Python UUID Module to Generat...
+ Add tags
Share · Manage Highlights

Posted In

Python · Python Random

Tweet · F share · in share · P Pin

Python Random Data

- Random Seed
- Random Shuffle
- Get Random Float Numbers
- Generate Random String

chrome enterprise
Download & boost productivity
Easy, centralized management for your enterprise. Manage from the cloud or on-prem.
AD INTELLIPAAT.COM

Home » Python » Random » Python UUID Module to Generate Universally Unique Identifiers

Python UUID Module to Generate Universally Unique Identifiers

Updated on: March 9, 2021 | + Leave a Comment

In this article, we will learn how to use the **Python UUID module to generate the universally unique identifiers**. There are various versions of UUIDs. We will see each one by one with examples.

Goals of this article: –

- How to generate a version 1, 3, 4, and 5 UUIDs as specified in RFC 4122
- Why and When to use UUID
- Generate version 1 UUID using MAC address, sequence number, and the current time.
- Get a cryptographically secure random UUID of version 4
- Generate a UUID 3 and 5 based Name and cryptographic hash values.
- Understand the Structure of UUID
- Convert UUID to its String representation.
- Convert the String representation of UUID to valid UUID instance.
- Generate a reproducible UUID using a seed value
- Extract UUID attributes from UUID
- finally, What is safe and unsafe UUID

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

[su_note note_color="#fcfcfc" text_color="#222"]

Further reading

- Solve the Python Random data generation Exercise and Random data generation Quiz
- Also, read Working with random data in Python (Complete Guide)

[/su_note]

What is UUID

UUID is a Universally Unique Identifier. You can also call it as GUID, i.e., Globally Unique Identifier. However, What is it? Let's understand this in brief.

A UUID is 128 bits long number or ID to uniquely identify the documents, Users, resources or information in computer systems.

- UUID can guarantee the uniqueness of Identifiers across space and time. when we talk about space and time means when UUID generated according to the standard then the identifier does not duplicate one that has already been created or will be created to identify something else.
- Therefore UUID is useful where a unique value is necessary.

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

Do you want to learn more about what is UUID? then refer to this Wikipedia Link.

Python UUID module implemented as per **RFC 4122**. RFC 4122 is a standard and Copyright (C) of The Internet Society. **R**FC 4122 specification includes all the details and algorithms to generate the Unique Identifiers of all the versions. RFC 4122 document specifies three algorithms to generate UUIDs.

Hence using Python UUID module, you can generate versions 1, 3, 4, and 5 UUIDs. UUID generated using this module is immutable.

Python UUID module supports the following versions of UUIDs.

- UUID1 – Generate UUID using a Host MAC address, sequence number and the current time. This version uses the IEEE 802 MAC addresses.
- UUID3 and UUID 5 uses cryptographic hashing and application-provided text strings to generate UUID. UUID 3 uses MD5 hashing, and UUID 5 uses SHA-1 hashing.
- UUID4 uses pseudo-random number generators to generate UUID.

Now, Let see the simple example to get the universally unique Id.

```
import uuid

# make a UUID based on the host address and current time
uuidOne = uuid.uuid1()
print ("Printing my First UUID of version 1")
print(uuidOne)
```

Run

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

Output:

```
Printing my First UUID of version 1
a9e86162-d472-11e8-b36c-ccaf789d94a0
```

Structure of UUID

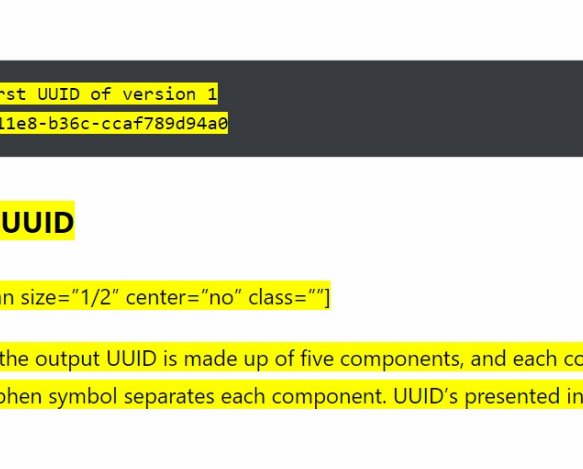
[su_row][su_column size="1/2" center="no" class=""]

As you can see in the output UUID is made up of five components, and each component has a fixed length. A hyphen symbol separates each component. UUID's presented in the **format "8-4-4-4-12"**.

