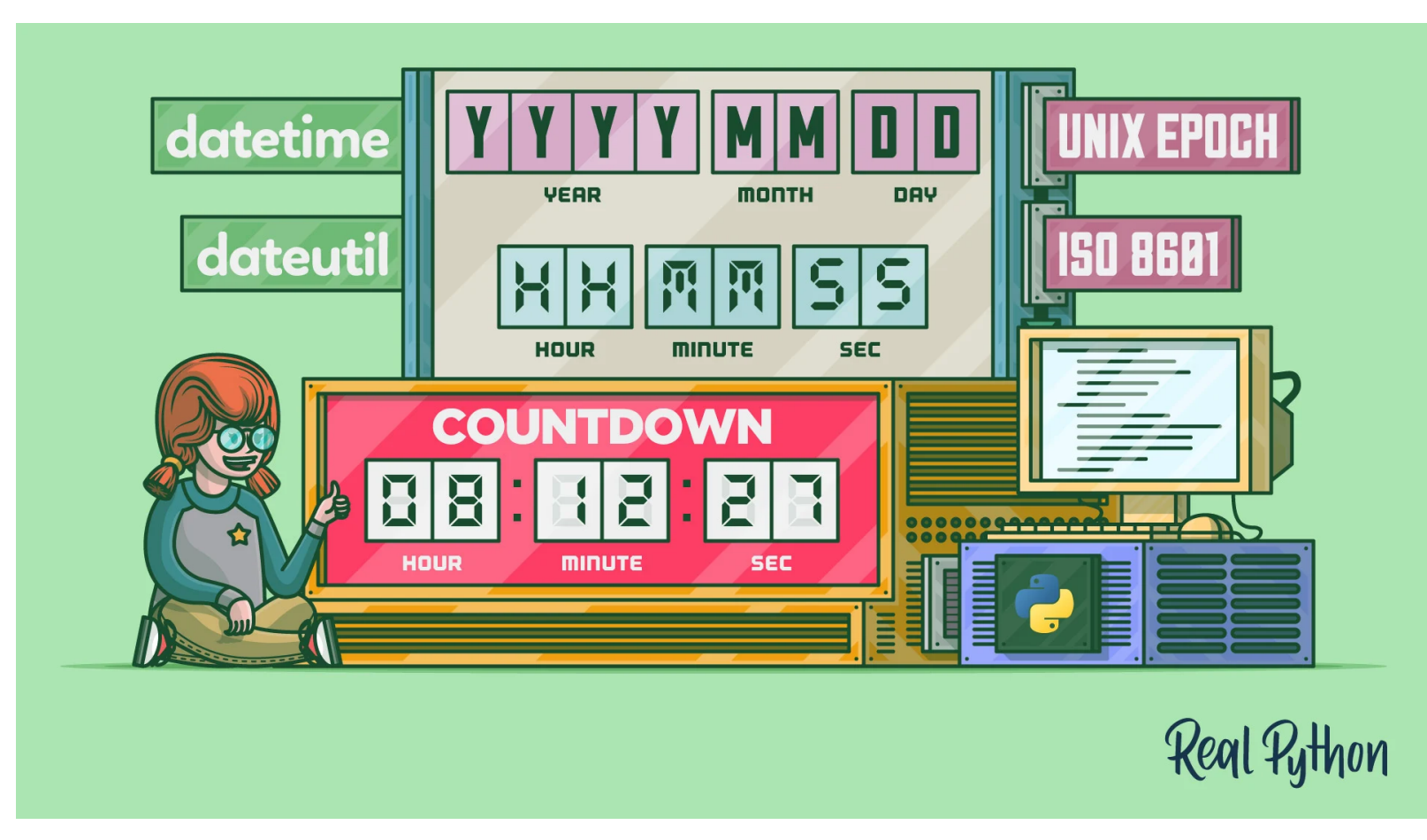
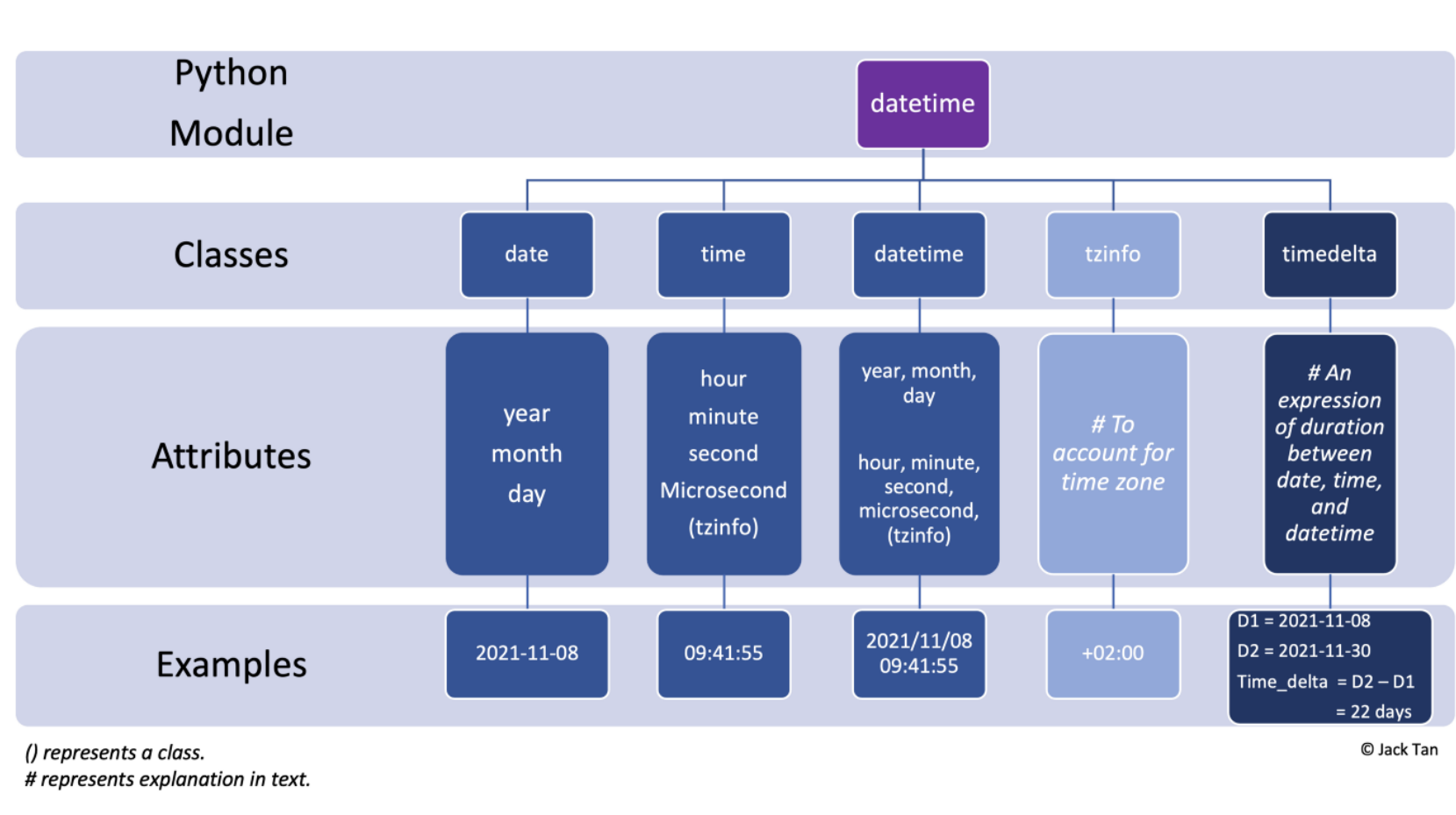
Date Time Module in Python



