3.10.2. datetime

Fig. 3.10.2.1 Photo by Edgar on Unsplash

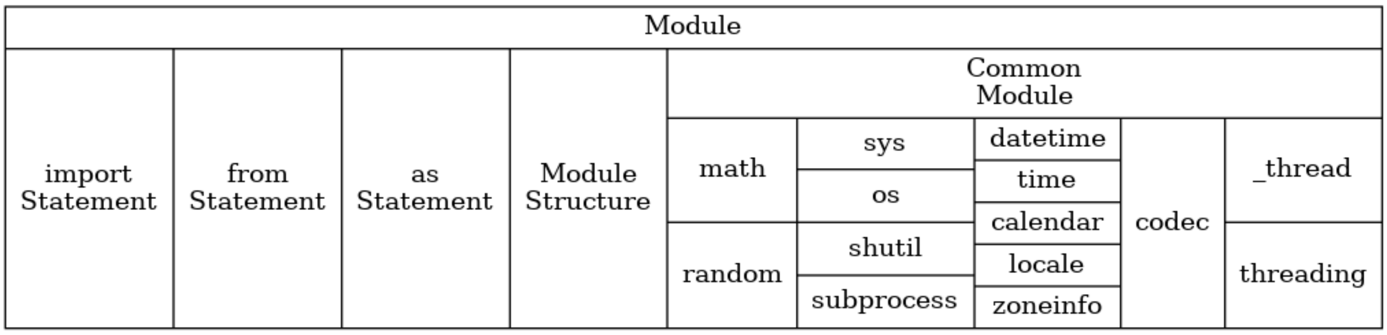
Note

Outline

Note

Roadmap

1. This topic: Module



Module								
import Statement	from Statement	as Statement	Module Structure	Common Module				
				math	sys	datetime	codec	_thread
					os	time		
						calendar		
				random	shutil	locale		threading
					subprocess	zoneinfo		

2. Course: Python 1
3. Subject: Programming
4. Field
    a. Software Engineering (SE)
    b. Computer Science and Information Engineering (CSIE)
    c. Electrical/Electronics Engineering (EE)

3.10.2.1. Overview

1. Module: datetime

3.10.2.1.1. Ex1: Constants

1. Code

Listing 3.10.2.1.1.1 /src/DateTime/DateTimeConstants/__init__.py

```python
import datetime

print('允許最小的西元年: ', datetime.MINYEAR)
print('允許最大的西元年: ', datetime.MAXYEAR)
```

2. Output

```
允許最小的西元年:  1
允許最大的西元年:  9999
```

3.10.2.1.2. Data Types

1. The following data types are immutable.
2. The subclass relationship is shown as below.

3.10.2.1.3. Object: timedelta

1. A timedelta object represents a duration, the difference between two dates or times.

3.10.2.1.3.1. Ex2: datetime.timedelta()

1. Code

Listing 3.10.2.1.3.1.1 /src/DateTime/DateTimeTimeDelta/__init__.py

```python
from datetime import timedelta

td365Day = timedelta(days = 365)
print('type(td365Day)=', type(td365Day))
print('td365Day =', td365Day)
print('td365Day.days =', td365Day.days)
print('td365Day.seconds =', td365Day.seconds)
print('td365Day.microseconds =', td365Day.microseconds)
print('td365Day.resolution =', td365Day.resolution)
print('td365Day.min =', td365Day.min)
print('td365Day.max =', td365Day.max)
print('td365Day.total_seconds() =', td365Day.total_seconds())

print()
td10Year = td365Day * 10
print('type(td10Year)=', type(td10Year))
print('td10Year =', td10Year)
print('td10Year.days =', td10Year.days)

print()
td10Day = td10Year // 365
print('type(td10Day)=', type(td10Day))
print('td10Day =', td10Day)
print('td10Day.days =', td10Day.days)

print()
td8Year = td10Year - td10Day - td365Day
print('type(td8Year)=', type(td8Year))
print('td8Year =', td8Year)
print('td8Year.days =', td8Year.days)