The formal definition of the UUID string representation is as follows.

[/su_column]

[su_column size="1/2" center="no" class=""]


Structure Of UUID
b1d0cac0-d50d-11e8-b57b-ccaf789d94a0
time_low
time_mid
time_hi_and_version
clock_seq_hi_and_reserved_And_clock_seq_low
Node (MAC Address)
PYnative

[/su_column][/su_row]

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

UUID = **time_low** "-" **time_mid** "-"**time_high_and_version** " "**clock_seq_and_reserved_And_clock_seq_low**"-" **Node**.

Let's understand Why and When to use UUID in our application.

Why and When to use UUID

Note: When to use UUID is depends on the situation, use case, conditions, and complexity.

- To Generate unique Uniform Resource Names. UUIDs are of a fixed size (128 bits) which is reasonably small compared to other alternatives. As you know UUID is unique and persistent, it is an excellent choice for Uniform Resource Names.
- Generating UUID doesn't require a registration process.
- We can even use UUID as a transaction ID.
- Notable uses in cryptographic applications.

In Web Application

- UUID's are also handy for generating the **unique session id** to help state management.
- To generate a User ID. If you are using auto-increment values to generate user ids Its very simple and easily guessed. People can use an integer value to guess and try to access user using user Id. However, when you use UUID, it is **difficult to guess because UUID not created in any sequential format**, so it is tough to guess its sequence.

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

In Database System

- UUID has a significant advantage because **UUID is environment independent**. i.e., UUID generated on any machine using any application is universally unique.
- As most of the applications are depends on the underlying database server to generate a unique or primary key. What if we want to change the database in which the key generation is different. In such a case, a good option is to use UUID in your application to generate a unique database key.
- Also, **UUID is good for a distributed environment**. We can have one table split and placed on multiple physical database servers. If we have an autoincrement key, we have to develop a suitable algorithm to manage that.
- Also, **UUID is a real value not a pseudo value like a number in an SQL table**. Marek Sirkovský medium post describes when to use UUID and when to use other approaches.

When considering the above scenarios, indeed UUID approach is much more unique and universal to generate database Keys. Because **Auto-increment isn't suitable for the distributed system**, Most of the database servers including MS SQL Server, MySQL or Oracle, and much more use UUID to generate database keys To identify resources or information uniquely.

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

Here are some StackOverflow questions which talk more about this in detail.

- https://stackoverflow.com/questions/45399/advantages-and-disadvantages-of-guid-uuid-database-keys
- https://stackoverflow.com/questions/9377100/when-is-it-appropriate-to-use-uuids-for-a-web-project

Finally, let's see how to use the UUID module and its functions now.

UUID 1 to Generate a unique ID using MAC Address

The `uuid.uuid1()` function is used to generate a UUID from the host ID, sequence number, and the current time. It uses the MAC address of a host as a source of uniqueness.

The syntax of uuid1()

```
uuid.uuid1(node=None, clock_seq=None)
```

- The **node and clock_seq are optional arguments**.
- The node is the hardware address, which is a **48-bit** positive integer. If the node not given, then **`uuid.getnode()`** function is used to obtain the Universally administered MAC addresses of a current host.
- If clock_seq is given, it used as the sequence number. Otherwise, a random 14-bit sequence number is chosen.

AD tableau Excel + Tableau: A Beautiful Partnership GET THE WHITEPAPER

Example to generate a unique ID for Host using MAC Address.

```
import uuid
# Generate a UUID from a host ID, sequence number, and the current time
uuidOne = uuid.uuid1()
print("UUID of version one", uuidOne)
```

Run

Output:

```
UUID of version one 5b6d0be2-d47f-11e8-9f9d-ccaf789d94a0
```

[su_note note_color="#dcf4f1" text_color="#090404"]

Note: uuid1 is not safe it has privacy concerns because it shows the computer's network address in UUID.

