Introduction To Python

Satya Prakash Modi
Software Developer, IBM

Presenter Intro

Hailing from the city of Bhagalpur Bihar, Satya Prakash Modi is working as a Software Developer at IBM USA. Before moving to United States for his Master's, he worked in India for several years on technologies related to Oracle Database. After completing his Master's in the field related to Cloud Computing, he is currently working on cloud technologies such as Amazon Web Services (AWS), Microsoft Azure and Google Cloud.

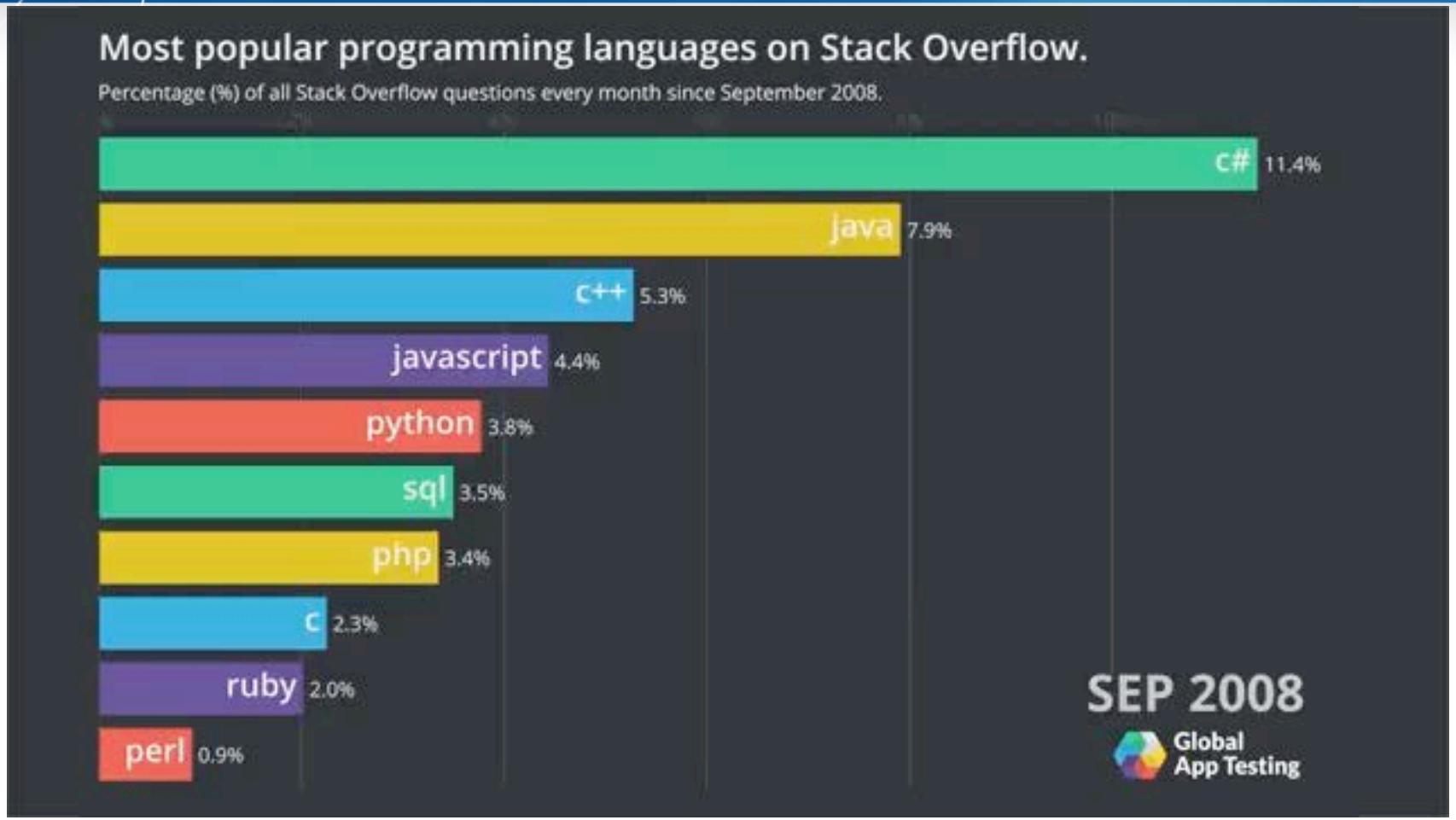
Introduction

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- It was developed by Guido van Rossum in the year 1989.
- Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Features

- Python is a free and open source software.
- Python code is processed at runtime by the interpreter. You do not need to compile your program before executing it.
- Dynamically typed language. It doesn't know about the type of the variable until the code is run.
- Python is a cross-platform language. It can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc.
- Python has a large and broad library and provides rich set of module and functions for rapid application development.