Date Time Module

- **Python DateTime module is provided in Python to work with dates and times**
- **In python, DateTime is an inbuilt module rather than being a primitive data type**
- **We just have to import the module to work with dates as date object.**



() represents a class.
represents explanation in text.

- **It provides three additional data types: date, time and datetime.**

Class	Discription
datetime.date	A date object represents a date, excluding time, according to the Gregorian calendar.
datetime.datetime	A datetime object represents a date and time, according to the Gregorian calendar.
datetime.time	A time object represents time, excluding date.
datetime.tzinfo	A base abstract class for time zone information objects.
datetime.timezone	A direct child class of the tzinfo class, in UTC (Coordinated Universal Time).
datetime.timedelta	A timedelta object represents a duration, the difference between two dates or times.

What's inside datetime ?

In [1]:
```
#importing datetime module
import datetime
print(dir(datetime))
```
```
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__packag
e__', '__spec__', 'date', 'datetime', 'datetime_CAPI', 'sys', 'time', 'timedelta', 'timezone', 'tzinfo']
```

date ( )

`datetime.date(year: int, month: int, day: int)`

In [2]:
```
from datetime import date

userdate = date(year=2020, month=1, day=31)
#datetime.date(2020, 1, 31)

print("User Date : ",userdate, "Type :", type(userdate))
print()
print("User Year  : ",userdate.year)
print()
print("User Month : ",userdate.month)
print()
print("User Day   : ",userdate.day)
```
```
User Date :  2020-01-31 Type : <class 'datetime.date'>

User Year  :  2020

User Month :  1

User Day   :  31
```

time ( )

