Sample rst2pdf doc

version

prashant

April 12, 2020

Contents

Welcome to tectTutorial’s documentation!

What is Python really?

- Python is an **interpreted** language. That means that, unlike languages like C and its variants, Python does not need to be compiled before it is run. Other interpreted languages include PHP and Ruby.

- Python is **dynamically** typed, this means that you don’t need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x=”I’m a string” without error

- Python is well suited to **object orientated programming** in that it allows **the definition of classes along with composition and inheritance**. Python does not have access specifiers (like C++’s public, private), the justification for this point is given as “we are all adults here“

- in Python, **functions are first-class objects**. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects

- Writing Python **code is quick** but running it is often *slower* than compiled languages. Fortunately■ Python allows the inclusion of C based extensions so bottlenecks can be optimized away and often are. The numpy package is a good example of this, it’s really quite quick because a lot of the number crunching it does isn’t actually done by Python

- Python finds use in many spheres - **web applications, automation, scientific modeling, big data** applications and many more. It’s also often used as “glue” code to get other languages and components to play nice

How python code is executed

At the command line:

```
a = "hello“
```

There are four steps that python takes when you hit return lexing, parsing, compiling, and interpreting

1. **Lexing** is breaking the line of code you just typed into **tokens** | 2. The **parser** takes those **tokens** and generates a structure that shows their relationship to each other (in this case, an **Abstract SyntaxTree**). | 3. The **compiler** then takes the **AST** and turns it into one (or more) **code objects**. (function objects, code objects, and bytecode) | 4. Finally, the **interpreter** takes each code object (It contains information that this interpreter needs to do its job) executes the code it represents

to understand how Lexing,prasar & AST work a link.

variables and types

- Python is completely object oriented
- No need to declare variable before using them or declare their type
- **Every variable** in Python is an **Object**

Strings

String constants

are delimited with " " (double quotes) or ' ' (single quotes)

```
>> h = "Hello World"
>> h = 'Hello World'
```

Multiline string

String constant can contain new line by using double quotes or single quotes:

```
>> h = """ Hi
    How are you"""
```

String Concatenation

Use + sign to concate 2 strings:

```
>> s = "hi"
>> t =  "How are you"
>> print(s + t)
Hi How are you
```

Strings are Immutable

String cannot be modified like in programming language (like c):

```
>> h = "Hello world"
>> h[2] = 'g'
TypeError: 'str' object does not support item assignment.
```

String Indexing and Splicing

String can be indexed like in C:

```
>> h = "hello world"
>> h[0]     # note index starts from zero
H
```

String can be sliced:

```
>>h[2:5]
llo
>>h[2:]
llo world
>>h[:2]
he
```

String Interpolation

The mod(%) operator in string is used for formatting or to insert variable in a string:

```
>> print "There are %d orange in the basket" % 32
There are 32 orange in the basket

>> print "There are %d %s and %d %s in the basket" % (32, 'orange', 12, 'apple')
There are 32 orange and 12 apple in the basket
```

Python Strings - Useful Functions

```
S = “Jack and Jill”
```

capitalize()

Capitalizes first letter of the string:

```
>> s.capitalize()
>> Jack And Jill
```

count(str, beg=0, end=len(string))

Count how many times str occurs in a string or in a substring (if beg and end is given):

```
>> s.count(‘J’)
2
```

find(str, beg=0, end=len(string))

Determine if str occurs in string or in a substring of string if starting index *beg* and ending index *end* are given return **index** if found else **-1**

```
>> s.find(‘and’)
>> 5
>> s.find(‘j’)
>> -1
```

- **Numbers**
  - Python Support 2 type of numbers Integer and Floating point number
  - To define an Integer use x = 7
  - To define a floating point number use x = 7.0
- **Object Types**
  - Python automatically assign object type based on the assignment
  - For example

```
>> x = 7
>> type(x)
<type 'int'>

>> x = 'abc'
>> type(x)
<type 'str'>

>> x = 7.7
>> type(x)
<type 'float'>
```

Indices and tables

- **genindex**
- **modindex**
- **search**