

# Manipulating Strings in Python

---



**Dr. Chris Brown**



# In This Lesson

String Literals

String Operators

String Formatting

String  
Manipulation and  
Testing

Representing  
Dates and Times

Using Regular  
Expressions



A Python string is an  
immutable sequence of  
Unicode characters



# String Literals

---



# String Tests

---



# String Operators

---



# String Formatting

---



# String Manipulation

---





# String Tests

---



# String Test Examples

```
while True:
    age = input("Enter your age: ")
    if age.isdecimal():
        break
    print("age should be an integer, try again!")

print(age)
```

## Other tests:

<code>isalpha()</code>	True if all characters are alphabetic
<code>isalnum()</code>	True if all characters are alphanumeric
<code>islower()</code>	True if all characters are lower case
<code>isupper()</code>	True if all characters are upper case
<code>isspace()</code>	True if all characters are whitespace



# Dates and Times

---





Proleptic  
Gregorian  
Calendar

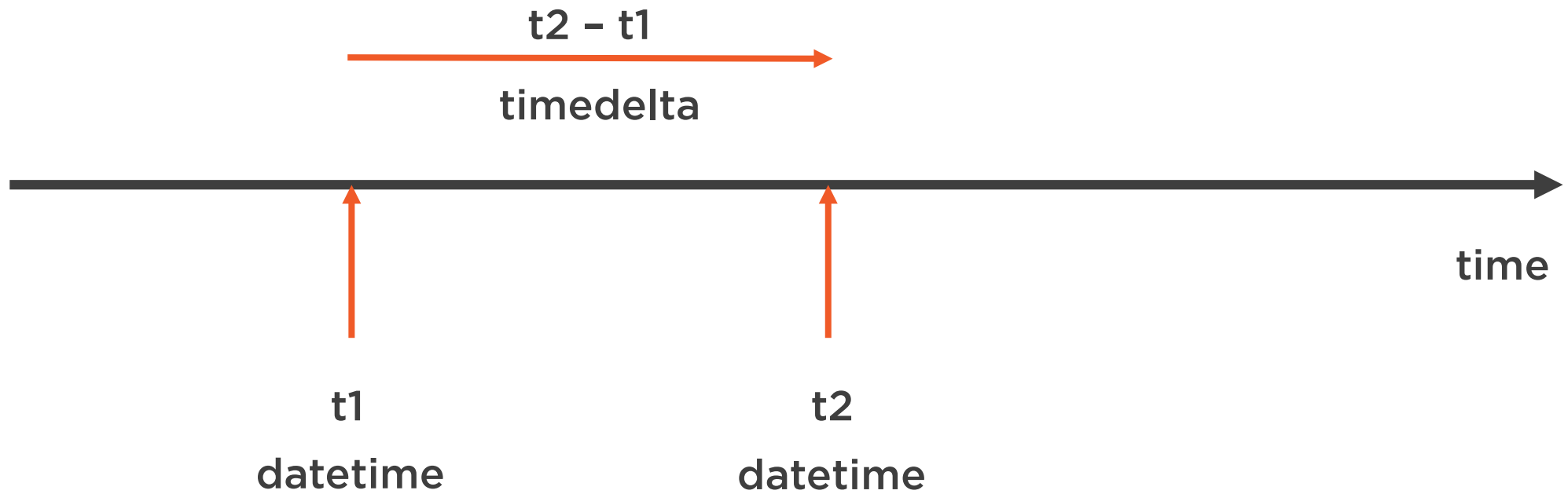
Leap Years

Time  
zones

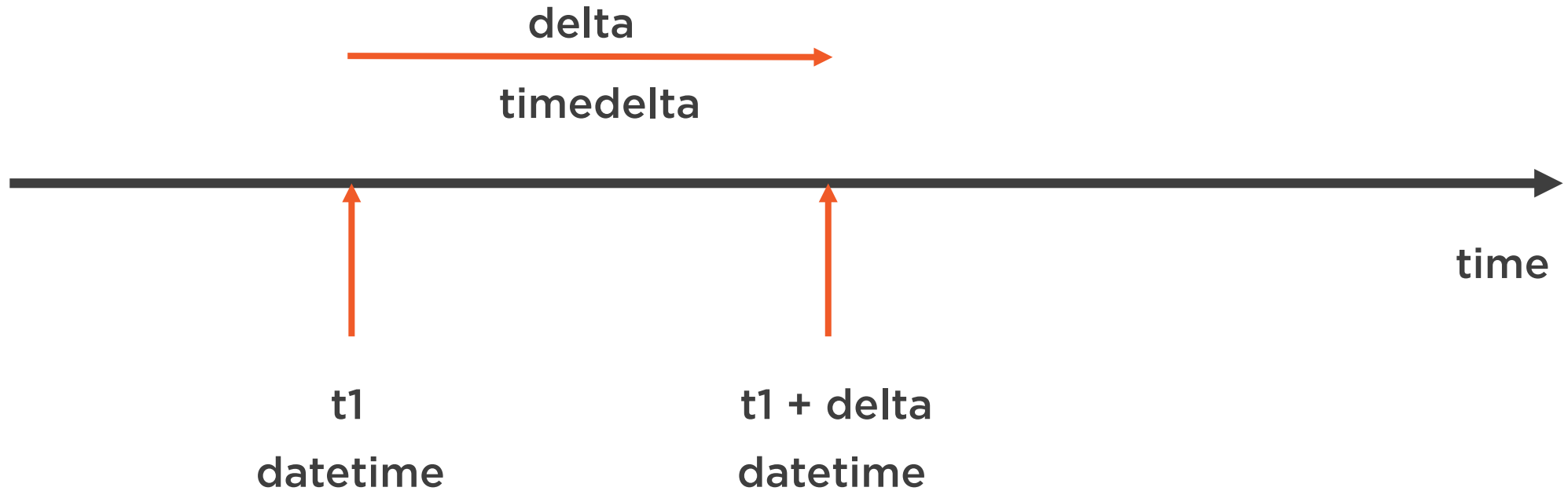
Daylight  
Savings  
Time



# datetime and timedelta



# datetime and timedelta



# Regular Expressions

---



# Introducing Regular Expressions

Regular expressions provide a notation for matching patterns in text

Input Validation

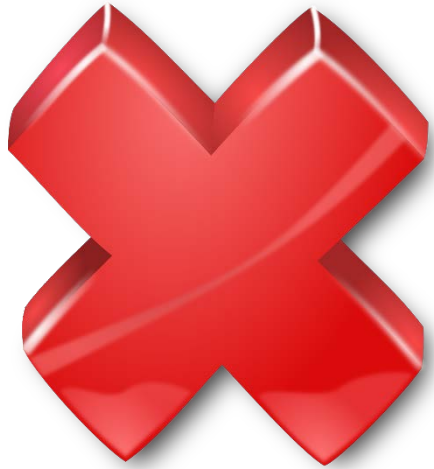
Searching

Text substitution





# What We Will and Won't Be Doing



**The details of regular  
expression syntax**



**Some examples of  
using Python's "re"  
module**

# Date Re-Writing

MM-DD-YYYY → DD-MM-YYYY

01-20-2017 → 20-01-2017



# Dissecting a Date

01-20-2017

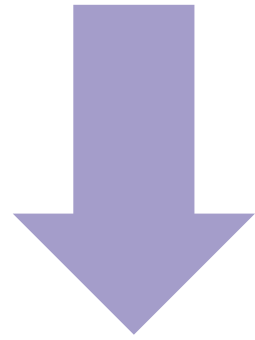
`r"(\d\d)-(\d\d)-(\d{4})"`

Group 1      Group 2      Group 3