```

2. Output

```
type(td365Day)= <class 'datetime.timedelta'>
td365Day = 365 days, 0:00:00
td365Day.days = 365
td365Day.seconds = 0
td365Day.microseconds = 0
td365Day.resolution = 0:00:00.000001
td365Day.resolution = 0:00:00.000001
td365Day.min = -999999999 days, 0:00:00
td365Day.max = 999999999 days, 23:59:59.999999
td365Day.total_seconds() = 31536000.0

type(td10Year)= <class 'datetime.timedelta'>
td10Year = 3650 days, 0:00:00
td10Year.days = 3650

type(td10Day)= <class 'datetime.timedelta'>
td10Day = 10 days, 0:00:00
td10Day.days = 10

```

```
type(td8Year)= <class 'datetime.timedelta'>
td8Year = 3275 days, 0:00:00
td8Year.days = 3275
```

3.10.2.1.4. Object: date

3.10.2.1.4.1. Ex3a: date.today()

1. Code

Listing 3.10.2.1.4.1.1 /src/DateTime/DateTimeDateToday/__init__.py

```python
from datetime import date

dToday = date.today()
print('type(dToday)=', type(dToday))
print('dToday =', dToday)

print('dToday.resolution =', dToday.resolution)
print('dToday.min =', dToday.min)
print('dToday.max =', dToday.max)
print('dToday.year =', dToday.year)
print('dToday.month =', dToday.month)
print('dToday.day =', dToday.day)
```

2. Output

```
type(dToday)= <class 'datetime.date'>
dToday = 2022-05-22
dToday.resolution = 1 day, 0:00:00
dToday.min = 0001-01-01
dToday.max = 9999-12-31
dToday.year = 2022
dToday.month = 5
dToday.day = 22
```

3.10.2.1.4.2. Ex3b: date.weekday()

1. Code

Listing 3.10.2.1.4.2.1 /src/DateTime/DateTimeDateWeekday/__init__.py

```python
from datetime import date, timedelta

dToday = date.today()
print('type(dToday)=', type(dToday))
print('dToday =', dToday)
print('dToday.weekday() =', dToday.weekday())
print('dToday.isoweekday() =', dToday.isoweekday())
print()
dtDay = timedelta(days = 1)
print('dtDay = ', dtDay)
print()
dTheDayAfterToday = dToday + dtDay
print('dTheDayAfterToday =', dTheDayAfterToday)
print('dTheDayAfterToday.weekday() =', dTheDayAfterToday.weekday())
print('dTheDayAfterToday.isoweekday() =', dTheDayAfterToday.isoweekday())
```

2. Output

```
type(dToday)= <class 'datetime.date'>
dToday = 2022-05-22                  # Sunday
dToday.weekday() = 6                 # Sunday
dToday.isoweekday() = 7              # Sunday

dtDay =  1 day, 0:00:00

dTheDayAfterToday = 2022-05-23       # Monday
dTheDayAfterToday.weekday() = 0      # Monday
dTheDayAfterToday.isoweekday() = 1   # Monday
```

3.10.2.1.4.3. Ex3c: Format

1. Code

Listing 3.10.2.1.4.3.1 /src/DateTime/DateTimeDateFormat/__init__.py

```python
from datetime import date

dToday = date.today()
print('dToday =', dToday)
print('dToday.isoformat() =', dToday.isoformat())
print('dToday.strftime() =', dToday.strftime('%y/%m/%d'))
print('dToday.strftime() =', dToday.strftime('%A %d %B %Y'))
print('dToday.ctime() =', dToday.ctime())
```

2. Output

```
dToday = 2022-05-22
dToday.isoformat() = 2022-05-22
dToday.strftime() = 22/05/22
dToday.strftime() = Sunday 22 May 2022
dToday.ctime() = Sun May 22 00:00:00 2022
```

3.10.2.1.4.4. Ex3d: date.timetuple()

1. Code

Listing 3.10.2.1.4.4.1 /src/DateTime/DateTimeDateTimetuple/__init__.py

```python
from datetime import date, time

dToday = date.today()
print('dToday =', dToday)
print('dToday.timetuple() =', dToday.timetuple())

lsStructTime = ['tm_year', 'tm_mon', 'tm_mday', 'tm_hour', 'tm_min',
                'tm_sec', 'tm_wday', 'tm_yday', 'tm_isdst']
dsStructTime = dict(zip(lsStructTime, list(dToday.timetuple())))
print(dsStructTime)
for k, v in dsStructTime.items():
    print(k, v)
```

