In [ ]:

```
What is a Module?

Consider a module to be the same as a code library.

A file containing a set of functions you want to include in your application.
```

In [1]:

```
from mymodule import person1

print(person1["age"])
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-d791fa9538a2> in <module>
----> 1 from mymodule import person1
      2 
      3 print(person1["age"])

ModuleNotFoundError: No module named 'mymodule'
```

In [2]:

```
import platform

x = dir(platform)
print(x)
```

```
['_WIN32_CLIENT_RELEASES', '_WIN32_SERVER_RELEASES', '__builtins__', '__cach
ed__', '__copyright__', '__doc__', '__file__', '__loader__', '__name__', '__
package__', '__spec__', '__version__', '_comparable_version', '_component_r
e', '_default_architecture', '_follow_symlinks', '_ironpython26_sys_version_
parser', '_ironpython_sys_version_parser', '_java_getprop', '_libc_search',
'_mac_ver_xml', '_node', '_norm_version', '_platform', '_platform_cache', '_
pypy_sys_version_parser', '_sys_version', '_sys_version_cache', '_sys_versio
n_parser', '_syscmd_file', '_syscmd_uname', '_syscmd_ver', '_uname_cache',
'_ver_output', '_ver_stages', 'architecture', 'collections', 'java_ver', 'li
bc_ver', 'mac_ver', 'machine', 'node', 'os', 'platform', 'processor', 'pytho
n_branch', 'python_build', 'python_compiler', 'python_implementation', 'pyth
on_revision', 'python_version', 'python_version_tuple', 're', 'release', 'sy
s', 'system', 'system_alias', 'uname', 'uname_result', 'version', 'win32_edi
tion', 'win32_is_iot', 'win32_ver']
```

In [3]:

```
import platform

x = platform.system()
print(x)
```

```
Windows
```

In [4]:

```
import mymodule as mx

a = mx.person1["age"]
print(a)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-4-d394793d6100> in <module>
----> 1 import mymodule as mx
      2 
      3 a = mx.person1["age"]
      4 print(a)

ModuleNotFoundError: No module named 'mymodule'
```

In [5]:

```
import mymodule

a = mymodule.person1["age"]
print(a)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-5-a0f0a311b65c> in <module>
----> 1 import mymodule
      2 
      3 a = mymodule.person1["age"]
      4 print(a)

ModuleNotFoundError: No module named 'mymodule'
```

In [6]:

```
import mymodule

mymodule.greeting("Jonathan")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-6-926bb1f42553> in <module>
----> 1 import mymodule
      2 
      3 mymodule.greeting("Jonathan")

ModuleNotFoundError: No module named 'mymodule'
```

In [ ]:

```
Python Datetime
```

In [1]:

```python
import datetime

x = datetime.datetime.now()

print(x)
```

```
2021-12-06 17:34:50.558698
```

In [8]:

```python
import datetime

x = datetime.datetime.now()

print(x.year)
print(x.strftime("%A"))
```

```
2021
Thursday
```

In [9]:

```python
import datetime

x = datetime.datetime(2020, 5, 17)

print(x)
```

```
2020-05-17 00:00:00
```

In [10]:

```python
import datetime

x = datetime.datetime(2018, 6, 1)

print(x.strftime("%B"))
```

```
June
```

In [ ]:

```
Directive   Description Example

%a  Weekday, short version  Wed
%A  Weekday, full version   Wednesday
%w  Weekday as a number 0-6, 0 is Sunday    3
%d  Day of month 01-31  31
%b  Month name, short version   Dec
%B  Month name, full version    December
%m  Month as a number 01-12 12
%y  Year, short version, without century    18
%Y  Year, full version  2018
%H  Hour 00-23  17
%I  Hour 00-12  05
%p  AM/PM   PM
%M  Minute 00-59    41
%S  Second 00-59    08
%f  Microsecond 000000-999999   548513
%z  UTC offset  +0100
%Z  Timezone    CST
%j  Day number of year 001-366  365
%U  Week number of year, Sunday as the first day of week, 00-53 52
%W  Week number of year, Monday as the first day of week, 00-53 52
%c  Local version of date and time  Mon Dec 31 17:41:00 2018
%C  Century 20
%x  Local version of date   12/31/18
%X  Local version of time   17:41:00
%%  A % character   %
%G  ISO 8601 year   2018
%u  ISO 8601 weekday (1-7)  1
%V  ISO 8601 weeknumber (01-53) 01
```

In [11]:

```python
import datetime

x = datetime.datetime.now()

print(x.strftime("%a"))
```

```
Thu
```

In [12]:

```python
import datetime

x = datetime.datetime.now()

print(x.strftime("%A"))
```

```
Thursday
```