[/su_note]

Example to generate a unique ID for Host using node and clock sequence

Each computer has a different MAC address so on each computer you will get different Id. Let's simulate this by setting explicit node IDs to simulate running on different hosts.

AD Data Analytics for Business IIT Madras CCE & Intellipaat Open

```
import uuid

# Generate a UUID using a clock sequence and node

print("UUID of version one")
clock_seq = 4115

for node in [0xccaf789d94a0, 0xadaf456d94a0]:
    print(uuid.uuid1(node, clock_seq))
```

Run

Output:

```
UUID of version one
55da37d0-d481-11e8-9013-ccaf789d94a0
55da37d1-d481-11e8-9013-adaf456d94a0
```

uuid.getnode()

To generate UUID of version 1 we need a hardware address, i.e., MAC address. It is a 48-bit positive integer.

AD chrome enterprise Download & boost productivity Easy, centralized management for your enterprise. Manage from the cloud or on-prem. Learn More

- The **`uuid.getnode()`** function is used to get the MAC address of a network interface. If the machine has more than one network interface universally administered MAC addresses are returned instead of over locally administered MAC addresses. administered MAC addresses guaranteed to be globally unique
- if **getnode() function fails** to get MAC address it returns the random 48-bit number with the multicast bit as recommended in RFC 4122.

Example: –

```
import uuid

# Get the hardware address as a 48-bit positive integer
print("MAC address integer format", uuid.getnode())
print("MAC address Hex format", hex(uuid.getnode()))
```

Run

Output:

```
MAC address integer format 225054014936224
MAC address Hex format 0xccaf789d94a0
```

AD Intellipaat.com Advance Data Analytics Program OPEN

UUID 4 to generate a random UUID

The UUID generated using a uuid4() function is created using a truly Random or Pseudo-Random generator.

Let see the example now.

```
import uuid

for i in range(2):
    uuidFour = uuid.uuid4()
    print("uuid of version four", uuidFour)
```

Run

Output:

```
uuid of version four 0056a369-4618-43a4-ad88-e7c371bf5582
uuid of version four e5e9394c-daed-498e-b9f3-69228b44fbfa
```

When should one use uuid1 and uuid4 in python?

AD tableau Excel + Tableau: A Beautiful Partnership GET THE WHITEPAPER

uuid1() is guaranteed not to produce any collisions. You can create duplicates UUIDs by creating more 16384 uuid1 in less than 100ns. Don't use uuid1 when you don't want to make the MAC address of your machine visible.

UUID4() uses the cryptographically secure random number generator to generate UUID.

uuid4() generates a random UUID. The chance of a collision is small. When UUIDs require to generate on separate machines, or you want to generate a secure UUIDs use UUID4().

furthermore, this excellent answer by Bob Aman on StackOverflow explained this in detail.

UUID 3 and UUID 5 to Create a Name-Based UUID

Version 3 or 5 UUID meant for generating UUIDs from "names." we can use name and namespace to create a series of unique UUIDs. In simple words version, 3 and 5 UUIDs is nothing but hashing namespace identifier with a name.

The `uuid.uuid3(namespace, name)` generate a UUID based on the **MD5** hash of a namespace identifier (which is a UUID) and a string.

Similarly, the `uuid.uuid5(namespace, name)` generate a UUID based on the **SHA-1** hashing technique of a namespace identifier (which is a UUID) and a name.

AD Data Analytics for Business IIT Madras CCE & Intellipaat Open

The **UUID module defines the following namespace identifiers to use with uuid3() or uuid5()**.

- **UUID.NAMESPACE_DNS** means a fully qualified domain name. For example, https://pynative.com.
- **UUID.NAMESPACE_URL** When this namespace is specified, It means it is a URL.
- **UUID.NAMESPACE_OID** When this namespace is specified, the name string is an ISO OID.
- **UUID.NAMESPACE_X500** When this namespace is specified, the name string is an X.500 DN in DER or a text output format.

