Modules

Import Python Modules

The Python **import** statement can be used to import Python modules from other files. Modules can be imported in three different ways: `import module`, `from module import functions`, or `from module import *`. `from module import *` is discouraged, as it can lead to a cluttered local namespace and can make the namespace unclear.

```
# Three different ways to import modules:
# First way
import module
module.function()

# Second way
from module import function
function()

# Third way
from module import *
function()
```

Module importing

In Python, you can import and use the content of another file using `import filename`, provided that it is in the same folder as the current file you are writing.

```
# file1 content
# def f1_function():
#         return "Hello World"

# file2
import file1

# Now we can use f1_function, because we imported file1
f1_function()
```

Aliasing with ‘as’ keyword

In Python, the `as` keyword can be used to give an alternative name as an alias for a Python module or function.

```
# Aliasing matplotlib.pyplot as plt
from matplotlib import pyplot as plt
plt.plot(x, y)

# Aliasing calendar as c
import calendar as c
print(c.month_name[1])
```

Date and Time in Python

Python provides a module named `datetime` to deal with dates and times.
It allows you to set `date` , `time` or both `date` and `time` using the `date()` , `time()` and `datetime()` functions respectively, after importing the `datetime` module .

```
import datetime
feb_16_2019 = datetime.date(year=2019, month=2, day=16)
feb_16_2019 = datetime.date(2019, 2, 16)
print(feb_16_2019) #2019-02-16

time_13_48min_5sec = datetime.time(hour=13, minute=48, second=5)
time_13_48min_5sec = datetime.time(13, 48, 5)
print(time_13_48min_5sec) #13:48:05

timestamp= datetime.datetime(year=2019, month=2, day=16, hour=13, minute=48, second=5)
timestamp = datetime.datetime(2019, 2, 16, 13, 48, 5)
print (timestamp) #2019-01-02 13:48:05
```

`random.randint()` and `random.choice()`

In Python, the `random` module offers methods to simulate non-deterministic behavior in selecting a random number from a range and choosing a random item from a list.
The `randint()` method provides a uniform random selection from a range of integers. The `choice()` method provides a uniform selection of a random element from a sequence.

```
# Returns a random integer N in a given range, such that start <= N <= end
# random.randint(start, end)
r1 = random.randint(0, 10)
print(r1) # Random integer where 0 <= r1 <= 10

# Prints a random element from a sequence
seq = ["a", "b", "c", "d", "e"]
r2 = random.choice(seq)
print(r2) # Random element in the sequence
```