In [13]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%w"))
```

```
4
```

In [14]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%d"))
```

```
26
```

In [15]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%b"))
```

```
Aug
```

In [16]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%B"))
```

```
August
```

In [17]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%m"))
```

```
08
```

In [18]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%y"))
```

```
21
```

In [19]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%Y"))
```

```
2021
```

In [20]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%H"))
```

```
04
```

In [21]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%I"))
```

```
04
```

In [22]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%p"))
```

```
AM
```

In [23]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%M"))
```

```
23
```

In [24]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%V"))
```

```
34
```

In [25]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%S"))
```

```
10
```

In [26]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%f"))
```

```
380539
```

In [27]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%j"))
```

```
238
```

In [28]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%U"))
```

```
34
```

In [29]:

```
import datetime

x = datetime.datetime(2018, 5, 31)

print(x.strftime("%W"))
```

```
22
```

In [30]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%c"))
```

```
Thu Aug 26 04:24:40 2021
```

In [31]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%C"))
```

```
20
```

In [32]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%C"))
```

```
20
```

In [33]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%x"))
```

```
08/26/21
```

In [34]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%X"))
```

```
04:25:06
```

In [35]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%%"))
```

```
%
```

In [36]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%G"))
```

```
2021
```

In [37]:

```
import datetime

x = datetime.datetime.now()

print(x.strftime("%u"))
```

```
4
```

In [ ]:

```
Python Math
```

In [38]:

```
x = min(5, 10, 25)
y = max(5, 10, 25)

print(x)
print(y)
```

```
5
25
```

In [39]:

```
x = abs(-7.25)

print(x)
```

```
7.25
```

In [40]:

```
x = pow(4, 3)

print(x)
```

```
64
```

In [41]:

```
import math

x = math.sqrt(64)

print(x)
```

```
8.0
```

In [42]:

```
#Import math library
import math

#Round a number upward to its nearest integer
x = math.ceil(1.4)

#Round a number downward to its nearest integer
y = math.floor(1.4)

print(x)
print(y)
```

```
2
1
```

In [43]:

```
import math

x = math.pi

print(x)
```

```
3.141592653589793
```

In [ ]:

```
Python math Module

Python has a built-in module that you can use for mathematical tasks.

The math module has a set of methods and constants.

Math Methods

Method  Description

math.acos() Returns the arc cosine of a number
math.acosh()    Returns the inverse hyperbolic cosine of a number
math.asin() Returns the arc sine of a number
math.asinh()    Returns the inverse hyperbolic sine of a number
math.atan() Returns the arc tangent of a number in radians
math.atan2()    Returns the arc tangent of y/x in radians
math.atanh()    Returns the inverse hyperbolic tangent of a number
math.ceil() Rounds a number up to the nearest integer
math.comb() Returns the number of ways to choose k items from n items without repetition an
math.copysign() Returns a float consisting of the value of the first parameter and the sign
math.cos()  Returns the cosine of a number
math.cosh() Returns the hyperbolic cosine of a number
math.degrees()  Converts an angle from radians to degrees
math.dist() Returns the Euclidean distance between two points (p and q), where p and q are
math.erf()  Returns the error function of a number
math.erfc() Returns the complementary error function of a number
math.exp()  Returns E raised to the power of x
math.expm1()    Returns Ex - 1
math.fabs() Returns the absolute value of a number
math.factorial()    Returns the factorial of a number
math.floor()    Rounds a number down to the nearest integer
math.fmod() Returns the remainder of x/y
math.frexp()    Returns the mantissa and the exponent, of a specified number
math.fsum() Returns the sum of all items in any iterable (tuples, arrays, lists, etc.)
math.gamma()    Returns the gamma function at x
math.gcd()  Returns the greatest common divisor of two integers
math.hypot()    Returns the Euclidean norm
math.isclose()  Checks whether two values are close to each other, or not
math.isfinite() Checks whether a number is finite or not
math.isinf()    Checks whether a number is infinite or not
math.isnan()    Checks whether a value is NaN (not a number) or not
math.isqrt()    Rounds a square root number downwards to the nearest integer
math.ldexp()    Returns the inverse of math.frexp() which is x * (2**i) of the given number
math.lgamma()   Returns the log gamma value of x
math.log()  Returns the natural logarithm of a number, or the logarithm of number to base
math.log10()    Returns the base-10 logarithm of x
math.log1p()    Returns the natural logarithm of 1+x
math.log2() Returns the base-2 logarithm of x
math.perm() Returns the number of ways to choose k items from n items with order and withou
math.pow()  Returns the value of x to the power of y
math.prod() Returns the product of all the elements in an iterable
math.radians()  Converts a degree value into radians
math.remainder()    Returns the closest value that can make numerator completely divisible
math.sin()  Returns the sine of a number
math.sinh() Returns the hyperbolic sine of a number
math.sqrt() Returns the square root of a number
math.tan()  Returns the tangent of a number
math.tanh() Returns the hyperbolic tangent of a number
math.trunc()    Returns the truncated integer parts of a number
```

