Fecha y hora (datetime)

Fecha y hora

The **datetime** module provides a number of types to deal with dates, times, and time intervals.

- The **datetime** type represents a date and a time during that day.
- The **date** type represents just a date, between year 1 and 9999 (see below for more about the calendar used by the datetime module)
- The **time** type represents a time, independent of the date.
- The **timedelta** type represents the difference between two time or date objects.
- The **tzinfo** type is used to implement timezone support for time and datetime objects; more about that below.

Fecha y hora

```
>>> import datetime
>>> now = datetime.datetime(2003, 8, 4, 12, 30, 45)

>>> print now
2003-08-04 12:30:45

>>> print repr(now)
datetime.datetime(2003, 8, 4, 12, 30, 45)

>>> print type(now)
<type 'datetime.datetime'>

>>> print now.year, now.month, now.day
2003 8 4

>>> print now.hour, now.minute, now.second
12 30 45
```

Fecha y hora

```
>>> import datetime
>>> import time
>>> print datetime.datetime(2003, 8, 4, 21, 41, 43)
2003-08-04 21:41:43
>>> print datetime.datetime.today()
2003-08-04 21:41:43.522000
>>> print datetime.datetime.now()
2003-08-04 21:41:43.522000
>>> print datetime.datetime.fromtimestamp(time.time())
2003-08-04 21:41:43.522000
>>> print datetime.datetime.utcnow()
2003-08-04 19:41:43.532000
>>> print datetime.datetime.utcfromtimestamp(time.time())
2003-08-04 19:41:43.532000
```

Fecha y hora

```
>>> import datetime
>>> import time
>>> now = datetime.datetime.now()
>>> print now
2003-08-05 21:36:11.590000
>>> print now.ctime()
Tue Aug  5 21:36:11 2003
>>> print now.isoformat()
2003-08-05T21:36:11.590000
>>> print now.strftime("%Y%m%dT%H%M%S")
20030805T213611
```

Fecha y hora

```
import datetime

d = datetime.date(2003, 7, 29)

print d
print d.year, d.month, d.day

print datetime.date.today()

$ python datetime-example-4.py
2003-07-29
2003 7 29
2003-08-07
```

Fecha y hora

```
>>> import datetime

>>> d = datetime.date(2003, 7, 29)

>>> print d
2003-07-29
>>> print d.year, d.month, d.day
2003 7 29
>>> print datetime.date.today()
2003-08-07
```