`datetime.time(hour: int, minute: int, second: int)`

In [3]:
```
from datetime import time

usertime = time(hour=13, minute=14, second=31)
#datetime.time(13, 14, 31)

print("User Time : ",usertime, "Type :", type(usertime ))
print()
print("User Hour   : ",usertime.hour)
print()
print("User Minute : ",usertime.minute)
print()
print("User Second : ",usertime.second)

#Additionally Time Takes microsecond and tzinfo attributes
print()
print("User Micro Second : ",usertime.microsecond)
```
```
User Time :  13:14:31 Type : <class 'datetime.time'>

User Hour   :  13

User Minute :  14

User Second :  31

User Micro Second :  0
```

In [4]:
```
#Replace
import datetime

time1 = datetime.time()
print('Without passing any attribute: ', time1)
print()
time2 = datetime.time(hour=12, minute=55, second=50)
print('By passing hour, minute and second attributes: ', time2)
print()
time1 = time1.replace(hour=17)
print('By replacing the hour attribute in time1: ', time1)
print()
time2 = time2.replace(minute=00)
print('By replacing the minute attribute in time2: ', time2)
```
```
Without passing any attribute:  00:00:00

By passing hour, minute and second attributes:  12:55:50

By replacing the hour attribute in time1:  17:00:00

By replacing the minute attribute in time2:  12:00:50
```

datetime ( )

`datetime.datetime(year, month, day, hour, minute, second)`

In [5]:
```
from datetime import datetime

userdatetime = datetime(year=2020, month=1, day=31, hour=13, minute=14, second=31)
#datetime.datetime(2020, 1, 31, 13, 14, 31)

print("User Date Time : ",userdatetime, "Type :", type(userdatetime))
print()
print("User Year   : ",userdatetime.year)
print()
print("User Month  : ",userdatetime.month)
print()
print("User Day    : ",userdatetime.day)
print()
print("User Hour   : ",userdatetime.hour)
print()
print("User Minute : ",userdatetime.minute)
```
```
User Date Time :  2020-01-31 13:14:31 Type : <class 'datetime.datetime'>

User Year   :  2020

User Month  :  1

User Day    :  31

User Hour   :  13

User Minute :  14
```

now ( ) and today ( )

now and today methods return a datetime object with system's exact day and time

In [6]:
```
from datetime import datetime
now = datetime.now()
print("Present Date Time : ",now, "Type :", type(now))
```
```
Present Date Time :  2022-09-14 10:43:10.685726 Type : <class 'datetime.datetime'>
```

In [7]:
```
print("Present Date: ",now.date())
print()
print("Present Time: ",now.time())
print()
print("Current Year: ",now.year)
print()
print("Current Hour: ",now.hour)
```
```
Present Date:  2022-09-14

Present Time:  10:43:10.685726

Current Year:  2022

Current Hour:  10
```

Additionally, now can take a timezone object as an optional parameter

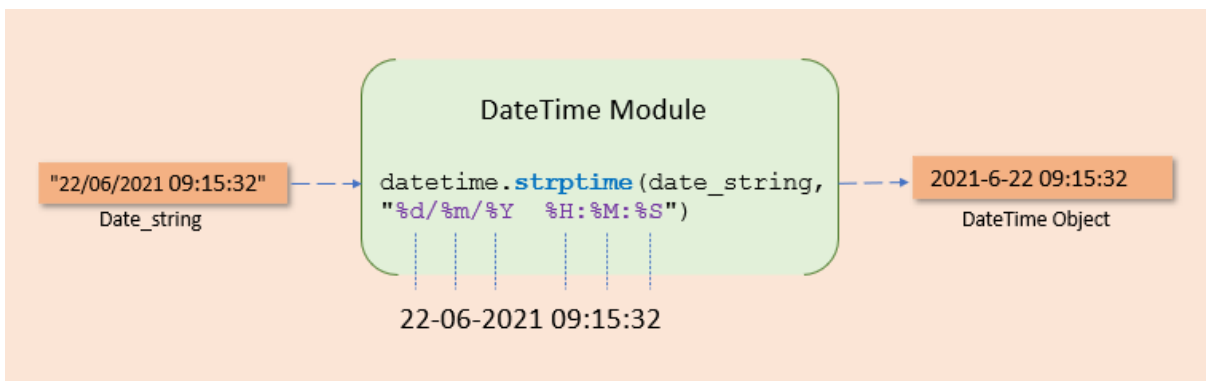
In [8]:
```
from datetime import datetime, timezone
datetime.now(timezone.utc)
```
Out[8]: `datetime.datetime(2022, 9, 14, 5, 14, 0, 704346, tzinfo=datetime.timezone.utc)`

In [9]:
```
#today
today = date.today()
print(today)
```
```
2022-09-14
```

In [10]:
```
#combine
current_time = time(now.hour, now.minute, now.second)
combined = datetime.combine(today, current_time)
print(combined)
```
```
2022-09-14 10:43:10
```

strftime ( ) and strptime ( )

We can easily transform between strings and datetime objects with the strftime and strptime methods.