# Changing the Domain of a URL

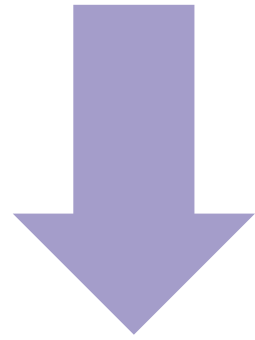
`http://mail.oldplace.com/mailbox?id=5432`



`http://mail.newplace.org/mailbox?id=5432`

# Changing the Domain of a URL

`https://docs.oldplace.com/chris/report`



`https://docs.newplace.org/chris/report`

# Matching Parts of a URL

`http://mail.oldplace.com/mailbox?id=5432`

`(https?)://(\w+)\.oldplace\.com/(\S+)`

`https://docs.oldplace.com/chris/report`



# Going Further with Regular Expressions

**"Regular  
Expression  
Fundamentals"**

**(Pluralsight)**

**"Mastering  
Regular  
Expressions"**

**by**

**Jeffrey Friedl**



**The re module  
documentation at  
[docs.python.org](https://docs.python.org)**



# Improved Regex Module at [pypi.python.org](https://pypi.python.org)

## Index of Packages Matching 'regex'

**Not Logged In**

[Login](#)  
[Register](#)  
[Lost Login?](#)  
Use [OpenID](#)   
[Login with Google](#) 

**Status**

[Nothing to report](#)

Package	Weight*	Description
<a href="#">django-regex-field 0.2.0</a>	9	Django Regex Field
<a href="#">flake8-regex 0.3</a>	9	Arbitrary regex checker, extension for flake8
<a href="#">regex 2017.02.08</a>	9	Alternative regular expression module, to replace re.
<a href="#">collective.regexredirector 0.2.3</a>	7	Addon for plone.app.redirector concerning regex redirector
<a href="#">commonregex 1.5.4</a>	7	Find all dates, times, emails, phone numbers, links, emails, ip addresses, prices, bitcoin address, and street addresses in a string.





# Summary



String literals and "raw strings"

String operators: + and \*

Building formatted strings

- The % operator and % format codes

- The format() method, and {}

String manipulation and tests

Representing dates and times

Regular expressions



# In the Next Lesson

