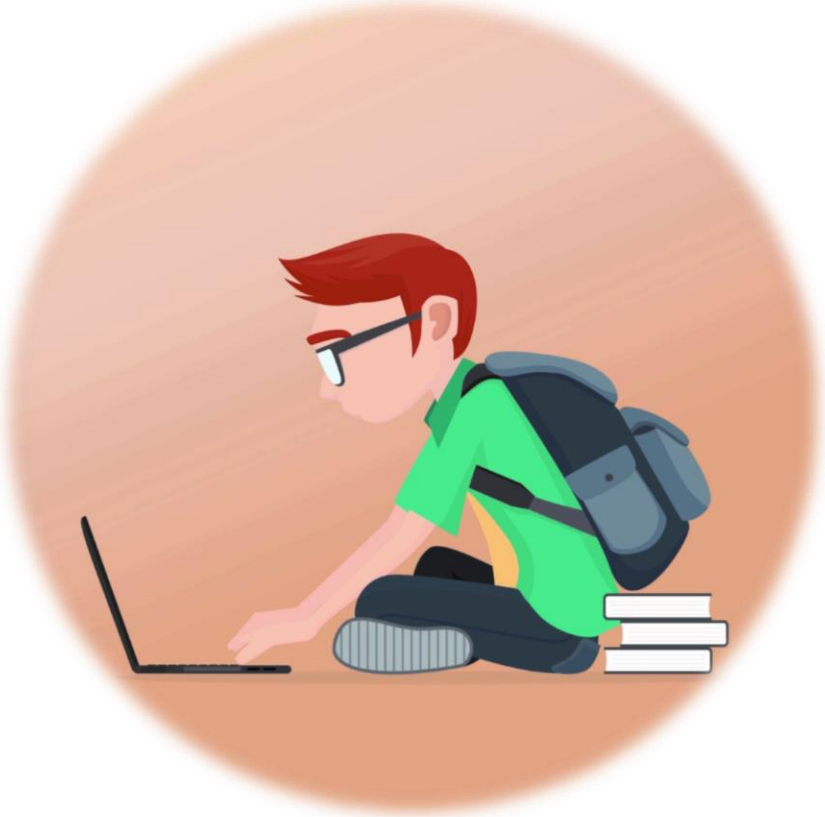
ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON

Возможности встроенной библиотеки Python

Крашенинников Роман Сергеевич
Главный специалист отдела системного администрирования РХТУ им. Д.И. Менделеева, ведущий программист кафедры информационных компьютерных технологий

ТЕМЫ

- Встроенные функции
- Особенности импорта модулей в Python
- Модуль **datetime**
- Модуль **itertools**
- Библиотека **math**
- Модуль **random**



ПОЛЕЗНЫЕ РЕСУРСЫ

- Встроенные функции документация
- Модуль datetime
- Модуль itertools
- Библиотека math
- Модуль random

ВСТРОЕННЫЕ ФУНКЦИИ

Встроенные функции - это функции, которые доступны для пользователя с самого начала без импортирования дополнительных библиотек. Всего существует 69 встроенных функций, однако не все имеют одинаковую значимость для решения вычислительных задач. Далее рассматриваются 24 наиболее полезные встроенные функции.

ВСТРОЕННЫЕ ФУНКЦИИ

input

```
res1 = input()
print(res1)
res2 = input("Enter data: ")
print(res2)
```

str

```
res1 = str(1.5)
print(res1)

res2 = str([1, 2])
print(res2)

res3 = str(abs)
print(res3)
```

list

```
res1 = list()
pritn(res1)
res2 = list((1, 2, 3))
print(res2)
res3 = list("abc")
print(res3)
res4 = list({"a": 1, "b": 2})
print(res4)
```

tuple

```
res1 = tuple()
pritn(res1)
res2 = tuple([1, 2, 3])
print(res2)
res3 = tuple("abc")
print(res3)
res4 = tuple({"a": 1, "b": 2})
print(res4)
```

len

```
a = [1, 2, 3, 4]
b = {"a": 1, "b": 2}
print(len(a))
print(len(b))
```

int

```
res1 = int()
print(res1)
res2 = int(2.5)
print(res2)
res3 = int("125")
print(res3)
# Преобразование из других
# систем счисления
res4 = int("FF", base=16)
print(res4)
```

print

```
print(1)
print(1, 2, 3)
print(1, 2, 3, sep=", ")
print(1, 2, 3, sep=", ", end=".\n")
```

float

```
res1 = float()
print(res1)
res2 = float(1)
print(res2)
res2 = float("3.7")
print(res2)
```

ВСТРОЕННЫЕ ФУНКЦИИ

Функция **abs** возвращает модуль числа или магнитуду комплексного числа. Функция **all** возвращает True если все элементы коллекции приводятся к True, иначе False.

```
res1 = abs(-4)
print(res1)

res2 = abs(1 - 5j)
print(res2)
```

```
4
5.099019513592784
```

```
res1 = all([True, True, False])
print(res1)
res2 = all([True, 1, "abc"])
print(res2)
res3 = all([-1, 2, 0, 3, 1])
print(res3)
```

```
False
True
False
```

