

Time Module - asctime() and mktime() functions

```
import time
timestamp = 1572879180
st = time.gmtime(timestamp)
print(time.asctime(st))
print(time.localtime((2019, 11, 4, 14, 53, 0, 0, 308, 0)))
```

Import Modules

```
import math
import sys
import math, sys
from math import pi
from math import pi, sin
import math as m
from math import pi as p
from math import pi as PI, sin as sinus
```

Working with modules

```
dir()    list all entities within a module
```

Random Module

```
from random import seed(a=None, version=2) - initializes random number generator, a=None current time used
from random import random() - random float ranging 0.0<= X < 1.0
import random
from random import randrange(start, stop[, step])
import randrange
from random import randint(a, b) - a <= N <= b. Alias for randrange(a, b+1)
import randint
from random import choice(seq) - return random element
import choice
from random import sample(population, k, *, counts=None) - return list of elements
import sample
```

Platform Module

```
from platform import platform(aliased=0, terse=0) - returns single string showing underlying platform
from platform import machine() - return machine type e. g. 'AMD64'
import machine
```

Platform Module (cont)

```
from platform import processor() - return real processor name
processor    'amd64'
from platform import system() - return OS name e. g. 'Windows', 'Linux'
system
from platform import version() - return system release
version
from platform import python_implementation() - return python version as string 'major.minor.patchlevel'
python_implementation
```

Sorting

```
sorted()    new list gets created
sort()      performed in situ, changes the original list
```

OOP - check attributes

```
class Sampleclass:
    a = 1
    def __init__(self):
        self.b = 2

object = Sampleclass()

print(hasattr(object, 'b'))
print(hasattr(object, 'a'))
print(hasattr(Sampleclass, 'b'))
print(hasattr(Sampleclass, 'a'))
```

OOP - class variable

```
class Sampleclass:
    counter = 0
    def __init__(self, var = 1):
        self.__first = var
        Sampleclass.counter += 1
```



OOP - instance variable

```
class Sampleclass:
    def __init__(self, var = 1):
        self.first = var
    def set_second(self, var):
        self.second = var
```

Object Oriented Programming (OOP) - general

class is a set of objects

objects belongs to a class

object is incarnation of requirements, properties, qualities of a class

OOP - Method Vars

```
class Sample:
    varia = 2
    def method(self):
        print(self.varia, self.var)

obj = Sample()
obj.var = 3
obj.method()
```

OOP - Method Resolution Order (MRO)

Way a language scans through the upper part of classes

class Bottom(Middle, Top) works when Top created before Middle

class Bottom(Top, Middle) fails when Top created before Middle

OOP - Vars (some)

<code>__dict__</code>	shows entities
<code>__name__</code>	name of class
<code>__module__</code>	name of module
<code>__bases__</code>	names of direct superclasses

OOP - Sub/Superclass and Incarnation

`issubclass()` identifies relationship between 2 classes

each class considers itself to be a subclass of itself

`isinstance()` Checks if object is an incarnation of a class

"Is" operator

checks whether two variables refer to the same object

variables don't store the object, they point to internal Python memory

OOP - Inheritance with more than one class

class Sub(Left, Right) Python scans left to right

if vars duplicate, "Left" vars will be shown

Generators and Iterators - general

Generator produce a series of values and control iteration process (`range()`)

Iterator is an object conforming to iterator protocol (`__iter__()`, `__next__()`)

Iterator with yield

```
FAILS
def func(n):
    for i in range(n):
        return i

WORKS
def func(n):
    for i in range(n):
        yield i
```



List vs Generator

```
# list generated and iterated through:
for v in [1 if x % 2 == 0 else 0 for x in range( -
10)]:
    print(v, end=" ")
print()
# subsequent values are generated one by one:
for v in (1 if x % 2 == 0 else 0 for x in range( -
10)):
    print(v, end=" ")
print()
```

File streams

read
write
update

File streams open, close, std

open()	open stream
close()	close stream
standard streams with 'import sys'	sys.stdin
	sys.stdout
	sys.stderr

File streams continued

'r'	read mode
'w'	write mode
'a'	append mode
'r+'	read and update mode
'w+'	write and update mode
'rt' / 'rb'	read text / binary
'wt' / 'wb'	write text / binary
'at' / 'ab'	append text / binary
'r+t' / 'r+b'	read and update text/binary
'w+t' / 'w+b'	write and update text/binary

