In [ ]:

```
# Write your understanding on Package
#  method name: randint
    #  method name: random
    # method name : randrange
   # method name : uniform
```

In [8]:

```
from random import randint
randint(1,1000)
```

Out[8]:

```
414
```

In [36]:

```
import random
print(dir(random))
random.randint(10,20)
```

```
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemR
andom', 'TWOPI', '_ONE', '_Sequence', '__all__', '__builtins__', '__cached__', '_
_doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_accum
ulate', '_acos', '_bisect', '_ceil', '_cos', '_e', '_exp', '_fabs', '_floor', '_i
ndex', '_inst', '_isfinite', '_lgamma', '_log', '_log2', '_os', '_pi', '_random',
'_repeat', '_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_
warn', 'betavariate', 'binomialvariate', 'choice', 'choices', 'expovariate', 'gam
mavariate', 'gauss', 'getrandbits', 'getstate', 'lognormvariate', 'normalvariat
e', 'paretovariate', 'randbytes', 'randint', 'random', 'randrange', 'sample', 'se
ed', 'setstate', 'shuffle', 'triangular', 'uniform', 'vonmisesvariate', 'weibullv
ariate']
```

Out[36]:

```
13
```

In [33]:

```
from random import random
rand=random()
print(rand)
```

```
0.9064664449220112
```

In [25]:

```
help(random)
```

```
Help on built-in function random:

random() method of random.Random instance
    random() -> x in the interval [0, 1).
```

In [44]:

```
help(random.randrange)
random.randrange(1, 100, 3)
```

```
Help on method randrange in module random:

randrange(start, stop=None, step=1) method of random.Random instance
    Choose a random item from range(stop) or range(start, stop[, step]).

    Roughly equivalent to ``choice(range(start, stop, step))`` but
    supports arbitrarily large ranges and is optimized for common cases.
```

Out[44]:

```
22
```

In [45]:

```
help(random.uniform)
```

```
Help on method uniform in module random:

uniform(a, b) method of random.Random instance
    Get a random number in the range [a, b) or [a, b] depending on rounding.

    The mean (expected value) and variance of the random variable are:

        E[X] = (a + b) / 2
        Var[X] = (b - a) ** 2 / 12
```

In [47]:

```
random.uniform(12, 24)
```

Out[47]:

```
19.188405304372665
```

In [ ]:

```
# Package name : math
    # Method name: sqrt
    # Method name: pi  # constant no brackets
    # Method name: pow
    # Method name: sin
    # method name: e  # constant no brackets
    # method name: factorial
    # method name: gcd
```

In [54]:

```
from math import sqrt
```

In [55]:

```
help(sqrt)
```

```
Help on built-in function sqrt in module math:

sqrt(x, /)
    Return the square root of x.
```