strftime ( ) : Datetime to String Conversion

- **Allow us to create human formatted strings out of a Python datetime object**

In [11]:
```
#lets create a date time object
from datetime import datetime
now = datetime.now()
print("Present Date Time : ",now, "Type :", type(now))
```
```
Present Date Time :  2022-09-14 10:46:41.408111 Type : <class 'datetime.datetime'>
```

In [14]:
```
#Conert Date Time to String
Conversion1 = now.strftime("%d-%b-%Y")
print("Conversion1 : ",Conversion1, "Type :", type(Conversion1))
print()

Conversion2 = now.strftime("%d-%m-%Y")
print("Conversion2 : ",Conversion2, "Type :", type(Conversion2))
print()

Conversion3 = now.strftime("%d-%m-%y")
print("Conversion3 : ",Conversion3, "Type :", type(Conversion3))
print()

Conversion4 = now.strftime("%m/%d/%Y")
print("Conversion4 : ",Conversion4, "Type :", type(Conversion4))
print()

Conversion5 = now.strftime("%b/%d/%Y - %H:%M:%S")
print("Conversion5 : ",Conversion5, "Type :", type(Conversion5))
```
```
Conversion1 :  14-Sep-2022 Type : <class 'str'>

Conversion2 :  14-09-2022 Type : <class 'str'>

Conversion3 :  14-09-22 Type : <class 'str'>

Conversion4 :  09/14/2022 Type : <class 'str'>

Conversion5 :  Sep/14/2022 - 10:46:41 Type : <class 'str'>
```

strptime ( ) : String to Date Time Conversion

- **The strptime method creates a datetime object from a string.**

`obj.strptime(datetime_string, format)`

In [15]:
```
from datetime import datetime

datetime_str = '12-Jul-2023'
conversion1 = datetime.strptime(datetime_str, '%d-%b-%Y')
print("Conversion1 : ",conversion1, "Type :", type(conversion1))
print()

datetime_str = 'Jul/12/2023 - 14:38:37'
format_str = '%b/%d/%Y - %H:%M:%S'
conversion2 = datetime.strptime(datetime_str, format_str )
print("Conversion2 : ",conversion2, "Type :", type(conversion2))
```
```
Conversion1 :  2023-07-12 00:00:00 Type : <class 'datetime.datetime'>

Conversion2 :  2023-07-12 14:38:37 Type : <class 'datetime.datetime'>
```

Format Codes :

Python datetime format codes			
Date Example: 2021-09-05 (Sep 5, 2021)		Time Example: 09:41:55	
%y	Year without century as a zero-padded decimal number, **21**	%H	Hour (24-hour clock) as a zero-padded decimal number, **09**
%Y	Year with century as a decimal number, **2021**	%-H	Hour (24-hour clock) as a decimal number (Platform specific), **9**
%m	Month as a zero-padded decimal number, **09**	%M	Minute as a zero-padded decimal number, **41**
%-m	Month as a decimal number (Platform specific), **9**	%-M	Minute as a decimal number (Platform specific), **41**
%d	Day of the month as a zero-padded decimal number, **05**	%S	Second as a zero-padded decimal number, **55**
%-d	Day of the month as a decimal number (Platform specific), **5**	%-S	Second as a decimal number (Platform specific), **55**
%b	Month as locale's abbreviated name, **Sep**		
%B	Month as locale's full name, **September**		

© Jack Tan

In [17]:
```
#lets see format codes
todaydatetime =datetime.now()

print("Present date and time: ", todaydatetime)
print()
print("Fetched out year from current date: ", todaydatetime.strftime('%Y'))
print()
print("Fetched out month from current date: ", todaydatetime.strftime('%B'))
print()
print("Fetched out day from current date: ", todaydatetime.strftime('%a'))
print()
print("Formatted the time from above generated date time: ", todaydatetime.strftime('%H:%M:%S'))
print()
print("Formatted both date and time: ",todaydatetime.strftime("%d/%m/%Y %H:%M:%S"))
```
```
Present date and time:  2022-09-14 10:53:00.033021

Fetched out year from current date:  2022

Fetched out month from current date:  September

Fetched out day from current date:  Wed

Formatted the time from above generated date time:  10:53:00

Formatted both date and time:  14/09/2022 10:53:00
```

timedelta ( )

The timedelta object represents the difference between two dates or times.

