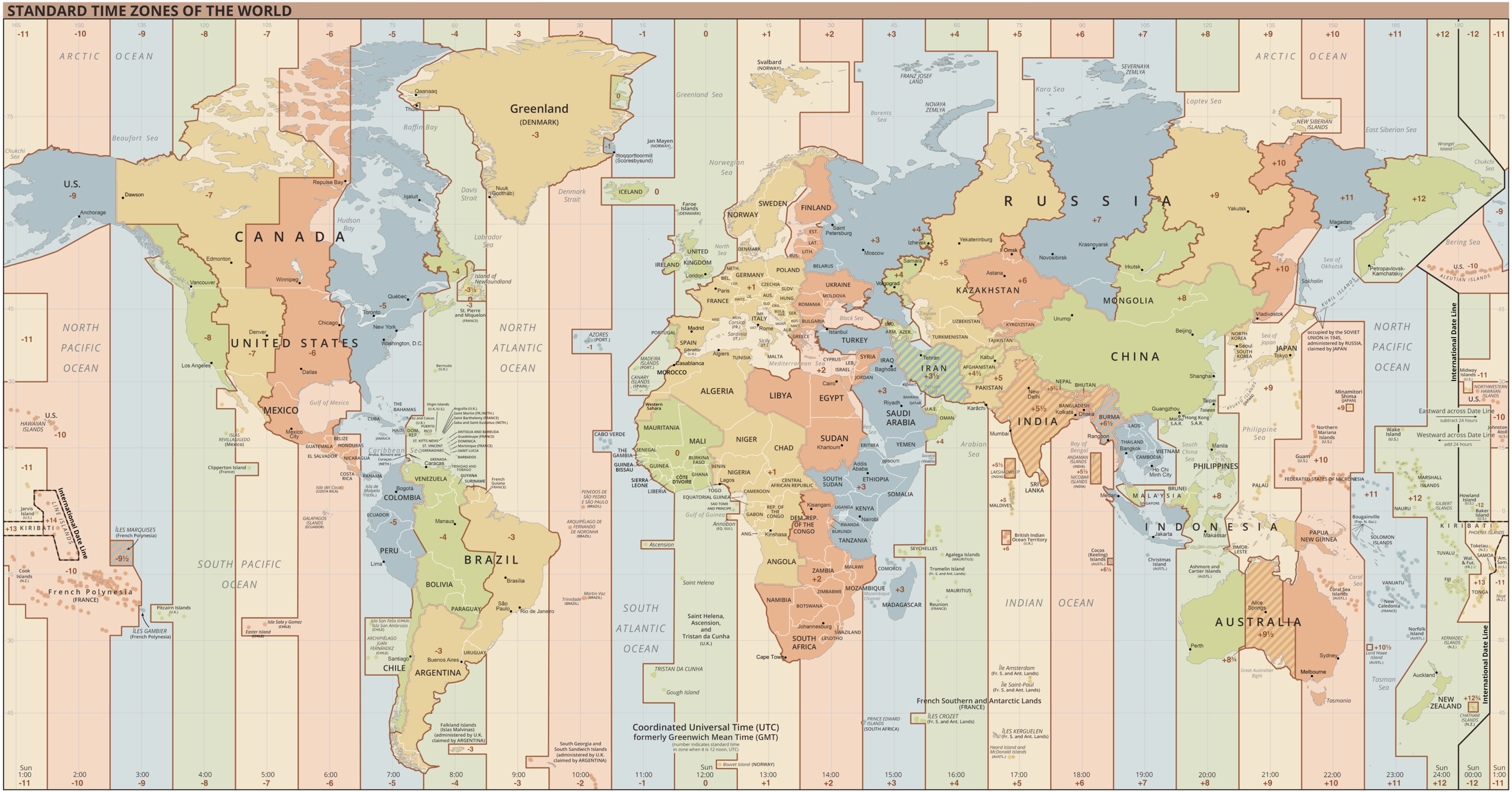
Datetime

ESS 116 | Fall 2024

Prof. Henri Drake, Prof. Jane Baldwin, and Prof. Michael Pritchard

(Modified from Ethan Campbell and Katy Christensen's materials for UW's Ocean 215)



https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/World_Time_Zones_Map.png/1599px-World_Time_Zones_Map.png

Loading datetime

```
from datetime import datetime
```

This is the module

This is the class within the module

Pseudocode:

From the datetime module, I am importing the datetime class which will allow me to use the functions stored there.

Class datetime objects

Get the current date and time

`datetime.now()`

```
1 from datetime import datetime
2 t_now = datetime.now()
3 print(t_now)
4
```

```
2020-10-19 12:41:05.636254
```

Class datetime objects

Get the current date and time

`datetime.now()`

```
1 from datetime import datetime
2 t_now = datetime.now()
3 print(t_now)
4
```

```
2020-10-19 12:41:05.636254
```

Retrieve the individual values from the datetime object

```
1 print(t_now.year)
2 print(t_now.month)
3 print(t_now.day)
4 print(t_now.hour)
5 print(t_now.minute)
6 print(t_now.second)
7 print(t_now.microsecond)
```

```
2020
10
19
12
41
5
636254
```

Class datetime objects

Get the current date and time

`datetime.now()`

```
1 from datetime import datetime
2 t_now = datetime.now()
3 print(t_now)
4
```

```
2020-10-19 12:41:05.636254
```