In [53]:

```
s =sqrt(2)
print(s)
```

```
1.4142135623730951
```

In [57]:

```
from math import pi
help(pi)
```

```
Help on float object:

class float(object)
 |  float(x=0, /)
 |  
 |  Convert a string or number to a floating point number, if possible.
 |  
 |  Methods defined here:
 |  
 |  __abs__(self, /)
 |      abs(self)
 |  
 |  __add__(self, value, /)
 |      Return self+value.
 |  
 |  __bool__(self, /)
 |      True if self else False
 |  
 |  __ceil__(self, /)
 |      Return the ceiling as an Integral.
 |  
 |  __divmod__(self, value, /)
 |      Return divmod(self, value).
 |  
 |  __eq__(self, value, /)
 |      Return self==value.
 |  
 |  __float__(self, /)
 |      float(self)
 |  
 |  __floor__(self, /)
 |      Return the floor as an Integral.
 |  
 |  __floordiv__(self, value, /)
 |      Return self//value.
 |  
 |  __format__(self, format_spec, /)
 |      Formats the float according to format_spec.
 |  
 |  __ge__(self, value, /)
 |      Return self>=value.
 |  
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |  
 |  __getnewargs__(self, /)
 |  
 |  __gt__(self, value, /)
 |      Return self>value.
 |  
 |  __hash__(self, /)
 |      Return hash(self).
 |  
 |  __int__(self, /)
 |      int(self)
 |  
 |  __le__(self, value, /)
 |      Return self<=value.
 |  
 |  __lt__(self, value, /)
 |      Return self<value.
 |  
 |  __mod__(self, value, /)
 |      Return self%value.
 |  
 |  __mul__(self, value, /)
 |      Return self*value.
 |  
 |  __ne__(self, value, /)
 |      Return self!=value.
 |  
 |  __neg__(self, /)
 |      -self
 |  
 |  __pos__(self, /)
 |      +self
 |  
 |  __pow__(self, value, mod=None, /)
 |      Return pow(self, value, mod).
 |  
 |  __radd__(self, value, /)
 |      Return value+self.
 |  
 |  __rdivmod__(self, value, /)
 |      Return divmod(value, self).
 |  
 |  __repr__(self, /)
 |      Return repr(self).
 |  
 |  __rfloordiv__(self, value, /)
 |      Return value//self.
 |  
 |  __rmod__(self, value, /)
 |      Return value%self.
 |  
 |  __rmul__(self, value, /)
 |      Return value*self.
 |  
 |  __round__(self, ndigits=None, /)
 |      Return the Integral closest to x, rounding half toward even.
 |      
 |      When an argument is passed, work like built-in round(x, ndigits).
 |  
 |  __rpow__(self, value, mod=None, /)
 |      Return pow(value, self, mod).
 |  
 |  __rsub__(self, value, /)
 |      Return value-self.
 |  
 |  __rtruediv__(self, value, /)
 |      Return value/self.
 |  
 |  __sub__(self, value, /)
 |      Return self-value.
 |  
 |  __truediv__(self, value, /)
 |      Return self/value.
 |  
 |  __trunc__(self, /)
 |      Return the Integral closest to x between 0 and x.
 |
 |  as_integer_ratio(self, /)
 |      Return a pair of integers, whose ratio is exactly equal to the original f
loat.
 |
 |      The ratio is in lowest terms and has a positive denominator.  Raise
 |      OverflowError on infinities and a ValueError on NaNs.
 |
 |      >>> (10.0).as_integer_ratio()
 |      (10, 1)
 |      >>> (0.0).as_integer_ratio()
 |      (0, 1)
 |      >>> (-.25).as_integer_ratio()
 |      (-1, 4)
 |
 |  conjugate(self, /)
 |      Return self, the complex conjugate of any float.
 |
 |  hex(self, /)
 |      Return a hexadecimal representation of a floating-point number.
 |
 |      >>> (-0.1).hex()
 |      '-0x1.999999999999ap-4'
 |      >>> 3.14159.hex()
 |      '0x1.921f9f01b866ep+1'
 |
 |  is_integer(self, /)
 |      Return True if the float is an integer.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __getformat__(typestr, /)
 |      You probably don't want to use this function.
 |
 |        typestr
 |          Must be 'double' or 'float'.
 |
 |      It exists mainly to be used in Python's test suite.
 |
 |      This function returns whichever of 'unknown', 'IEEE, big-endian' or 'IEE
E,
 |      little-endian' best describes the format of floating point numbers used b
y the
 |      C type named by typestr.
 |
 |  fromhex(string, /)
 |      Create a floating-point number from a hexadecimal string.
 |
 |      >>> float.fromhex('0x1.ffffp10')
 |      2047.984375
 |      >>> float.fromhex('-0x1p-1074')
 |      -5e-324
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |  
 |  imag
 |      the imaginary part of a complex number
 |  
 |  real
 |      the real part of a complex number
```

In [68]:

```
r = 30 * pi
print(round(r,3))
```

```
94.248
```

In [61]:

```
30* 3.14
```

Out[61]:

```
94.2
```

In [69]:

```
from math import pow
```

In [70]:

```
help(pow)
```

```
Help on built-in function pow in module math:

pow(x, y, /)
    Return x**y (x to the power of y).
```

In [71]:

```
pow(2,3)
```

Out[71]:

```
8.0
```

In [73]:

```
pow(3,3)
```

Out[73]:

```
27.0
```

In [74]:

```
3*3
```

Out[74]:

```
9
```

In [75]:

```
3*9
```

Out[75]:

```
27
```

In [76]:

```
3*3*3
```

Out[76]:

```
27
```

In [81]:

```
import math
```

In [83]:

```
help(math.sin)
```

```
Help on built-in function sin in module math:

sin(x, /)
    Return the sine of x (measured in radians).
```

In [77]:
```
from math import sin
```

In [78]:
```
help(sin)
```

```
Help on built-in function sin in module math:

sin(x, /)
    Return the sine of x (measured in radians).
```