2. Output

```
dToday = 2022-05-22
dToday.timetuple() = time.struct_time(tm_year=2022, tm_mon=5, tm_mday=22, tm_hour=0,
tm_min=0, tm_sec=0, tm_wday=6, tm_yday=142, tm_isdst=-1)
{'tm_year': 2022, 'tm_mon': 5, 'tm_mday': 22, 'tm_hour': 0, 'tm_min': 0, 'tm_sec': 0,
'tm_wday': 6, 'tm_yday': 142, 'tm_isdst': -1}
```

```
tm_year 2022
tm_mon 5
tm_mday 22
tm_hour 0
tm_min 0
tm_sec 0
tm_wday 6
tm_yday 142
tm_isdst -1
```

3.10.2.1.4.5. Ex3e: date.isocalendar()

1. Code

Listing 3.10.2.1.4.5.1 /src/DateTime/DateTimeDateIsoCalendar/__init__.py

```python
from datetime import date, time

dToday = date.today()
print('dToday =', dToday)
print('dToday.isocalendar() =', dToday.isocalendar())

lsIsoCal = ['year', 'week', 'weekday']
dsIsoCal = dict(zip(lsIsoCal, list(dToday.isocalendar())))
print(dsIsoCal)
for k, v in dsIsoCal.items():
    print(k, v)
```

2. Output

```
dToday = 2022-05-22
dToday.isocalendar() = datetime.IsoCalendarDate(year=2022, week=20, weekday=7)
{'year': 2022, 'week': 20, 'weekday': 7}
year 2022
week 20
weekday 7
```

3.10.2.1.5. Object: datetime

3.10.2.1.5.1. Ex4a: datetime.today()

1. Return the current local datetime, with tzinfo None.
2. Code

Listing 3.10.2.1.5.1.1 /src/DateTime/DateTimeDateTimeToday/__init__.py

```python
from datetime import datetime

dtToday = datetime.today()
print('type(dtToday)=', type(dtToday))
print('dtToday =', dtToday)

print('dtToday.resolution =', dtToday.resolution)
print('dtToday.min =', dtToday.min)
print('dtToday.max =', dtToday.max)
print('dtToday.year =', dtToday.year)
print('dtToday.month =', dtToday.month)
print('dtToday.day =', dtToday.day)
print('dtToday.hour =', dtToday.hour)
print('dtToday.minute =', dtToday.minute)
print('dtToday.second =', dtToday.second)
print('dtToday.microsecond =', dtToday.microsecond)
print('dtToday.tzinfo =', dtToday.tzinfo)
```

3. Output

```
type(dtToday)= <class 'datetime.datetime'>
dtToday = 2022-05-22 18:55:58.758795
dtToday.resolution = 0:00:00.000001
dtToday.min = 0001-01-01 00:00:00
dtToday.max = 9999-12-31 23:59:59.999999
dtToday.year = 2022
dtToday.month = 5
dtToday.day = 22
dtToday.hour = 18
dtToday.minute = 55
dtToday.second = 58
dtToday.microsecond = 758795
dtToday.tzinfo = None
```

3.10.2.1.5.2. Ex4b: datetime.now()

1. Return the current local date and time.

2. Code

Listing 3.10.2.1.5.2.1 /src/DateTime/DateTimeDateTimeNow/__init__.py

```
from datetime import datetime

dtNow = datetime.now()
print('type(dtNow)=', type(dtNow))
print('dtNow =', dtNow)

print('dtNow.resolution =', dtNow.resolution)
print('dtNow.min =', dtNow.min)
print('dtNow.max =', dtNow.max)
print('dtNow.year =', dtNow.year)
print('dtNow.month =', dtNow.month)
print('dtNow.day =', dtNow.day)
print('dtNow.hour =', dtNow.hour)
print('dtNow.minute =', dtNow.minute)
print('dtNow.second =', dtNow.second)
print('dtNow.microsecond =', dtNow.microsecond)
print('dtNow.tzinfo =', dtNow.tzinfo)
```