ВСТРОЕННЫЕ ФУНКЦИИ

Функция **any** возвращает True если хотя бы один элемент коллекции приводится к True, иначе False. Функция **all** используется для преобразования различных типов данных в комплексное число. В случае преобразовании строки в комплексное число, строка не должна содержать пробелов между числами и плюсом.

```
res1 = all(["", 0, False])
print(res1)
res2 = all([True, 1, "abc"])
print(res2)
res3 = all([-1, 2, 0, 3, 1])
print(res3)
```

```
False
True
True
```

```
res1 = complex()
print(res1)
res2 = complex(1)
print(res2)
res2 = complex(1, 2)
print(res2)
res2 = complex("1+2j")
print(res2)
```

```
0j
(1+0j)
(1+2j)
(1+2j)
```

ВСТРОЕННЫЕ ФУНКЦИИ

Функция **divmod** возвращает результат целочисленного деления и остаток от деления. Функция **enumerate** используется для перечисления по коллекции с отслеживанием текущего индекса элемента.

```
res = divmod(7, 2)
pritn(res)
```

→

```
(3, 1)
```

```
a = [1, 2.5, "a", True]
print("From 0:")
for i, v in enumerate(a):
    print(f"{i}: {v}")
print("From 5:")
for i, v in enumerate(a, start=5):
    print(f"{i}: {v}")
```

→

```
From 0:
0) 1
1) 2.5
2) 'a'
3) True
From 5:
5) 1
6) 2.5
7) 'a'
8) True
```

ВСТРОЕННЫЕ ФУНКЦИИ

Функция **help** выводит справку по какому либо объекту. Функция **map** Применяет указанную функцию к элементам коллекции (или нескольких коллекций). Функция map возвращает объект map с которым можно работать непосредственно в цикле, либо преобразовать его к списку или кортежу.

```
help(2)
```

```
Help on int object:

class int(object)
 |  int([x]) -> integer
 |  int(x, base=10) -> integer
...
```

```
a = [1, 2, 3]
b = [4, 5 ,6]
m = map(lambda: x: x**2, a)
res1 = list(m)
print(res1)

m = map(lambda x1, x2: x1 + x2, a, b)
res2 = list(m)
print(res2)
```

```
[1, 4, 9]
[5, 7, 9]
```

ВСТРОЕННЫЕ ФУНКЦИИ

Функции **min** и **max** используются для нахождения минимума или максимума коллекции, соответственно. Функция **round** используется для округления чисел с плавающей точкой с учетом правил математики.

```
a = [1, 2, -3, 0, -5, 6]

res1 = min(a)
print(res1)

res2 = min(a, key=abs)
print(res2)
```

```
-5
0
```

```
res1 = round(4.576345)
pritn(res1)

res2 = round(4.576345, 2)
print(res2)
```

```
5
4.58
```

ВСТРОЕННЫЕ ФУНКЦИИ

Функции **sorted** используется для сортировки элементов коллекции. Функция **sum** возвращает сумму элементов коллекции. Второй аргумент отвечает за число к которому будут прибавляться элементы коллекции (по умолчанию 0).

```
a = [1, 2, -3, 0, -5, 6]

res1 = sorted(a)
print(res1)

res2 = sorted(a, key=abs)
print(res2)

res3 = sorted(a,
        key=abs,
        reverse=True)
print(res3)
```

→

```
[-5, -3, 0, 1, 2, 6]
[0, 1, 2, -3, -5, 6]
[6, -5, -3, 2, 1, 0]
```

```
a = [1, 2, 3, 4]

res1 = sum(a)
print(res1)

res2 = sum(a, 10)
print(res2)
```

→

```
10
20
```

ВСТРОЕННЫЕ ФУНКЦИИ

Функции **type** возвращает тип указанного объекта. Функция **zip** позволяет реализовывать одновременное перечисление по нескольким коллекциям. перечисление осуществляется до конца самой короткой коллекции.

```
res1 = type(2)
print(res1)

res2 = type(0.5)
print(res2)

res3 = type(abs)
print(res3)

res4 = type(5) == int
print(res4)
```

→

```
<class 'int'>
<class 'float'>
<class 'builtin_function_or_method'>
True
```

```
a = [1, 2, 3, 4]
b = [5, 6, 7, 8, 9, 10]

for i, j, k in zip(a, b, a):
    print(f"{i} - {j} - {k}")
```