Retrieve the individual values from the datetime object

```
1 print(t_now.year)
2 print(t_now.month)
3 print(t_now.day)
4 print(t_now.hour)
5 print(t_now.minute)
6 print(t_now.second)
7 print(t_now.microsecond)
```

```
2020
10
19
12
41
5
636254
```

Create a datetime object for any time

`datetime(year, month, day, hour, minute, second, microsecond)`

This part is optional

```
1 t_other = datetime(2020,3,8,8,0,0,0)
2 print(t_other)
```

```
2020-03-08 08:00:00
```

Datetime objects to/from strings

```
1 from datetime import datetime
2 t_now = datetime.now()
3 print(t_now)
4
```

```
2020-10-19 12:41:05.636254
```

Change into string

`datetime.strftime()`

```
1 datestring = datetime.strftime(t_now,'%Y/%m/%d %H.%M.%S')
2
3 print(datestring)
4 print(type(datestring))
```

```
2020/10/19 12.41.05
<class 'str'>
```

Change from a string

`datetime.strptime()`

```
1 t_now_back = datetime.strptime(datestring,'%Y/%m/%d %H.%M.%S')
2
3 print(t_now_back)
4 print(type(t_now_back))
```

```
2020-10-19 12:41:05
<class 'datetime.datetime'>
```

String datetime formatting

Directive	Meaning
`%a`	Weekday as locale's abbreviated name.
`%A`	Weekday as locale's full name.
`%w`	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.
`%d`	Day of the month as a zero-padded decimal number.
`%b`	Month as locale's abbreviated name.
`%B`	Month as locale's full name.
`%m`	Month as a zero-padded decimal number.
`%y`	Year without century as a zero-padded decimal number.
`%Y`	Year with century as a decimal number.
`%H`	Hour (24-hour clock) as a zero-padded decimal number.
`%I`	Hour (12-hour clock) as a zero-padded decimal number.
`%p`	Locale's equivalent of either AM or PM.
`%M`	Minute as a zero-padded decimal number.
`%S`	Second as a zero-padded decimal number.
`%f`	Microsecond as a decimal number, zero-padded on the left.
`%z`	UTC offset in the form +HHMM or -HHMM (empty string if the the object is naive).
`%Z`	Time zone name (empty string if the object is naive).
`%j`	Day of the year as a zero-padded decimal number.
`%U`	Week number of the year (Sunday as the first day of the week) as a zero padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.
`%W`	Week number of the year (Monday as the first day of the week) as a decimal number. All days in a new year preceding the first Monday are considered to be in week 0.
`%c`	Locale's appropriate date and time representation.
`%x`	Locale's appropriate date representation.
`%X`	Locale's appropriate time representation.
`%%`	A literal `'%'` character.

How to deal with time passing

datetime objects are

snapshots of a specific time

```
1 from datetime import datetime
2 t_now = datetime.now()
3 print(t_now)
4
```

```
2020-10-19 12:41:05.636254
```

How to deal with time passing

datetime objects are snapshots of a specific time

```
1 from datetime import datetime
2 t_now = datetime.now()
3 print(t_now)
4
```

```
2020-10-19 12:41:05.636254
```

To reflect time passing, use timedelta objects

```
1 t1 = datetime(2020,3,8)
2 t2 = datetime(2020,10,21)
3
4 time_diff = t2 - t1
5 print(time_diff)
6 print(type(time_diff))
```

```
227 days, 0:00:00
<class 'datetime.timedelta'>
```

How to deal with time passing

datetime objects are snapshots of a specific time

```
1 from datetime import datetime
2 t_now = datetime.now()
3 print(t_now)
4
```

```
2020-10-19 12:41:05.636254
```

To reflect time passing, use timedelta objects

```
1 t1 = datetime(2020,3,8)
2 t2 = datetime(2020,10,21)
3
4 time_diff = t2 - t1
5 print(time_diff)
6 print(type(time_diff))
```

```
227 days, 0:00:00
<class 'datetime.timedelta'>
```

Retrieve the individual values from the timedelta object

```
1 print(time_diff.days)
2 print(time_diff.seconds)
3 print()
4
5 every_sec = time_diff.total_seconds()
6 print(every_sec)
```

```
227
0

19612800.0
```

How to deal with time passing

datetime objects are snapshots of a specific time

```
1 from datetime import datetime
2 t_now = datetime.now()
3 print(t_now)
4
```

```
2020-10-19 12:41:05.636254
```

To reflect time passing, use timedelta objects

```
1 t1 = datetime(2020,3,8)
2 t2 = datetime(2020,10,21)
3
4 time_diff = t2 - t1
5 print(time_diff)
6 print(type(time_diff))
```

```
227 days, 0:00:00
<class 'datetime.timedelta'>
```

Retrieve the individual values from the timedelta object

```
1 print(time_diff.days)
2 print(time_diff.seconds)
3 print()
4
5 every_sec = time_diff.total_seconds()
6 print(every_sec)
```

```
227
0

19612800.0
```

A timedelta object can alter a datetime object

```
1 from datetime import datetime
2 from datetime import timedelta
3
4 time1 = datetime(1991,7,8)
5 time1_future = time1 + timedelta(days=365)
6
7 print(time1)
8 print(time1_future)
```

```
1991-07-08 00:00:00
1992-07-07 00:00:00
```

Datetime resource

https://docs.python.org/3.4/library/datetime.html

Table Of Contents

8.1. `datetime` — Basic date and time types

The `datetime` module supplies classes for manipulating dates and times in both simple
supported, the focus of the implementation is on efficient attribute extraction for output f
see also the `time` and `calendar` modules.