```
Math Constants

Constant    Description
math.e  Returns Euler's number (2.7182...)
math.inf    Returns a floating-point positive infinity
math.nan    Returns a floating-point NaN (Not a Number) value
math.pi Returns PI (3.1415...)
math.tau    Returns tau (6.2831...)
```

In [ ]:

```
Python JSON
```

In [ ]:

```
JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

JSON in Python
Python has a built-in package called json, which can be used to work with JSON data.
```

In [44]:

```
import json

# some JSON:
x = '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["age"])
```

```
30
```

In [45]:

```
import json

# a Python object (dict):
x = {
  "name": "John",
  "age": 30,
  "city": "New York"
}

# convert into JSON:
y = json.dumps(x)

# the result is a JSON string:
print(y)
```

```
{"name": "John", "age": 30, "city": "New York"}
```

In [46]:

```
import json

print(json.dumps({"name": "John", "age": 30}))
print(json.dumps(["apple", "bananas"]))
print(json.dumps(("apple", "bananas")))
print(json.dumps("hello"))
print(json.dumps(42))
print(json.dumps(31.76))
print(json.dumps(True))
print(json.dumps(False))
print(json.dumps(None))
```

```
{"name": "John", "age": 30}
["apple", "bananas"]
["apple", "bananas"]
"hello"
42
31.76
true
false
null
```

In [47]:

```
import json

x = {
  "name": "John",
  "age": 30,
  "married": True,
  "divorced": False,
  "children": ("Ann","Billy"),
  "pets": None,
  "cars": [
    {"model": "BMW 230", "mpg": 27.5},
    {"model": "Ford Edge", "mpg": 24.1}
  ]
}

# convert into JSON:
y = json.dumps(x)

# the result is a JSON string:
print(y)
```

```
{"name": "John", "age": 30, "married": true, "divorced": false, "children":
["Ann", "Billy"], "pets": null, "cars": [{"model": "BMW 230", "mpg": 27.5},
{"model": "Ford Edge", "mpg": 24.1}]}
```

In [48]:

```python
import json

x = {
  "name": "John",
  "age": 30,
  "married": True,
  "divorced": False,
  "children": ("Ann","Billy"),
  "pets": None,
  "cars": [
    {"model": "BMW 230", "mpg": 27.5},
    {"model": "Ford Edge", "mpg": 24.1}
  ]
}

# use four indents to make it easier to read the result:
print(json.dumps(x, indent=4))
```

```
{
    "name": "John",
    "age": 30,
    "married": true,
    "divorced": false,
    "children": [
        "Ann",
        "Billy"
    ],
    "pets": null,
    "cars": [
        {
            "model": "BMW 230",
            "mpg": 27.5
        },
        {
            "model": "Ford Edge",
            "mpg": 24.1
        }
    ]
}
```

In [49]:

```
import json

x = {
  "name": "John",
  "age": 30,
  "married": True,
  "divorced": False,
  "children": ("Ann","Billy"),
  "pets": None,
  "cars": [
    {"model": "BMW 230", "mpg": 27.5},
    {"model": "Ford Edge", "mpg": 24.1}
  ]
}

# use . and a space to separate objects, and a space, a = and a space to separate keys from
print(json.dumps(x, indent=4, separators=(". ", " = ")))
```

```
{
    "name" = "John".
    "age" = 30.
    "married" = true.
    "divorced" = false.
    "children" = [
        "Ann".
        "Billy"
    ].
    "pets" = null.
    "cars" = [
        {
            "model" = "BMW 230".
            "mpg" = 27.5
        }.
        {
            "model" = "Ford Edge".
            "mpg" = 24.1
        }
    ]
}
```

In [50]:

```
import json

x = {
  "name": "John",
  "age": 30,
  "married": True,
  "divorced": False,
  "children": ("Ann","Billy"),
  "pets": None,
  "cars": [
    {"model": "BMW 230", "mpg": 27.5},
    {"model": "Ford Edge", "mpg": 24.1}
  ]
}

# sort the result alphabetically by keys:
print(json.dumps(x, indent=4, sort_keys=True))
```

```
{
    "age": 30,
    "cars": [
        {
            "model": "BMW 230",
            "mpg": 27.5
        },
        {
            "model": "Ford Edge",
            "mpg": 24.1
        }
    ],
    "children": [
        "Ann",
        "Billy"
    ],
    "divorced": false,
    "married": true,
    "name": "John",
    "pets": null
}
```