Journey of Python



Variables

Variables are not statically typed in Python.

Python

```
x = 5
y = x + 7
z = 3.14

name = “Andi”
```

Java/C++

```
int x = 5;
int y = x + 7;
double z = 3.14;

String name = “Rishi”; // Java
string name(“Rishi”); // C++
```

Other Objects

- Lists (mutable sets of strings)
  - `var = []  # create list`
  - `var = ['one', 2, 'three', 'banana']`
- Tuples (immutable sets)
  - `var = ('one', 2, 'three', 'banana')`
- Dictionaries (associative arrays or 'hashes')
  - `var = {}  # create dictionary`
  - `var = {'lat': 40.20547, 'lon': -74.76322}`
  - `var['lat'] = 40.2054`

Lists

```
# Create a new list
empty = []
letters = ['a', 'b', 'c', 'd']
numbers = [2, 3, 5]

# Lists can contain elements of different types
mixed = [4, 5, "seconds"]

# Append elements to the end of a list
numbers.append(7)     # numbers == [2, 3, 5, 7]
numbers.append(11)    # numbers == [2, 3, 5, 7, 11]
```

Lists

```
# Access elements at a particular index
numbers[0]          # => 2
numbers[-1]         # => 11

# You can also slice lists - the same rules apply
letters[:3]         # => ['a', 'b', 'c']
numbers[1:-1]       # => [3, 5, 7]

# Lists really can contain anything - even other lists!
x = [letters, numbers]
x # => [['a', 'b', 'c', 'd'], [2, 3, 5, 7, 11]]
x[0]          # => ['a', 'b', 'c', 'd']
x[0][1]       # => 'b'
x[1][2:]      # => [5, 7, 11]
```

Tuples

- A tuple is a sequence of immutable Python objects.
- Tuples are sequences, just like lists.
- The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

```
t = (12345, 54321, 'hello!’)
t[0]                                #12345

my_tup[0] = 5 => Error!
```

Dictionaries

- Keys are unique within a dictionary while values may not be.
- The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

```
d = {"one": 1, "two": 2, "three": 3}
len(d.keys())       # => 3
Print(d['one’])     #=> 1
Print(d['five’])    # => ERROR!
d['five'] = 5       #=> OK, creates new key
d.keys()            # iterator over k
d.values()          # iterator over v
d.items()           # iterator over (k, v) pairs
```

Iteration

Most python objects can be iterated over in the same way:

```
mylist = ['a', 'b', 'c']
for item in mylist:
        print(item)

#we don't need the index of the element to access it.
mytuple = ('a', 'b', 'c')
for item in mytuple:
        print(item)

#When iterating over a dictionary like this, we iterate
over the keys
dict = {'a': 10, 'b': 15}
for key in dict:
        print(key, dict[key])
```

Iteration

We can also iterate over indices:

```
for i in range(4):
      print(i)                                  #0123

for i in range(1, 10, 2):
      print (i)                                 #13579

mylist = ['a', 'b', 'c’]                        #0'a'
for i in range(len(mylist):                     #1'b'
      print(i, mylist[i])                       #2'c'

mylist = ['a', 'b', 'c’]                        #0'a'
for idx, item in enumerate(mylist):             #1'b'
      print(idx, item)                          #2'c'
```

List Comprehensions

Input: `nums = [1, 2, 3, 4, 5]`

Goal: `sq_nums = [1, 4, 9, 16, 25]`

Here's how we could already do this:

```
sq_nums = []
for n in nums:
        sq_nums.append(n**2)
```

Or... we could use a comprehension:

```
#square brackets show we're making a list
#apply some operation to the loop variable
#loop over the specified iterable
sq_nums = [n ** 2 for n in nums]
```

List Comprehensions

Template

```
new_list = [f(x) for x in iterable]
```

```
words = ['hello', 'this', 'is', 'python']
caps = [word.upper() for word in words]

powers = [(x**2, x**3, x**4) for x in range(10)]
```

Functions

- Function blocks begin with the keyword def.
- return is optional, The statement return [expression] exits a function, optionally passing back an expression to the caller.
- Parameters have no explicit types.

```
def fn_name(param1, param2):
        value = do_something()
        return value

def isEven(num):
        return (num % 2 == 0)

myNum = 100
if isEven(myNum):
        print(str(myNum) + " is even")
```

Modules

- A module is a file containing Python definitions and statements.
- A module can be imported into other modules or into the main module.

```
# Fibonacci numbers module
def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()


>>> import fibo
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```