In [80]:
```
sin(2)
```

Out[80]:
```
0.9092974268256817
```

In [1]:
```
from math import e
```

In [2]:
```
help(e)
```

```
Help on float object:

class float(object)
 |  float(x=0, /)
 |
 |  Convert a string or number to a floating point number, if possible.
 |
 |  Methods defined here:
 |
 |  __abs__(self, /)
 |      abs(self)
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __bool__(self, /)
 |      True if self else False
 |
 |  __ceil__(self, /)
 |      Return the ceiling as an Integral.
 |
 |  __divmod__(self, value, /)
 |      Return divmod(self, value).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __float__(self, /)
 |      float(self)
 |
 |  __floor__(self, /)
 |      Return the floor as an Integral.
 |
 |  __floordiv__(self, value, /)
 |      Return self//value.
 |
 |  __format__(self, format_spec, /)
 |      Formats the float according to format_spec.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getnewargs__(self, /)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __int__(self, /)
 |      int(self)
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |  
 |  __mod__(self, value, /)
 |      Return self%value.
 |  
 |  __mul__(self, value, /)
 |      Return self*value.
 |  
 |  __ne__(self, value, /)
 |      Return self!=value.
 |  
 |  __neg__(self, /)
 |      -self
 |  
 |  __pos__(self, /)
 |      +self
 |  
 |  __pow__(self, value, mod=None, /)
 |      Return pow(self, value, mod).
 |  
 |  __radd__(self, value, /)
 |      Return value+self.
 |  
 |  __rdivmod__(self, value, /)
 |      Return divmod(value, self).
 |  
 |  __repr__(self, /)
 |      Return repr(self).
 |  
 |  __rfloordiv__(self, value, /)
 |      Return value//self.
 |  
 |  __rmod__(self, value, /)
 |      Return value%self.
 |  
 |  __rmul__(self, value, /)
 |      Return value*self.
 |  
 |  __round__(self, ndigits=None, /)
 |      Return the Integral closest to x, rounding half toward even.
 |      
 |      When an argument is passed, work like built-in round(x, ndigits).
 |  
 |  __rpow__(self, value, mod=None, /)
 |      Return pow(value, self, mod).
 |  
 |  __rsub__(self, value, /)
 |      Return value-self.
 |  
 |  __rtruediv__(self, value, /)
 |      Return value/self.
 |  
 |  __sub__(self, value, /)
 |      Return self-value.
 |  
 |  __truediv__(self, value, /)
 |      Return self/value.
 |  
 |  __trunc__(self, /)
 |      Return the Integral closest to x between 0 and x.
 |
 |  as_integer_ratio(self, /)
 |      Return a pair of integers, whose ratio is exactly equal to the original f
loat.
 |
 |      The ratio is in lowest terms and has a positive denominator.  Raise
 |      OverflowError on infinities and a ValueError on NaNs.
 |
 |      >>> (10.0).as_integer_ratio()
 |      (10, 1)
 |      >>> (0.0).as_integer_ratio()
 |      (0, 1)
 |      >>> (-.25).as_integer_ratio()
 |      (-1, 4)
 |
 |  conjugate(self, /)
 |      Return self, the complex conjugate of any float.
 |
 |  hex(self, /)
 |      Return a hexadecimal representation of a floating-point number.
 |
 |      >>> (-0.1).hex()
 |      '-0x1.999999999999ap-4'
 |      >>> 3.14159.hex()
 |      '0x1.921f9f01b866ep+1'
 |
 |  is_integer(self, /)
 |      Return True if the float is an integer.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __getformat__(typestr, /)
 |      You probably don't want to use this function.
 |
 |        typestr
 |          Must be 'double' or 'float'.
 |
 |      It exists mainly to be used in Python's test suite.
 |
 |      This function returns whichever of 'unknown', 'IEEE, big-endian' or 'IEE
E,
 |      little-endian' best describes the format of floating point numbers used b
y the
 |      C type named by typestr.
 |
 |  fromhex(string, /)
 |      Create a floating-point number from a hexadecimal string.
 |
 |      >>> float.fromhex('0x1.ffffp10')
 |      2047.984375
 |      >>> float.fromhex('-0x1p-1074')
 |      -5e-324
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |  
 |  imag
 |      the imaginary part of a complex number
 |  
 |  real
 |      the real part of a complex number
```

In [5]:

```
import math

# Use e in some calculation
x = 3
y = math.exp(x)  # This calculates e^x, or e^3

# Print the result
print(f"e^{x} = {y}")
```

```
e^3 = 20.085536923187668
```

In [6]:

```
from math import factorial
```

In [7]:

```
help(factorial)
```

```
Help on built-in function factorial in module math:

factorial(n, /)
    Find n!.
    
    Raise a ValueError if x is negative or non-integral.
```

In [11]:

```
factorial(8)
```

Out[11]:

```
40320
```

In [12]:

```
from math import gcd
```

In [13]:

```
help(gcd)
```

```
Help on built-in function gcd in module math:

gcd(*integers)
    Greatest Common Divisor.
```

In [20]:

```
# Find the GCD of 36 and 60
import math
a = 36
b = 60
math.gcd(a, b)

a = 36
b = 60

gcd_value = math.gcd(a, b)
print(f"The GCD of {a} and {b} is {gcd_value}")
```

```
The GCD of 36 and 60 is 12
```

```
In [85]: from time import sleep
         help(sleep)
```

```
Help on built-in function sleep in module time:

sleep(...)
    sleep(seconds)

    Delay execution for a given number of seconds.  The argument may be
    a floating point number for subsecond precision.
```

```
In [88]: a = 10
         print(a)
         sleep(2)

         b =20

         sleep(2)
         print(b)

         a+b
```

```
10
20
```

```
Out[88]: 30
```

```
In [21]: from sys import version
```

```
In [22]: help(version)
```

```
No Python documentation found for '3.12.4 | packaged by Anaconda, Inc. | (main, J
un 18 2024, 15:03:56) [MSC v.1929 64 bit (AMD64)]'.
Use help() to get the interactive help utility.
Use help(str) for help on the str class.
```

```
In [23]: version
```

```
Out[23]: '3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 15:03:56) [MSC v.192
         9 64 bit (AMD64)]'
```

```
In [ ]:
```