3. Output

```
type(dtNow)= <class 'datetime.datetime'>
dtNow = 2022-05-22 18:55:15.070814
dtNow.resolution = 0:00:00.000001
dtNow.min = 0001-01-01 00:00:00
dtNow.max = 9999-12-31 23:59:59.999999
dtNow.year = 2022
dtNow.month = 5
dtNow.day = 22
dtNow.hour = 18
dtNow.minute = 55
dtNow.second = 15
dtNow.microsecond = 70814
dtNow.tzinfo = None
```

3.10.2.1.5.3. Ex4c: datetime.utcnow()

1. Return the current UTC date and time, with tzinfo None.
2. Code

Listing 3.10.2.1.5.3.1 /src/DateTime/DateTimeDateTimeUtcNow/__init__.py

```python
from datetime import datetime

dtUtcNow = datetime.utcnow()
print('type(dtUtcNow)=', type(dtUtcNow))
print('dtUtcNow =', dtUtcNow)

print('dtUtcNow.resolution =', dtUtcNow.resolution)
print('dtUtcNow.min =', dtUtcNow.min)
print('dtUtcNow.max =', dtUtcNow.max)
print('dtUtcNow.year =', dtUtcNow.year)
print('dtUtcNow.month =', dtUtcNow.month)
print('dtUtcNow.day =', dtUtcNow.day)
print('dtUtcNow.hour =', dtUtcNow.hour)
print('dtUtcNow.minute =', dtUtcNow.minute)
print('dtUtcNow.second =', dtUtcNow.second)
print('dtUtcNow.microsecond =', dtUtcNow.microsecond)
print('dtUtcNow.tzinfo =', dtUtcNow.tzinfo)
```

3. Output

```
type(dtUtcNow)= <class 'datetime.datetime'>
dtUtcNow = 2022-05-22 10:54:36.941827
dtUtcNow.resolution = 0:00:00.000001
dtUtcNow.min = 0001-01-01 00:00:00
dtUtcNow.max = 9999-12-31 23:59:59.999999
dtUtcNow.year = 2022
dtUtcNow.month = 5
dtUtcNow.day = 22
dtUtcNow.hour = 10
dtUtcNow.minute = 54
dtUtcNow.second = 36
dtUtcNow.microsecond = 941827
dtUtcNow.tzinfo = None
```

3.10.2.1.6. Object: time

3.10.2.1.6.1. Ex5: time.time()

1. Code

Listing 3.10.2.1.6.1.1 /src/DateTime/DateTimeTimeTime/__init__.py

```python
from datetime import time

dtTime = time(hour = 0,
              minute = 2,
              second = 30,
              microsecond = 20,
              tzinfo = None)
print('type(dtTime)=', type(dtTime))
print('dtTime =', dtTime)

print('dtTime.resolution =', dtTime.resolution)
print('dtTime.min =', dtTime.min)
print('dtTime.max =', dtTime.max)
print('dtTime.hour =', dtTime.hour)
print('dtTime.minute =', dtTime.minute)
print('dtTime.second =', dtTime.second)
print('dtTime.microsecond =', dtTime.microsecond)
print('dtTime.tzinfo =', dtTime.tzinfo)
```

2. Output

```
type(dtTime)= <class 'datetime.time'>
dtTime = 00:02:30.000020
dtTime.resolution = 0:00:00.000001
dtTime.min = 00:00:00
dtTime.max = 23:59:59.999999
dtTime.hour = 0
dtTime.minute = 2
dtTime.second = 30
dtTime.microsecond = 20
dtTime.tzinfo = None
```

3.10.2.1.7. Object: tzinfo

3.10.2.1.7.1. Ex6: tzinfo.tzname()

1. Return the time zone name corresponding to the datetime object dt, as a string.
2. Code

Listing 3.10.2.1.7.1.1 /src/DateTime/TzinfoTzname/__init__.py

```python
from datetime import datetime
import pytz

dt = datetime.now()

print(dt)
print(dt.tzinfo)
print("Timezone:", dt.tzname())
print()

timezone = pytz.timezone("Asia/Taipei")
mydt = timezone.localize(dt)
print(mydt)
print("Tzinfo:", mydt.tzinfo)
print("Timezone name:", mydt.tzname())
```