Let see the examples now. Generate a UUID 3 and UUID 5 using the different hostname and namespace.

```
import uuid

hostNames = ['pynative.com', 'stackoverflow.com']

for host in hostNames:
    print('Generate uuid of version 3 using name as',host,' and namespace as uuid.NAM
    print(uuid.uuid3(uuid.NAMESPACE_DNS, host))
    print('Generate uuid of version 5 using name as', host, ' and namespace as uuid.N
    print(uuid.uuid5(uuid.NAMESPACE_DNS, host))
    print()
```

Run

AD tableau Excel + Tableau: A Beautiful Partnership GET THE WHITEPAPER

Output:

```
Generate uuid of version 3 using name as pynative.com  and namespace as uuid.NAMESPACE
6ddc8513-dc7b-3b37-b21b-a1ca9440fe14
Generate uuid of version 5 using name as pynative.com  and namespace as uuid.NAMESPACE
8d6a1314-170a-559c-afe7-b68d1d7ee9ac

Generate uuid of version 3 using name as stackoverflow.com  and namespace as uuid.NAM
6d079ab3-a985-3dc7-8086-3dc32dc08cb9
Generate uuid of version 5 using name as stackoverflow.com  and namespace as uuid.NAM
cd84c40a-6019-50c7-87f7-178668ab9c8b
```

Example to generate a UUID 3 and UUID 5 using a different namespace.

```
import uuid

nameSpaces = [uuid.NAMESPACE_DNS, uuid.NAMESPACE_URL, uuid.NAMESPACE_OID, uuid.NAMESP
hostName = 'pynative.com'
print("Generate uuid using namespace")

for namespace in nameSpaces:
    print('uuid 3 is', uuid.uuid3(namespace, hostName))
    print('uuid 5 is', uuid.uuid5(namespace, hostName))
    print()
```

Run

AD EXPO 2020 DUBAI UAE Step into the world's largest 360° projection dome at Al Wasl Plaza. Buy your tickets. VISIT BEFORE 31 MARCH 2022

Output:

```
Generate uuid using namespace
uuid 3 is 6ddc8513-dc7b-3b37-b21b-a1ca9440fe14
uuid 5 is 8d6a1314-170a-559c-afe7-b68d1d7ee9ac

uuid 3 is 5dcfef3e-bcc9-38bc-b989-4a7516a05974
uuid 5 is 3a4a6c31-8d6a-5583-8497-d2ed90b1f13a

uuid 3 is 84d9730f-330f-3634-9542-4acfcdcd6c60
uuid 5 is 899f3d4b-6095-5ee6-9805-68e0c51dcb39

uuid 3 is b140fa3b-983a-3efe-85ef-92f07d5e09a0
uuid 5 is 73b723ef-5c5e-5eb4-8fcc-aabb5c4e7803
```

The Behaviour of uuid3 and UUID 5: –

- The UUIDs generated at a different times using the same namespace and same name are equal.
- The unique Ids generated from two different names in the same namespace are different.
- The UUIDs generated from the same name in two different namespaces are different.

AD EXPO 2020 DUBAI UAE The future isn't far away, it even says Hi! Buy your tickets. VISIT BEFORE 31 MARCH 2022

Example: –

```
import uuid

print('Generate uuid of version 3 using name as pynative.com and namespace as uuid.NA
print(uuid.uuid3(uuid.NAMESPACE_DNS, "pynative.com"))

print('Generate uuid of version 3 using name as pynative.com and namespace as uuid.NA
print(uuid.uuid3(uuid.NAMESPACE_DNS, "pynative.com"))
```

Run

You should get the same UUID both the times.

```
Generate uuid of version 3 using name as pynative.com and namespace as uuid.NAMESPACE
6ddc8513-dc7b-3b37-b21b-a1ca9440fe14

Generate uuid of version 3 using name as pynative.com and namespace as uuid.NAMESPACE
6ddc8513-dc7b-3b37-b21b-a1ca9440fe14
```

Extract UUID attributes read-only attributes

AD Intellipaat.com Advance Data Analytics Program OPEN

The internal representation of a UUID is a specific sequence of bits in memory, as described in RFC4211. It is necessary to convert the bit sequence to a string representation to represent UUID in string format.

UUID module provides the various read-only argument to access the value of each component of the UUID object. You can extract the values from UUID so we can use this value for a different purpose. For example, You want to Extract the time from a UUID version1 in python.

UUID Read-only Attribute includes the following: –

- **UUID.bytes:** The UUID as a 16-byte string (containing the six integer fields in big-endian byte order).
- **UUID.bytes_le:** It is a 16-byte string that consists of a time_low, time_mid, and time_hi_version.
- **UUID.fields:** A tuple of the six integer fields of the UUID, which are also available as six individual attributes and two derived attributes: **UUID.fields has the following fields**.

[su_table]

Field	Meaning
time_low	the first 32 bits of the UUID
time_mid	the next 16 bits of the UUID
time_hi_version	the next 16 bits of the UUID
clock_seq_hi_variant	the next 8 bits of the UUID
clock_seq_low	the next 8 bits of the UUID
node	the last 48 bits of the UUID
time	the 60-bit timestamp
clock_seq	the 14-bit sequence number

AD EXPO 2020 DUBAI UAE Take a journey to 192 countries. Buy your tickets. VISIT BEFORE 31 MARCH 2022

[/su_table]

- **UUID.hex:** The UUID as a 32-character hexadecimal string.
- **UUID.int:** The integer representation of a UUID as a 128-bit integer.
- **UUID.urn:** The UUID as a uniform resource name.
- **UUID.variant:** The UUID variant, which determines the internal layout of the UUID. This will be one of the constants RESERVED_NCS, RFC_4122, RESERVED_MICROSOFT, or RESERVED_FUTURE.
- **UUID.version:** the version of UUID. anything between 1, 4, 3, and 5.
- **UUID.is_safe:** To get to know that UUID generation is safe or not. we will see this in the latter section of the article.

Let see how to access these read-only attribute of UUID.

```
import uuid

UUID = uuid.uuid1()

print("UUID is ", UUID)
print("UUID Type is ",type(UUID))
print('UUID.bytes   :', UUID.bytes)
print('UUID.bytes_le :', UUID.bytes_le)
print('UUID.hex     :', UUID.hex)
print('UUID.int     :', UUID.int)
print('UUID.urn     :', UUID.urn)
print('UUID.variant :', UUID.variant)
print('UUID.version :', UUID.version)
print('UUID.fields  :', UUID.fields)
print("Prining each field seperately")
print('UUID.time_low            : ', UUID.time_low)
print('UUID.time_mid            : ', UUID.time_mid)
print('UUID.time_hi_version     : ', UUID.time_hi_version)
print('UUID.clock_seq_hi_variant: ', UUID.clock_seq_hi_variant)
print('UUID.clock_seq_low       : ', UUID.clock_seq_low)
print('UUID.node                : ', UUID.node)
print('UUID.time                : ', UUID.time)
print('UUID.clock_seq           : ', UUID.clock_seq)
print('UUID.SafeUUID           : ', UUID.is_safe)
```

Run

AD Introducing the Antminer S1... Globalstoresltd Visit Site

Output:

```
UUID is  3b212454-d494-11e8-92f4-ccaf789d94a0
UUID Type is  <class 'uuid.UUID'>
UUID.bytes   : b';!$T\xd4\x94\x11\xe8\x92\xf4\xcc\xaf\x9d\x94\xa0'
UUID.hex     : 3b212454d49411e892f4ccaf789d94a0
UUID.int     : 78596534435342896145298010144107238560
UUID.urn     : urn:uuid:3b212454-d494-11e8-92f4-ccaf789d94a0
UUID.variant : specified in RFC 4122
UUID.version : 1
UUID.fields  : (992027732, 54420, 4584, 146, 244, 225054014936224)
Prining each field seperately
UUID.time_low            :  992027732
UUID.time_mid            :  54420
UUID.time_hi_version     :  4584
UUID.clock_seq_hi_variant:  146
UUID.clock_seq_low       :  244
UUID.node                :  225054014936224
UUID.time                :  137593521747076180
UUID.clock_seq           :  4852
UUID.SafeUUID           :  SafeUUID.unknown
```

UUID to String and String to UUID in Python

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

When we call a `uuid.uuid1` or any other version of UUID you will get an instance of UUID class. When we want UUID in string format for comparison, manipulation or maybe for any reason we can get its string representation using a str class. Let see how do change a UUID to a string.

```
import uuid

UUID1 = uuid.uuid1()
print("UUID of version 1 is ", UUID1)
# convert a UUID to a string of hex digits in standard form
print("UUID of version 1 in String format", str(UUID1))
```

Run

Output:

```
UUID of version 1 is  018c168c-d509-11e8-b096-ccaf789d94a0
UUID of version 1 in String format 018c168c-d509-11e8-b096-ccaf789d94a0
```

You can also get UUID without dashes. You need to use the string's replace method to remove the dashes from the string for example.

```
import uuid

UUID1 = uuid.uuid1()
print("UUID of version 1 is ", UUID1)
# convert a UUID to a string of hex digits in standard form
uuidString = str(UUID1).replace("-", "")
print("UUID of version 1 in String removing dashes", uuidString)
```

Run

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

Output:

```
UUID of version 1 is  c7c9de0a-d676-11e8-8d62-ccaf789d94a0
UUID of version 1 in String removing dashes c7c9de0ad67611e88d62ccaf789d94a0
```

Now let's see how to create UUID from String

Assume you received UUID in string format. Now in your application, you need to convert it in the UUID class instance for some operation. **Let see how to use a `uuid.UUID` class to generate a valid UUID from String.**

The `uuid.UUID` is a class which returns a UUID instance when we pass the argument of UUID. Let assume that you have this UUID in string format `{018c168c-d509-11e8-b096-ccaf789d94a0}`.

```
import uuid

UUIDStrings = ["{55da37d1-d481-11e8-9013-adaf456d94a0}", "018c168c-d509-11e8-b096-cca

for string in UUIDStrings:
    # make a UUID from a string of hex digits (braces and hyphens ignored)
    myUUID = uuid.UUID(string)
    print("My UUID is", myUUID)
    print("My UUID time component is", myUUID.time)
    print()
```

Run

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

Output:

```
My UUID is 55da37d1-d481-11e8-9013-adaf456d94a0
My UUID time component is 137593440591034321

My UUID is 018c168c-d509-11e8-b096-ccaf789d94a0
My UUID time component is 137594023292180108

My UUID is e5e9394c-daed-498e-b9f3-69228b44fbfa
My UUID time component is 688728508333635916
```

Generate Reproducible UUIDs

How to generate a random UUID which is reproducible with seed or with existing UUID attribute in Python?

To Generate the same UUID anytime, you need a seed value. UUID has various attributes as we already discussed above. Using any of its attribute values, we can reproduce the same UUID. Alternatively, you can use seed value to generate the same UUID. Let see both ways.

You can create a UUID instance by passing argument value to `uuid.UUID` class. For example, you can create a UUID instance from the following inputs: –

AD EXPO 2020 DUBAI UAE Step into the world's largest 360° projection dome at Al Wasl Plaza. Buy your tickets. VISIT BEFORE 31 MARCH 2022

- UUID.String
- UUID.Bytes
- UUID.Bytes_le
- UUID.Fields
- UUID.int

If you have any of the above values, you can generate a UUID. Let see this with an example. I have used all these values of the same UUID. therefore, the result must be the same UUID in all cases.

```
import uuid

print("Generating UUID from int")
UUID_x = uuid.UUID(int=236357465324988601727440242910546465952)
print("UUID is", UUID_x)

print("UUID from URN")
UUID_x1 = uuid.UUID('urn:uuid:b1d0cac0-d50d-11e8-b57b-ccaf789d94a0')
print("UUID is", UUID_x1)

print("UUID from bytes")
UUID_x2 = uuid.UUID(bytes=b'\xb1\xd0\xca\xc0\xd5\r\x11\xe8\xb5{\xcc\xaf\x9d\x94\xa0'
print("UUID is", UUID_x2)

print("UUID from bytes_len")
UUID_x3 = uuid.UUID(bytes_le=b'\xc0\xca\xd0\xb1\r\xd5\xe8\x11\xb5{\xcc\xaf\x9d\x94\x
print("UUID is", UUID_x3)

print("UUID from fields")
UUID_x4 = uuid.UUID(fields=(2983250624, 54541, 4584, 181, 123, 225054014936224))
print("UUID is", UUID_x4)
```