In [18]:
```
from datetime import datetime

date_1 = datetime.strptime('12-Jul-2023', '%d-%b-%Y')
print('date1: ', date_1)
print()

date_2 = datetime.strptime('01-Jan-2024', '%d-%b-%Y')
print('date2: ', date_2)
print()

difference = date_2 - date_1
print(f"The difference is {difference} with type {type(difference)}")
print()
print(f"The difference is {difference.days}")
```
```
date1:  2023-07-12 00:00:00

date2:  2024-01-01 00:00:00

The difference is 173 days, 0:00:00 with type <class 'datetime.timedelta'>

The difference is 173
```

In [20]:
```
import datetime

date1= datetime.timedelta(days=1, hours=23, minutes=30)
print('date1: ', date1)
print()

date2= datetime.timedelta(days=4, hours=11, minutes=30)
print('date2: ', date2)
print()

difference = date2-date1
print(f"Difference between date2 and date1 :{difference} with type {type(difference)}")
```
```
date1:  1 day, 23:30:00

date2:  4 days, 11:30:00

Difference between date2 and date1 :2 days, 12:00:00 with type <class 'datetime.timedelta'>
```

In [21]:
```
#timedelta can add days, seconds and microseconds to a datetime object
from datetime import datetime, timedelta
now = datetime.now()
print(f"Time before adding time delta {now}")

added = now + timedelta(days=10, seconds=15)
print(f"Time after adding time delta {added}")
```
```
Time before adding time delta 2022-09-14 10:58:08.654785

Time after adding time delta 2022-09-24 10:58:23.654785
```

In [22]:
```
#And can subtract days, seconds and microseconds to a datetime object:
from datetime import datetime, timedelta
now = datetime.now()
print(f"Time before subtracting time delta {now}")

subtracted = now - timedelta(days=10, seconds=15)
print(f"Time after subtracting time delta {subtracted}")
```
```
Time before subtracting time delta 2022-09-14 10:58:32.429135

Time after subtracting time delta 2022-09-04 10:58:17.429135
```

Timezone :

- **A time zone represents the standard time which depends upon which part of the world is considered.**
- **Python datetime instances support two types of operation, naive and aware.**
- **The basic difference between them is that naive instances don't contain time zone information, whereas aware instances do.**
- **We can import third party packages like pytz or dateutil for using the time zone of different part of the world**

In [23]:
```
#Install dateutil
#pip install python-dateutil
from dateutil import tz
from datetime import datetime
```

In [24]:
```
#Local time
now = datetime.now(tz=tz.tzlocal())
print(f"Present time in {now.tzname()} is {now}")
```
```
Present time in India Standard Time is 2022-09-14 11:02:16.092992+05:30
```

In [25]:
```
#London time
London_tz = tz.gettz("Europe/London")
now = datetime.now(tz=London_tz)
print(f"Present time in {now.tzname()} is {now}")
```
```
Present time in BST is 2022-09-14 06:33:13.067687+01:00
```

In [26]:
```
#The method demonstrated here returns an aware datetime instance.
now = datetime.now(tz=tz.UTC)
print(f"Present time in {now.tzname()} is {now}")
```
```
Present time in UTC is 2022-09-14 05:33:24.084170+00:00
```

In [27]:
```
#added another module for using different time zones.
import pytz

localtime= datetime.now()
print('local time: ', localtime)
print()
utctime = datetime.now(pytz.utc)
print('UTC time zone: ', utctime)
print()
ustime = datetime.now(pytz.timezone('US/CENTRAL'))
print('US time zone: ', ustime)
```
```
local time:  2022-09-14 11:04:11.409990

UTC time zone:  2022-09-14 05:34:11.410988+00:00

US time zone:  2022-09-14 00:34:11.941933-05:00
```

How to Get Current timeStamp ? :

- **A timestamp is a sequence of characters used to find when a particular event occurred**
- **Generally giving the date and time of the day which is accurate to a small fraction of a second.**

In [31]:
```
import datetime

timestamp =datetime.datetime.now()
print('current date and time: ', timestamp)
print()
print('current timestamp', timestamp.timestamp())
```
```
current date and time  2022-09-14 11:06:20.879618

current timestamp 1663133780.879618
```

More on Date Time

In [32]:
```
#Constants
import datetime
#min - The earliest representable date, date(MINYEAR, 1, 1).
print(f"The earliest representable date {datetime.date.min}")
print()

#max - The latest representable date, date(MAXYEAR, 12, 31).
print(f"The latest representable date {datetime.date.max}")
print()

#resolution - The smallest possible difference between non-equal date objects, timedelta(days=1).
print(f"The smallest possible difference {datetime.date.resolution}")
```
```
The earliest representable date 0001-01-01

The latest representable date 9999-12-31

The smallest possible difference 1 day, 0:00:00
```

In [33]:
```
#lets take a date to see avaliable methods
userdate = datetime.date(2002, 12, 4)

#weekday() - Return the day of the week as an integer,where Monday is 0 and Sunday is 6.
print(f"Weekday of User Date is {userdate.weekday()}")
print()

#isoweekday() - Return the day of the week as an integer (According to ISO standard), where Monday is 1 and Sunday is 7.
print(f"ISO Weekday of User Date is {userdate.isoweekday()}")
print()

#isocalendar - Return a 3-tuple, (ISO year, ISO week number, ISO weekday).
print(f"ISO Calender of User Date is {userdate.isocalendar()}")
print()

#isoformat() Return a string representing the date in ISO 8601 format, 'YYYY-MM-DD'.
print(f"ISO format of User Date is {userdate.isoformat()}")
print()

#For a date d, str(d) is equivalent to d.isoformat().
print(f"String format of User Date  {userdate.__str__()}")
print()

#ctime() - Return a string representing the date
print(f"Day Representation of User Date  {userdate.ctime()}")
```
```
Weekday of User Date is 2

ISO Weekday of User Date is 3

ISO Calender of User Date is (2002, 49, 3)

ISO format of User Date is 2002-12-04

String format of User Date  2002-12-04

Day Representation of User Date  Wed Dec  4 00:00:00 2002
```

The constructor of the date class can raise a ValueError if the values passed in are invalid (out of range).

MINYEAR <= year <= MAXYEAR

1 <= month <= 12

1 <= day <= number of days in the given month and year

In [37]:
```
#feb 30
datetime.date(2019,2,30)
```
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-37-4d0e36356cf6> in <module>
      1 #feb 30
----> 2 datetime.date(20193,1,28)

ValueError: year 20193 is out of range
```

In [38]:
```
# compare dates

# importing datetime module
import datetime

# date in yyyy/mm/dd format
d1 = datetime.datetime(2018, 5, 3)
d2 = datetime.datetime(2018, 6, 1)

# Comparing the dates will return either True or False
print("d1 is greater than d2 : ", d1 > d2)
print()
print("d1 is less than d2 : ", d1 < d2)
print()
print("d1 is not equal to d2 : ", d1 != d2)
```
```
d1 is greater than d2 :  False

d1 is less than d2 :  True

d1 is not equal to d2 :  True
```

dateutil

- **The dateutil module specializes in providing an extension of features to the existing datetime module,**
- **It provides extra functionality, such as timedeltas that are expressed in units larger than a day**

In [39]:
```
# We'll need to import methods from the datetime module as a base.
from datetime import datetime , time , date
# Now, let's import some methods from the dateutil subclasses.
from dateutil.relativedelta import *
from dateutil.easter import *
from dateutil.parser import *
from dateutil.rrule import *
```

In [40]:
```
# Creating a few datetime objects to work with
NOW = datetime.now()
print("The datetime right now : ", NOW)
TODAY = date.today()
print("The date today : ", TODAY)
```
```
The datetime right now :  2022-09-14 11:14:20.556853
The date today :  2022-09-14
```

In [42]:
```
# Next month
print(NOW + relativedelta(months=+1))

# Next month, plus one week
print(NOW + relativedelta(months=+1, weeks=+1))

# Next month, plus one week, at 5 PM
print(NOW + relativedelta(months=+1, weeks=+1, hour=17))

# Next friday
print(TODAY + relativedelta(weekday=TU))
```
```
2022-10-14 11:14:20.556853
2022-10-21 11:14:20.556853
2022-10-21 17:14:20.556853
2022-09-20
```

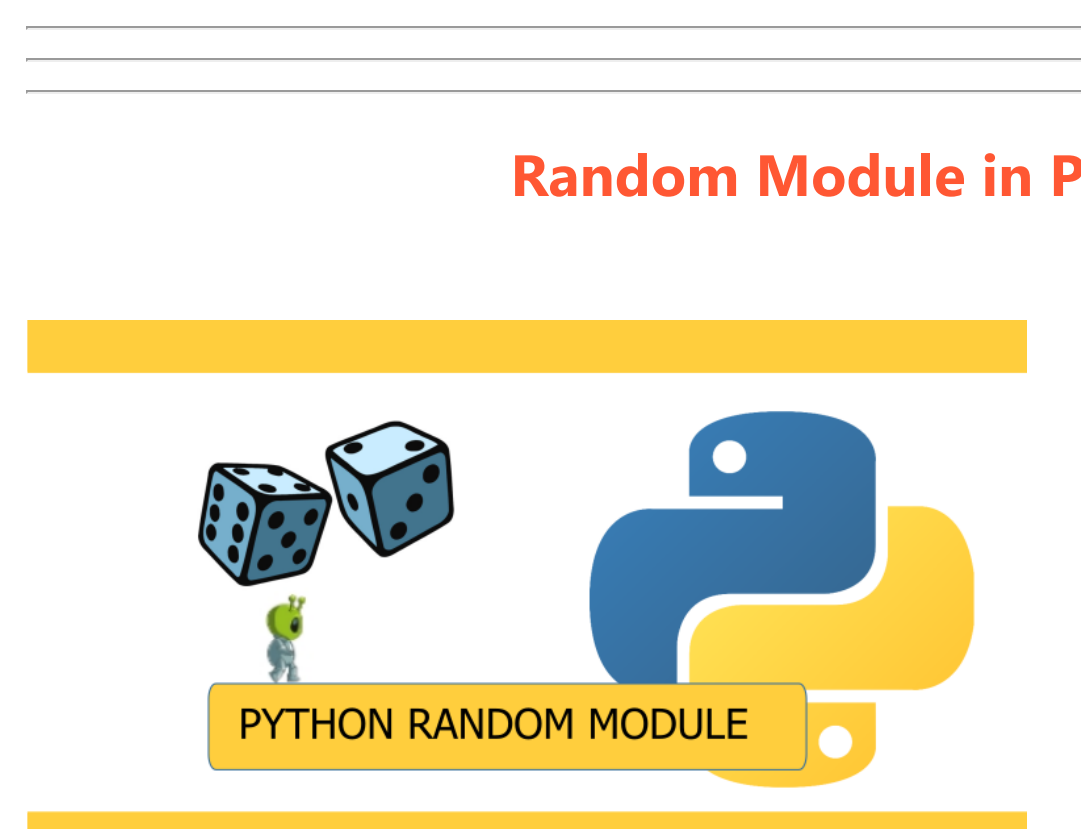
In [46]:
```
# Finding out the last tuesday in the month
print(TODAY + relativedelta(day=31, weekday=TU(-1)))

# We can also work with datetime objects directly - Example: Age of Sacra
sacra_birthday = datetime(1939, 4, 5, 12, 0)
print("Sacra's Age : ", relativedelta(NOW, sacra_birthday).years)
```
```
2022-09-27
Sacra's Age :  83
```

Alternatives to Python datetime and dateutil

- **NumPy provides a similar API to the built-in Python datetime library, but the NumPy version can be used in arrays.**
- **Pandas provides support for time-series data in DataFrames, usually sequential values of time-based events.**
- **cftime provides support for calendars other than the proleptic Gregorian calendar**
- **other packages includes Arrow,Pendulum,Maya,dateparser.**

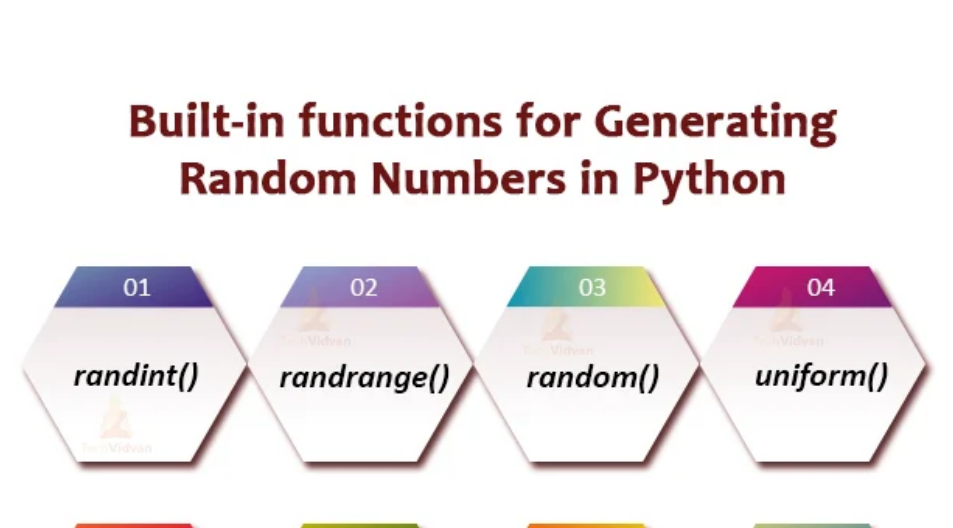
Random Module in Python:



</center>

- **The random module is a built-in module to generate the pseudo-random variables.**
- **It can be used perform some action randomly such as to get a random number, selecting a random elements from a list, shuffle elements randomly, etc.**
- **It has numerous classes and methods for generating different sorts of random numbers :**

Built-in functions for Generating Random Numbers in Python



In [47]:
```
#Importing Random Module
import random
print(dir(random))
```
```
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemRandom', 'TWOPI', '_Sequence',
'_Set', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package_
_', '__spec__', '_accumulate', '_acos', '_bisect', '_ceil', '_cos', '_e', '_exp', '_inst', '_log', '_os', '_p
i', '_random', '_repeat', '_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn', 'betavar
iate', 'choice', 'choices', 'expovariate', 'gammavariate', 'gauss', 'getrandbits', 'getstate', 'lognormvariat
e', 'normalvariate', 'paretovariate', 'randint', 'random', 'randrange', 'sample', 'seed', 'setstate', 'shuffl
e', 'triangular', 'uniform', 'vonmisesvariate', 'weibullvariate']
```

random( ) :

Generates a random floating-point number in the interval [0,1)

In [49]:
```
print(random.random())
print(random.random())
print(random.random())
```
```
0.2775806823883379
0.37592275314423684
0.2084827928560018
```

randint (x, y):

Generates a random integer from x to y, including the x and y.

In [51]:
```
print(random.randint(0,9))
print(random.randint(10,99))
print(random.randint(100,999))
print(random.randint(1000, 9999))
```
```
7
26
258
3598
```

randrange(start, stop, step):

Generates a random integer in the range(start, stop, step)

In [53]:
```
#In this example, the output will be a random integer from [1, 3, 5, 7, 9]
random.randrange(1, 10, 2)
```
Out[53]: `9`

uniform(x, y):

It generates a floating-point value between x and y.

```
In [54]: random.uniform(6, 9)
```

```
Out[54]: 8.062847320971375
```

choice(seq):

Chooses a random element from a non-empty sequence seq.

```
In [56]: seq = (12, 33, 67, 55, 78, 90, 34, 67, 88)
         random.choice(seq)
```

```
Out[56]: 88
```

sample(population, k):

Selects k unique random elements from a population sequence or set.

```
In [59]: seq = (12, 33, 67, 55, 78, 90, 34, 67, 88)
         random.sample(seq, 3)
```

```
Out[59]: [33, 12, 88]
```

shuffle(x):

Shuffles list x in place.

```
In [61]: l = [10, 20, 30, 40, 50]
         random.shuffle(l)
         print(l)
```

```
[20, 40, 30, 10, 50]
```

seed(x):

Generates the same sequence of random numbers every time you call seed(x).

```
In [63]: # seed value = 3
         random.seed(3)
         for i in range(3):
             print(random.random(), end = ' ')

         print('\n')

         # seed value = 8
         random.seed(8)
         for i in range(3):
             print(random.random(), end = ' ')

         print('\n')

         #whenever we call the seed() function with same value, it will produce the exact same sequence of random numbe
         # seed value again = 3
         random.seed(3)
         for i in range(3):
             print(random.random(), end = ' ')

         print('\n')


         # seed value = 8
         random.seed(8)
         for i in range(3):
             print(random.random(), end = ' ')

         print('\n')
```

```
0.23796462709189137 0.5442292252959519 0.36995516654807925

0.2267058593810488 0.9622950358343828 0.12633089865085956

0.23796462709189137 0.5442292252959519 0.36995516654807925

0.2267058593810488 0.9622950358343828 0.12633089865085956
```

Creating a list with random integers

```
In [64]: numbers = []

         for _ in range(10):
             num = random.randrange(10, 100)
             numbers.append(num)

         print(numbers)
```

```
[15, 20, 27, 41, 74, 36, 61, 92, 13, 68]
```

```
In [65]: numbers = [random.randrange(10, 100) for _ in range(10)]
         print(numbers)
```

```
[72, 68, 59, 73, 83, 34, 61, 21, 72, 39]
```

FAQ's :

- **What is datetime module in Python ?**
- **What is datetime datetime now( ) in Python ?**
- **How to print the date and time in Python ?**
- **How do you compare two times in Python ?**
- **How does Python calculate datetime difference ?**
- **What is datetime module in Python ?**
- **How to pick randomly from an array or a list ?**
- **Difference between random.randrange( ) and random.randint( )?**
- **Difference between random.uniform( ) and random.random( ) ?**
- **Difference between random.choices( ) and random.sample( ) ?**

To - Do

Date Time

- **Write a program to Know the Day of the Given Date?**
- **Generate a List of Dates from a Given Date**
- **Take date of births of 3 persons and print who is elder**
- **Subtract a week (7 days) from a given date in Python**
- **Add a week (7 days) and 12 hours to a given date**
- **Print current time in milliseconds**
- **Convert the following datetime into a string : given_date = datetime(2020, 2, 25)**
- **Calculate the date 4 months from the current date**

Random

- **Write a program to Generate 6 digit random secure OTP**
- **Write a program to randomly generate a floating-point number between 2.5 and 7.3**
- **Write a program to print a random value from the list [Phil, Cam, Luke, Lily, jay, Cameron, Michelle].**
- **Write a program to shuffle the list [4, 5, 6, 7, 8, 9, 10]**
- **Pick a random character from a given String**
- **Write a program to Generate random String of length 5**
- **Write a program to create a list with 4 random ten-digit integers**
- ***Generate a random Password which meets the following conditions :**
`Password length must be 10 characters long.`

`It must contain at least 2 upper case letters, 1 digit, and 1 special symbol.`

- **Generate a random date between given start and end dates**
- **Write a Program to Roll dice in such a way that every time you get the same number**
- **Random Lottery Pick : Generate 100 random lottery tickets and pick two lucky tickets from it as a winner:**
`The lottery number must be 10 digits long..`

`All 100 ticket number must be unique.`