OS Module

os.uname()	doesn't work with Windows
os.name()	works with Windows
os.listdir()	print directories
os.mkdir()	create directory
os.makedirs()	create multiple directories
os.chdir()	change directory
os.getcwd()	show current directory
os.rmdir()	delete directory
os.removedirs()	removes directories recursively

Datetime Module - Class Date

```
from datetime import date
v_today = date.today()
print(v_today.year)
print(v_today.month)
print(v_today.day)
print(v_today.today)
###
from datetime import date
d = date(2019, 11, 4)
print(d.weekday())
print(d.isoweekday())
```

Datetime Module - Class Time

```
from datetime import time
t = time(14, 53, 20, 1)
print("Time:", t)
print("Hour:", t.hour)
print("Minute:", t.minute)
print("Second:", t.second)
print("Microsecond:", t.microsecond)
```



Date and time operations

```
from datetime import date
from datetime import datetime
d1 = date(2020, 11, 4)
d2 = date(2019, 11, 4)
print(d1 - d2)
dt1 = datetime(2020, 11, 4, 0, 0, 0)
dt2 = datetime(2019, 11, 4, 14, 53, 0)
print(dt1 - dt2)
```

Calendar Module

```
import calendar
print(calendar.Calendar(2023))
calendar.prcal(2023)
```

Calendar for a month

```
import calendar
print(calendar.month(2020, 11))
```

Calendar Objects

```
import calendar
c = calendar.Calendar(calendar.SUNDAY)
for weekday in c.iterweekdays():
    print(weekday, end=" ")
```

Calendar Create Classes

calendar.Calendar	provides methods for calendar data formatting
calendar.TextCalendar	create text calendars
calendar.HTMLCalendar	create HTML calendars
calendar.LocalTextCalendar	subclass (see above) uses local parameter
calendar.LocalHTMLCalendar	subclass (see above) uses local parameter

Calendar set first week day

```
import calendar
calendar.setfirstweekday(calendar.SUNDAY)
calendar.pmonth(2020, 12)
```

PIP

pip --version	show pip version (pip version 2)
pip3 --version	show pip version (pip version 3)
pip help	show pip commands
pip help operation	show info about a specific operation
pip show package_name	show info about installed package
pip search string	search for a package
pip install package	package installation

String Operations

ord()	ordinal, code point
chr()	character
possible operations	indexing, iterating, slicing, 'in' and 'not' operators
impossible operations (immutable)	append, del, insert
max(), min()	uses ASCII value
additional operations	index(), list(), count()

String Methods

capitalize()	stRINGdiNg -> stRINGdiNg
endswith(suffix[, start[, end]])	'stringding'.endswith('ding') return True
find(sub[, start[, end]])	'stringding'.find('in') return 3 (index #)
isalnum()	'string10' -> True, 'string_10' -> False



String Methods (cont)

isalpha()	'string' -> True, 'string10' -> False
isdigit()	'10' -> True, 'string10' -> False
islower()	'stringding' -> True, 'strinGding' -> False
isspace()	'\n' -> True, '' -> True, 'string ding' -> False
isupper()	'string' -> False, 'stRING' -> False, 'STRING' -> True
join()	",".join(['li', 'st', 'joined']) -> 'li,st,joined'
lower()	'stRinGding' -> 'stringding'
lstrip(str)	'stringding'.lstrip('str') -> 'ingding'
rstrip(str)	'stringding'.rstrip('ding') -> 'str'
replace(old, new[, count])	'stringding'.lstrip('str') -> 'strAngdAng'
rfind(sub[, start[, end]])	'stringding'.rfind('n') -> 8, ('n', 5) -> 8, ('n', 10) -> -1 (failure)
split(str)	'stringding'.split('i') -> ['str', 'ngd', 'ng']
startswith(str)	'stringding'.startswith('str') -> True, 'stringding'.startswith('ring') -> False
strip() remove leading, trailing whitespaces	' stringding ' -> 'stringding'
swapcase()	'strinGDiNg' -> 'STRINgdlNg'

String Methods (cont)

title()	'this Is a title' -> 'This Is A Title'
upper()	'strinGDiNg' -> 'STRINGDING'

String Comparison

Compare	==, !=, <, >, <=, >=
string == number	False
string != umber	True
string >= number	Exception

OOP - Method Invocation

```
class Sample:
    def other( self):
        pri nt( " oth er")

    def method (self):
        pri nt( " met hod ")
        sel f.o ther()

obj = Sample()
obj.me thod()
```

