ki': 'cat', 'k2': 'dog', 'k3': 'chicken', 'k4': 'rabbit', 'k5': 'parrot'}
```

More than one value in a single key stored as lists

[56]:
```
workers = {'dept_1': 'James', 'dept_2':'Mary', 'dept_3':['Maureen', 'Jessica',␣
↪'John']}
```

[57]:
```
print(workers.get('dept_3'))
```

```
['Maureen', 'Jessica', 'John']
```

1.5 Control Flow Tools

If Statement

[62]:
```
# create a variable
x = 9
```

[63]:
```
if x>5:
    print (x*5)
```

```
45
```

If Statement, Elif, Else

[66]:
```
x = int(input("Please Enter an Integer: "))
if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

```
Please Enter an Integer:  9
```

```
More
```

1.5.1 If statement nested in a For loop

[75]:
```
for i in range(20):
    if i%2==0:
        print(str(i)," Even", end=" ")
    else:
        print(str(i), "Odd")
```

```
0  Even 1 Odd
2  Even 3 Odd
4  Even 5 Odd
6  Even 7 Odd
8  Even 9 Odd
10  Even 11 Odd
12  Even 13 Odd
14  Even 15 Odd
16  Even 17 Odd
18  Even 19 Odd
```

1.5.2 While loop

[73]:
```
index = 0
while index<5:
    print("index", str(index))
    index = index + 1
```

```
index 0
index 1
index 2
index 3
index 4
```

[ ]:

[ ]:

[ ]:

[ ]: