**Introduction**

Python is an interpreted, object-oriented, high-level programming language. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

Often, programmers fall in love with Python because of the increased productivity it provides. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

**Applications developed using Python**

YouTube.

DropBox.

Google.

Quora.

Instagram.

**Creating Variables**

Unlike other programming languages, Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Example

x = 5

print(x)

Output: 5

y = "John"

print(y)

Output:”John”

# **Python Strings**

Strings in python are surrounded by either single quotation marks, or double quotation marks.

Strings can be output to screen using the print function.

For example: print("hello").

Output:”hello”

Square brackets can be used to access elements of the string.

Example: Get the character at position 1 (remember that the first character has the position 0):

a = "Hello, World!"

print(a[1])

Output:”e”

**Substring(Slicing string)** - Get the characters from position 2 to position 5 (not included):

Example

b = "Hello, World!"

print(b[2:5])

Output:”llo”

The **len()** method returns the length of a string:

Example

a = "Hello, World!"

print(len(a))

Output:13

The **split()** method splits the string into substrings if it finds instances of the separator:

Example

a = "Hello, World!"

print(a.split(","))

Output:['Hello', ' World!']

The **strip()** method removes any whitespace from the beginning or the end:

Example

a = " Hello, World! "

print(a.strip())  
Output:"Hello, World!"

The **replace()** method replaces a string with another string:

Example

a = "Hello, World!"

print(a.replace("H", "J"))

Output:”Jello, World!”

Example:

a = "Hello, World!”

a.replace(",","")

Output:”Hello World!”

**Data Types in Python**

There are 4 types of data types in python namely Lists, Dictionaries, Tuples & Sets.

1. **Lists -** List is a collection which is ordered and changeable. Allows duplicate members.
2. **Dictionaries -** Dictionary is a collection which is unordered, changeable and indexed. No duplicate members.
3. **Tuples -** Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
4. **Sets -** Set is a collection which is unordered and unindexed. No duplicate members.

## **List**

A list is a collection which is ordered and changeable. In Python lists are written with square brackets. List is a mutable data type. I.e. value of the list can be altered or modified.

Example  
thislist = ["apple", "banana", "cherry"]  
print(thislist)

Output:['apple', 'banana', 'cherry']

**Access Items of the list**

You access the list items by referring to the index number:

Example

Print the second item of the list:

thislist = ["apple", "banana", "cherry"]

print(thislist[1])

Output:banana

**Change Item Value**

To change the value of a specific item, refer to the index number:

Example

Change the second item:

thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist)

Output:['apple', 'blackcurrant', 'cherry']

**Loop Through a List**

You can loop through the list items by using a for loop:

Example

Print all items in the list, one by one:

thislist = ["apple", "banana", "cherry"]

for x in thislist:

print(x)

Output:

apple

banana

cherry

**Check if Item Exists**

To determine if a specified item is present in a list use the **in** keyword:

Example

Check if "apple" is present in the list:

thislist = ["apple", "banana", "cherry"]

if "apple" in thislist:

print("Yes, 'apple' is in the fruits list")

Output:Yes, 'apple' is in the fruits list

**List Length**

To determine how many items a list has, use the **len()** method:

Example

Print the number of items in the list:

thislist = ["apple", "banana", "cherry"]

print(len(thislist))

Output:3

**Add Items**

To add an item to the end of the list, use the **append()** method:

Example

Using the append() method to append an item:

thislist = ["apple", "banana", "cherry"]

thislist.append("orange")

print(thislist)

Output:['apple', 'banana', 'cherry', 'orange']

**Remove Item**

Example

The **del** keyword removes the specified index:

thislist = ["apple", "banana", "cherry"]

del thislist[0]

print(thislist)

Output:['banana', 'cherry']

Example

The **del** keyword can also delete the list completely:

thislist = ["apple", "banana", "cherry"]

del thislist

Output:Traceback (most recent call last):

File "demo\_list\_del2.py", line 3, in <module>

print(thislist) #this will cause an error because you have succsesfully deleted "thislist".

NameError: name 'thislist' is not defined

Example

The **del list[:]** empties the list:

thislist = ["apple", "banana", "cherry"]

del thislist [:]

print(thislist)

Output:[]

**Dictionary**

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.List is a mutable data type. I.e. value of the dictionary can be altered or modified.

Example  
Create and print a dictionary:  
thisdict = {  
 "brand": "Ford",  
 "model": "Mustang",  
 "year": 1964  
}  
print(thisdict)

Output:{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

**Accessing Items**

You can access the items of a dictionary by referring to its key name, inside square brackets:

Example  
Get the value of the "model" key:

x = thisdict["model"]

Output:Mustang

**Change Values**

You can change the value of a specific item by referring to its key name:

Example

Change the "year" to 2018:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict["year"] = 2018

Output:{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}

**Loop Through a Dictionary**

You can loop through a dictionary by using a for loop.

When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.

Example

Print all **key names** in the dictionary, one by one:

for x in thisdict:

print(x)

Output:

brand

model

year

Example

Print all **values** in the dictionary, one by one:

for x in thisdict:

print(thisdict[x])  
Output:

Ford

Mustang

1964

**Dictionary Length**

To determine how many items (key-value pairs) a dictionary has, use the **len()** method.

Example

Print the number of items in the dictionary:

print(len(thisdict))

Output:3

**Adding Items**

Adding an item to the dictionary is done by using a new index key and assigning a value to it:

Example

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict["color"] = "red"

print(thisdict)

Output:{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}

**Removing Items**

Example

The **del** keyword removes the item with the specified key name:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

del thisdict["model"]

print(thisdict)

Output:{'brand': 'Ford', 'year': 1964}

Example

The **del** keyword can also delete the dictionary completely:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

del thisdict

print(thisdict) #this will cause an error because "thisdict" no longer exists.

Output:Traceback (most recent call last):

File "demo\_dictionary\_del3.py", line 7, in <module>

print(thisdict) #this will cause an error because "thisdict" no longer exists.

NameError: name 'thisdict' is not defined

Example

The **clear()** keyword empties the dictionary:

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict.clear()

print(thisdict)

Output:{}

**Tuple**

A tuple is a collection which is ordered and unchangeable. In Python tuples are written with round brackets. Tuple is a immutable data type i.e. value of the tupple cannot be changed or modified

Example

Create a Tuple:

thistuple = ("apple", "banana", "cherry")

print(thistuple)

Output:('apple', 'banana', 'cherry')

**Access Tuple Items**

You can access tuple items by referring to the index number, inside square brackets:

Example

Return the item in position 1:

thistuple = ("apple", "banana", "cherry")

print(thistuple[1])

Output:banana

**Change Tuple Values**

Once a tuple is created, you cannot change its values. Tuples are unchangeable.

Example

You cannot change values in a tuple:

thistuple = ("apple", "banana", "cherry")

thistuple[1] = "blackcurrant"

# The values will remain the same:

print(thistuple)

Output:('apple', 'banana', 'cherry')

**Loop Through a Tuple**

You can loop through the tuple items by using a for loop.

Example

Iterate through the items and print the values:

thistuple = ("apple", "banana", "cherry")

for x in thistuple:

print(x)

Output:

apple

banana

cherry

**Tuple Length**

To determine how many items a tuple has, use the **len()** method:

Example

Print the number of items in the tuple:

thistuple = ("apple", "banana", "cherry")

print(len(thistuple))

Output:3

**Add Items**

Once a tuple is created, you cannot add items to it. Tuples are unchangeable.

Example

You cannot add items to a tuple:

thistuple = ("apple", "banana", "cherry")

thistuple[3] = "orange" # This will raise an error

print(thistuple)

Output:Traceback (most recent call last):

File "demo\_tuple\_add.py", line 2, in <module>

thistuple[3] = "orange" # This will raise an error

TypeError: 'tuple' object does not support item assignment

**Remove Items**

Note: You cannot remove items in a tuple.

Tuples are unchangeable, so you cannot remove items from it, but you can delete the tuple completely:

Example

The del keyword can delete the tuple completely:

thistuple = ("apple", "banana", "cherry")

del thistuple

print(thistuple) #this will raise an error because the tuple no longer exists

Output:Traceback (most recent call last):

File "demo\_tuple\_del.py", line 3, in <module>

print(thistuple) #this will raise an error because the tuple no longer exists

NameError: name 'thistuple' is not defined

**Set**

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

Example

Create a Set:

thisset = {"apple", "banana", "cherry"}

print(thisset)

Note: Sets are unordered, so the items will appear in a random order.

Output:{'banana', 'cherry', 'apple'}

**Access Items**

You cannot access items in a set 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.

Example

Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}

for x in thisset: print(x)

Output:

banana

cherry

apple

**Change Items**

Once a set is created, you cannot change its items, but you can add new items.

**Add Items**

To add one item to a set use the **add()** method.

To add more than one item to a set use the **update()** method.

Example

Add an item to a set, using the add() method:

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)

Output:{'apple', 'banana', 'orange', 'cherry'}

Example

Add multiple items to a set, using the **update()** method:

thisset = {"apple", "banana", "cherry"}

thisset.update(["orange", "mango", "grapes"])

print(thisset)

Output:{'banana', 'grapes', 'cherry', 'mango', 'orange', 'apple'}

**Get the Length of a Set**

To determine how many items a set has, use the **len()** method.

Example

Get the number of items in a set:

thisset = {"apple", "banana", "cherry"}

print(len(thisset))

Output:3

**Remove Item**

To remove an item in a set, use the **remove()**, or the **discard()** method.

Example

Remove "banana" by using the remove() method:

thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)

Note: If the item to remove does not exist, remove() will raise an error.

Output:{'apple', 'cherry'}

Example

The clear() method empties the set:

thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset)

Output:set()

Example

The del keyword will delete the set completely:

thisset = {"apple", "banana", "cherry"}

del thisset

print(thisset)

Output:Traceback (most recent call last):

File "demo\_set\_del.py", line 5, in <module>

print(thisset)

Output:<type 'set'>