OOP - Constructor

```
class Sample:
def __init__(self, value = None):
self.var = value
obj_1 = Sample("object")
obj_2 = Sample()
print(obj_1.var)
print(obj_2.var)
```



OOP - Hiding

```
class Sample:
def visible(self):
print("visible")
def __hidden(self):
print("hidden")
obj = Sample()
obj.visible()
try:
obj.__hidden()
except:
print("failed")
obj._Sample__hidden()
```

Exception continued

```
try:
    this is what we want
except:
    this is what we do with errors
else:
    we do this when no exception has been raised
and our " try " was successful
final:
    no matter of what the outcome is we do that
```

Exceptions are classes

```
try:
    raise Exception
except Exception as e:
    print(e, e.__str__(), e.args)
```

Lambda

lambda parameter: expression	lambda x: x * 2
	lambda x, y: x * y
	lambda x, y, z: (x * y) / z

map() function

map(arg1, arg2)	list = list(map(lambda x: x * 2, sourcelist))
arg1	a function what to do
arg2	iterable object (list, tuple, generator, ...)

filter() function

filter items based on a given statement	
filter(arg1, arg2)	list = list(filter(lambda x: x > 2, sourcelist))
arg1	function to what to filter
arg2	iterable object (list, tuple, generator, ...)

Conditional Expression

```
print(True if 0 >= 0 else False)
```

Work with files

read(1)	read chars -> 1 byte chunks
read()	read chars all at once
readline(10)	read lines -> 10 byte chunks
readline()	read full line
readlines(10)	read all lines -> 10 byte size chunks
readlines()	read all lines
write(arg1)	write to a file



Bytearray

`data = bytearray(10)` creates a bytearray filled with 0's
are mutable, `len()`, `index()`
only integers, only 0 to 255

Bytearray - write to binary file

```
from os import strerror
data = bytearray(10)
for i in range(len(data)):
    data[i] = 10 + i
try:
    bf = open('file.bin', 'wb')
    bf.write(data)
    bf.close()
except IOError as e:
    print("I/O error occurred: ", strerror(e.errno))
```

Bytearray - read from binary file

```
from os import strerror
data = bytearray(10)
try:
    binary_file = open('file.bin', 'rb')
    binary_file.readinto(data) #
file.read(data) would also work
    binary_file.close()
    for b in data:
        print(hex(b), end=' ')
except IOError as e:
    print("I/O error occurred: ", strerror(e.errno))
```

Time Module

```
import time
timestamp = 1572879180
print(time.localtime(timestamp))
print(time.gmtime(timestamp))
print(time.localtime(timestamp))
print(time.time())
```

timestamp()

```
from datetime import datetime
dt = datetime(2020, 10, 4, 14, 55)
print("Timestamp:", dt.timestamp())
```

strftime()

```
from datetime import time
from datetime import datetime
t = time(14, 53)
print(t.strftime("%H:%M:%S"))
dt = datetime(2020, 11, 4, 14, 53)
print(dt.strftime("%Y/%B/%d %H:%M:%S"))
```

strftime() in Time Module

```
import time
timestamp = 1572879180
st = time.gmtime(timestamp)
print(time.strftime("%Y/%m/%d %H:%M:%S", st))
print(time.strftime("%Y/%m/%d %H:%M:%S"))
```

timedelta()

```
from datetime import timedelta
delta = timedelta(weeks=2, days=2, hours=3)
print(delta)
print("Days:", delta.days)
print("Seconds:", delta.seconds)
```

timedelta() (cont)

```
> print("Microseconds:", delta.microseconds)
###
from datetime import timedelta
from datetime import date
from datetime import datetime
delta = timedelta(weeks=2, days=2, hours=2)
print(delta)
delta2 = delta * 2
print(delta2)
d = date(2019, 10, 4) + delta2
print(d)
dt = datetime(2019, 10, 4, 14, 53) + delta2
print(dt)
```

Calendar weekday()

```
import calendar
print(calendar.weekday(2020, 12, 24))
```

Calendar Iterators

itermonth- returns days of the week as int
dates()

itermonth- return tuple day of the month int and week day int
dates2()

itermonth- return days as tuple year int, month int, day of month
days3() int

itermonth- return days as tuple with year int, month int day of
days4() month int, day of week int

Leap Year

isleap(<year>) Check is year specified is leap year

leapdays(<year1, returns number of leap years in specified
year2>) range



By **Termo**
cheatography.com/termo/

Not published yet.
Last updated 28th June, 2023.
Page 9 of 8.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>