3. Output

```
2022-05-23 00:15:53.595512
None
Timezone: None

2022-05-23 00:15:53.595512+08:00
Tzinfo: Asia/Taipei
Timezone name: CST
```

3.10.2.1.8. Object: timezone

3.10.2.1.8.1. Ex7: pytz.timezone()

1. Code

Listing 3.10.2.1.8.1.1 /src/DateTime/TimeZonePytz/__init__.py

```python
import pytz
from datetime import datetime, timezone

utc_dt = datetime.now(timezone.utc)

TWN = pytz.timezone('Asia/Taipei')
PST = pytz.timezone('US/Pacific')
EST = pytz.timezone('US/Eastern')

# Use astimezone() without an argument
print("Local time   {}".format(utc_dt.astimezone().isoformat()))
print("Taiwan time  {}".format(utc_dt.astimezone(TWN).isoformat()))
print("UTC time     {}".format(utc_dt.isoformat()))
print("Pacific time {}".format(utc_dt.astimezone(PST).isoformat()))
print("Eastern time {}".format(utc_dt.astimezone(EST).isoformat()))

```

2. Output

```
Local time   2022-05-22T23:53:04.356240+08:00
Taiwan time  2022-05-22T23:53:04.356240+08:00
UTC time     2022-05-22T15:53:04.356240+00:00
Pacific time 2022-05-22T08:53:04.356240-07:00
Eastern time 2022-05-22T11:53:04.356240-04:00
```

See also

1. `datetime: Basic date and time types, Python v3.10.4 <https://docs.python.org/3/library/datetime.html#module-datetime> `__

Note

1. Start: 20170719
2. System Environment:

Listing 3.10.2.1.8.1.2 requirements.txt

```
sphinx==7.1.2                           # Sphinx
graphviz>=0.20.1                        # Graphviz
sphinxbootstrap4theme>=0.6.0            # Theme: Bootstrap
sphinx-material>=0.0.35                 # Theme: Material
sphinxcontrib-plantuml>=0.25            # PlantUML
sphinxcontrib.bibtex>=2.5.0             # Bibliography
sphinx-autorun>=1.1.1                   # ExecCode: pycon
sphinx-execute-code-python3>=0.3        # ExecCode
btd.sphinx.inheritance-diagram>=2.3.1   # Diagram
sphinx-copybutton>=0.5.1                # Copy button
sphinx_code_tabs>=0.5.3                 # Tabs
sphinx-immaterial>=0.11.3               # Tabs

#------------------------------------------------------
#-- Library Upgrade Error by Library Itself
#    >> It needs to fix by library owner
#    >> After fixed, we need to try it later
#------------------------------------------------------
pydantic==1.10.10                       # 2.0: sphinx compiler error, 20230701

#------------------------------------------------------
#-- Minor Extension
#------------------------------------------------------
sphinxcontrib.httpdomain>=1.8.1         # HTTP API

#sphinxcontrib-blockdiag>=3.0.0         # Diagram: block
#sphinxcontrib-actdiag>=3.0.0           # Diagram: activity
#sphinxcontrib-nwdiag>=2.0.0            # Diagram: network
#sphinxcontrib-seqdiag>=3.0.0           # Diagram: sequence

#------------------------------------------------------
#-- Still Wait For Upgrading Version
#------------------------------------------------------

#------------------------------------------------------
#-- Still Under Testing
#------------------------------------------------------
#numpy>=1.24.2                          # Figure: numpy

#------------------------------------------------------
#-- NOT Workable
#------------------------------------------------------
#sphinxcontrib.jsdemo==0.1.4    # ExecCode: Need replace add_js_file()
#jupyter-sphinx==0.4.0          # ExecCode: Need gcc compiler
#sphinxcontrib.slide==1.0.0     # Slide: Slideshare
#hieroglyph==2.1.0              # Slide: make slides
#matplotlib>=3.7.1              # Plot: Need Python >= v3.8
#manim==0.17.2                  # Diagram: scipy, numpy need gcc
#sphinx_diagrams==0.4.0         # Diagram: Need GKE access
#sphinx-tabs>=3.4.1                     # Tabs: Conflict w/ sphinx-material
```