Fecha y hora

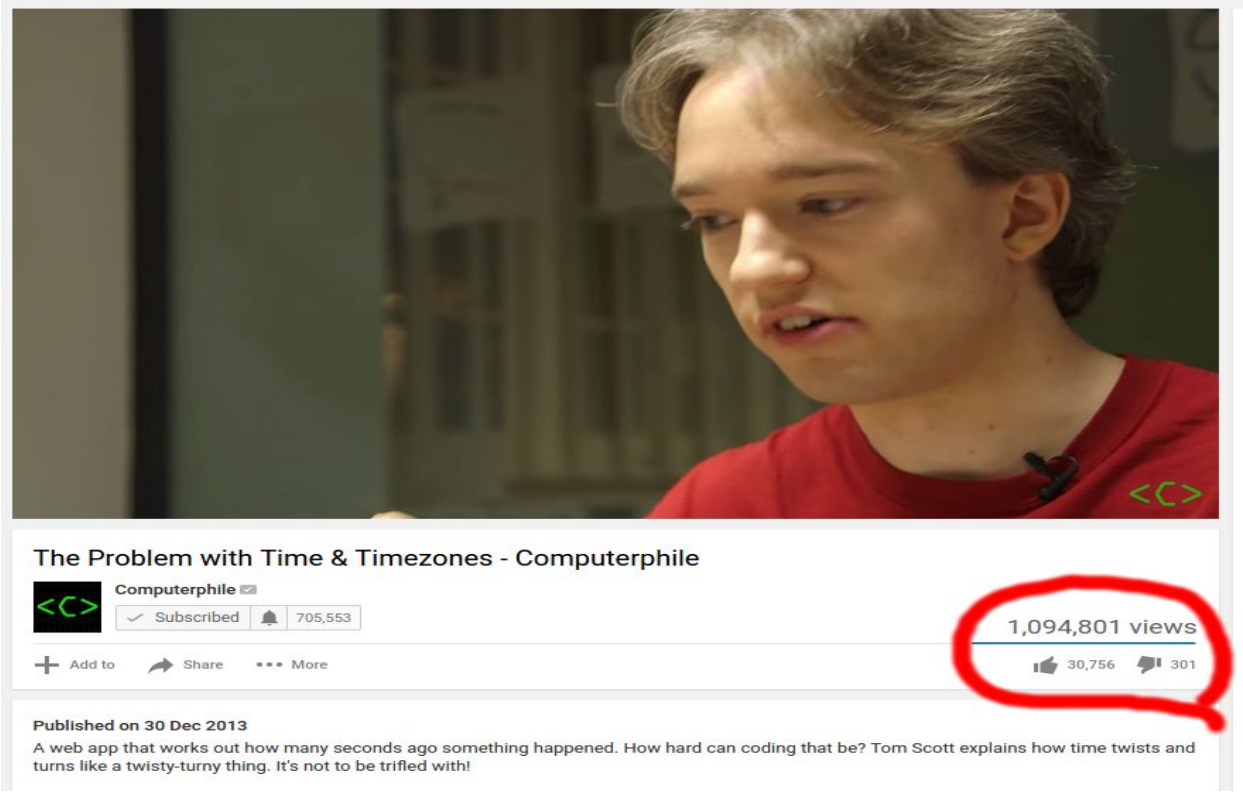
```
>>> import datetime

>>> d = datetime.date(2003, 7, 29)

>>> print d
2003-07-29
>>> print d.year, d.month, d.day
2003 7 29
>>> print datetime.date.today()
2003-08-07
```

Fecha y hora

```
>>> import datetime
>>> t = datetime.time(18, 54, 32)
>>> print t
18:54:32
>>> print t.hour, t.minute, t.second, t.microsecond
18 54 32 0
```

Fecha y hora - Porque es tan útil esta librería

https://www.youtube.com/watch?v=-5wpm-gesOY



Expresiones Regulares (`re`)

Expresiones Regulares

Regular expressions are a powerful and standardized way of searching, replacing, and parsing text with complex patterns of characters.

Expresiones Regulares

```
>>> s = '100 NORTH MAIN ROAD'
>>> s.replace('ROAD', 'RD.')
'100 NORTH MAIN RD.'
>>> s = '100 NORTH BROAD ROAD'
>>> s.replace('ROAD', 'RD.')
'100 NORTH BRD. RD.'
>>> s[:-4] + s[-4:].replace('ROAD', 'RD.')
'100 NORTH BROAD RD.'
>>> import re
>>> re.sub('ROAD$', 'RD.', s)
'100 NORTH BROAD RD.'
```

Expresiones Regulares

```
>>> s = '100 BROAD'
>>> re.sub('ROAD$', 'RD.', s)
'100 BRD.'
>>> re.sub('\\bROAD$', 'RD.', s)
'100 BROAD'
>>> re.sub(r'\bROAD$', 'RD.', s)
'100 BROAD'
>>> s = '100 BROAD ROAD APT. 3'
>>> re.sub(r'\bROAD$', 'RD.', s)
'100 BROAD ROAD APT. 3'
>>> re.sub(r'\bROAD\b', 'RD.', s)
'100 BROAD RD. APT 3'
```