Run

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

Output:

```
Generating UUID from int
UUID is b1d0cac0-d50d-11e8-b57b-ccaf789d94a0
UUID from URN
UUID is b1d0cac0-d50d-11e8-b57b-ccaf789d94a0
UUID from bytes
UUID is b1d0cac0-d50d-11e8-b57b-ccaf789d94a0
UUID from bytes_len
UUID is b1d0cac0-d50d-11e8-b57b-ccaf789d94a0
UUID from fields
UUID is b1d0cac0-d50d-11e8-b57b-ccaf789d94a0
```

Reproduce UUID with seed

For testing purposes, we need to generate the same UUID again, and again.i have used the faker module here for a seed value.

```
import uuid
from faker import Faker

fakerObj = Faker()
fakerObj.seed(8754)
print(fakerObj.uuid4())

fakerObj.seed(8754)
print(fakerObj.uuid4())
```

Run

AD DigitalOcean mongoDB. Provision MongoDB clusters in minutes. Build scalable performant apps using DigitalOcean's fully managed, MongoDB as a service. Try for free

Output:

```
b1d0cac0-d50d-11e8-b57b-ccaf789d94a0
b1d0cac0-d50d-11e8-b57b-ccaf789d94a0
```

Safe and Unsafe UUID

You may have a privacy problem with uuid1 as it use MAC address of your host. On the other hand uuid4() creates a UUID using a random generator.

Also, uuid1 is not safe, i.e. uuid1() may or may not return a "safe." when we call safe means **UUID must be generated using synchronization methods so that no two processes can obtain the same UUID**.

Generating safe UUID depends on support from the underlying operating system.

Python 3.7 has added a new attribute to the UUID class using that we can determine UUID is safe or unsafe. Now all the instance of UUID has an **is_safe** attribute which let us know the UUID is safe or not.

AD Introducing the Antminer S1... Globalstoresltd Visit Site

uuid.SafeUUID can return the following values.

- **safe:** The UUID was generated in a multiprocessing-safe way.
- **unsafe:** The UUID was not generated in a multiprocessing-safe way.
- **unknown:** The operating system does not provide information on whether the UUID was generated safely or not.

Let see the example I tested the following code on my windows 7 os to find UUID is safe or not. Let see what it returns.

```
import uuid

UUID1 = uuid.uuid1()
print("uuid 1 safety", UUID1.is_safe)
```

Run

Output:

```
uuid 1 safety SafeUUID.unknown
```

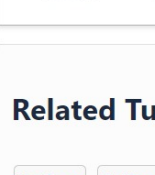
AD Script Comfort that changes with your activity. Get an adaptable sofa, with Script, a new perspective on furniture and accessories. BUY NOW. SWITCH SOFA

Filed Under: Python, Python Random

Did you find this page helpful? Let others know about it. **Sharing helps me** continue to create free Python resources.

Tweet · F share · in share · P Pin

About Vishal



Founder of PYnative.com I am a Python developer and I love to write articles to help developers. Follow me on Twitter. All the best for your future Python endeavors!

Related Tutorial Topics:

Python · Python Random

Python Exercises and Quizzes

Free coding exercises and quizzes cover Python basics, data structure, data analytics, and more.

15+ **Topic-specific Exercises and Quizzes**
Each Exercise contains 10 questions
Each Quiz contains 12-15 MCQ

Exercises · Quizzes

Leave a Reply

your email address will NOT be published. all comments are moderated according to our comment policy.

Comment

Use <pre> tag for posting code. E.g. <pre> Your code </pre>

Name *

Email *

Post Comment

About PYnative

PYnative.com is for Python lovers. Here, You can get Tutorials, Exercises, and Quizzes to practice and **improve your Python skills**.

Explore Python

- Learn Python
- Python Basics
- Python Databases
- Python Exercises
- Python Quizzes
- Online Python Code Editor
- Python Tricks

Follow Us

To get New Python Tutorials, Exercises, and Quizzes

- Twitter
- Facebook
- Sitemap

Legal Stuff

- About Us
- Contact Us

We use cookies to improve your experience. While using PYnative, you agree to have read and accepted our Terms Of Use, Cookie Policy, and Privacy Policy.

Copyright © 2018-2021 · [pynative.com]

AN ELITE CAFEMEDIA TECH PUBLISHER