Expresiones Regulares

Números Romanos

I = 1

V = 5

X = 10

L = 50

C = 100

D = 500

M = 1000

Expresiones Regulares

Números Romanos

- Characters are additive. I is 1, II is 2, and III is 3. VI is 6 (literally, “5 and 1”), VII is 7, and VIII is 8.
- The tens characters (I, X, C, and M) can be repeated up to three times. At 4, you need to subtract from the next highest fives character. You can't represent 4 as IIII; instead, it is represented as IV (“1 less than 5”). The number 40 is written as XL (10 less than 50), 41 asXLI, 42 as XLII, 43 as XLIII, and then 44 as XLIV (10 less than 50, then 1 less than 5).
- Similarly, at 9, you need to subtract from the next highest tens character: 8 is VIII, but 9 is IX (1 less than 10), not VIIII (since the Icharacter can not be repeated four times). The number 90 is XC, 900 is CM.
- The fives characters can not be repeated. The number 10 is always represented as X, never as VV. The number 100 is always C, neverLL.
- Roman numerals are always written highest to lowest, and read left to right, so the order the of characters matters very much. DC is600; CD is a completely different number (400, 100 less than 500). CI is 101; IC is not even a valid Roman numeral (because you can't subtract 1 directly from 100; you would need to write it as XCIX, for 10 less than 100, then 1 less than 10).

Expresiones Regulares

```
>>> import re
>>> pattern = '^M?M?M?$'
>>> re.search(pattern, 'M')
<SRE_Match object at 0106FB58>
>>> re.search(pattern, 'MM')
<SRE_Match object at 0106C290>
>>> re.search(pattern, 'MMM')
<SRE_Match object at 0106AA38>
>>> re.search(pattern, 'MMMM')
>>> re.search(pattern, '')
<SRE_Match object at 0106F4A8>
```

Expresiones Regulares

```
>>> import re
>>> pattern = '^M?M?M?$'
>>> re.search(pattern, 'M')
<_sre.SRE_Match object at 0x008EE090>
>>> pattern = '^M?M?M?$'
>>> re.search(pattern, 'MM')
<_sre.SRE_Match object at 0x008EEB48>
>>> pattern = '^M?M?M?$'
>>> re.search(pattern, 'MMM')
<_sre.SRE_Match object at 0x008EE090>
>>> re.search(pattern, 'MMMM')
>>>
```

Expresiones Regulares

```
>>> pattern = '^M{0,3}$'
>>> re.search(pattern, 'M')
<_sre.SRE_Match object at 0x008EEB48>
>>> re.search(pattern, 'MM')
<_sre.SRE_Match object at 0x008EE090>
>>> re.search(pattern, 'MMM')
<_sre.SRE_Match object at 0x008EEDA8>
>>> re.search(pattern, 'MMMM')
>>>
```

Expresiones Regulares

```
>>> pattern = '^M?M?M?(CM|CD|D?C?C?C?)(XC|XL|L?X?X?X?)$'
>>> re.search(pattern, 'MCMXL')
<_sre.SRE_Match object at 0x008EEB48>
>>> re.search(pattern, 'MCML')
<_sre.SRE_Match object at 0x008EEB48>
>>> re.search(pattern, 'MCMLX')
<_sre.SRE_Match object at 0x008EEB48>
>>> re.search(pattern, 'MCMLXXX')
<_sre.SRE_Match object at 0x008EEB48>
>>> re.search(pattern, 'MCMLXXXX')
>>>
```

Expresiones Regulares

```
>>> pattern =
'^M{0,2}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})$'
>>> re.search(pattern, 'MDLV')
<_sre.SRE_Match object at 0x008EEB48>
>>> re.search(pattern, 'MMDCLXVI')
<_sre.SRE_Match object at 0x008EEB48>
>>> re.search(pattern, 'MMMMDCCCLXXXVIII')
<_sre.SRE_Match object at 0x008EEB48>
>>> re.search(pattern, 'I')
<_sre.SRE_Match object at 0x008EEB48>
```

Expresiones Regulares

Summary

^ matches the beginning of a string.

$ matches the end of a string.

\b matches a word boundary.

\d matches any numeric digit.

\D matches any non-numeric character.

x? matches an optional x character (in other words, it matches an x zero or one times).

x* matches x zero or more times.

x+ matches x one or more times.

x{n,m} matches an x character at least n times, but not more than m times.

(a|b|c) matches either a or b or c.

(x) in general is a remembered group. You can get the value of what matched by using the groups() method of the object returned by re.search.