→

```
1 - 5 - 1
2 - 6 - 2
3 - 7 - 3
4 - 8 - 4
```

ИМПОРТ МОДУЛЕЙ

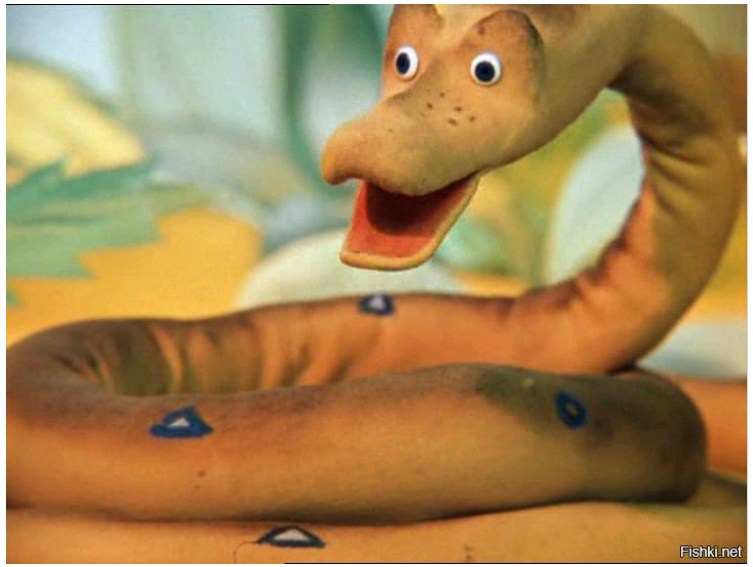
Подключить модуль можно с помощью инструкции import. осле ключевого слова import указывается название модуля. Одной инструкцией можно подключить несколько модулей, хотя этого не рекомендуется делать, так как это снижает читаемость кода. После импортирования модуля его название становится переменной, через которую можно получить доступ к атрибутам модуля. Если название модуля слишком длинное, или оно вам не нравится по каким-то другим причинам, то для него можно создать псевдоним, с помощью ключевого слова as. Подключить определенные атрибуты модуля можно с помощью инструкции from

```
#импортируем модуль, datetime можем использовать как переменную
import datetime

#сокращаем имя через псевдонимы
import itertools as it

#импортируем всё (не рекомендуется)
from json import *

#импортируем отдельные атрибуты и используем псевдонимы
from math import e, ceil as c
```



МОДУЛЬ DATETIME

Модуль datetime предоставляет классы для обработки времени и даты разными способами. Поддерживается и стандартный способ представления времени, однако больший упор сделан на простоту манипулирования датой, временем и их частями.

МОДУЛЬ DATETIME (ОСНОВНЫЕ ТИПЫ)

- **datetime.date(year, month, day)** - стандартная дата. Атрибуты: **year, month, day.** Неизменяемый объект
- **datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None)** - стандартное время, не зависит от даты. Атрибуты: **hour, minute, second, microsecond, tzinfo.**
- **datetime.timedelta** - разница между двумя моментами времени, с точностью до микросекунд.
- **datetime.datetime**(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None) - комбинация даты и времени

МОДУЛЬ DATETIME (ОСНОВНЫЕ ФУНКЦИИ)

- **datetime.today()** – текущая дата и время.
- **datetime.combine**(date, time) – datetime из комбинации объектов date и time.
- **datetime.strptime**(date_string, format) – преобразует строку в datetime.
- **datetime.strftime**(format) – преобразует datetime в строку.
- **datetime.weekday**() – день недели в виде числа, понедельник - 0, воскресенье - 6.

МОДУЛЬ ITERTOOLS

Данный модуль является сборником полезных итераторов, повышающих эффективность работы с циклами и генераторами последовательностей объектов. Это достигается за счет лучшего управления памятью в программе, быстрого выполнения подключаемых функций, а также сокращения и упрощения кода. Готовые методы, реализованные в данной библиотеке, принимают различные параметры для управления генератором последовательности, чтобы вернуть вызывающей подпрограмме необходимый набор объектов.

МОДУЛЬ ITERTOOLS (ФУНКЦИИ)

itertools.count(start=0, step=1) - бесконечная арифметическая прогрессия с первым членом start и шагом step.

itertools.cycle(iterable) - возвращает по одному значению из последовательности, повторенной бесконечное число раз.

itertools.repeat(elem, n=Inf) - повторяет elem n раз.

itertools.accumulate(iterable) - аккумулирует суммы.

itertools.chain(*iterables) - возвращает по одному элементу из первого итератора, потом из второго, до тех пор, пока итераторы не кончатся.

itertools.combinations(iterable, [r]) - комбинации длиной r из iterable без повторяющихся элементов.

itertools.combinations_with_replacement(iterable, r) - комбинации длиной r из iterable с повторяющимися элементами.

МОДУЛЬ ITERTOOLS (ФУНКЦИИ)

itertools.filterfalse(func, iterable) - все элементы, для которых func возвращает ложь.

itertools.groupby(iterable, key=None) - группирует элементы по значению. Значение получается применением функции key к элементу (если аргумент key не указан, то значением является сам элемент).

itertools.permutations(iterable, r=None) - перестановки длиной r из iterable.

itertools.product(*iterables, repeat=1) - аналог вложенных циклов.

itertools.starmap(function, iterable) - применяет функцию к каждому элементу последовательности (каждый элемент распаковывается).

itertools.takewhile(func, iterable) - элементы до тех пор, пока func возвращает истину.

itertools.tee(iterable, n=2) - кортеж из n итераторов.

itertools.zip_longest(*iterables, fillvalue=None) - как встроенная функция zip, но берет самый длинный итератор, а более короткие дополняет fillvalue.

БИБЛИОТЕКА MATH

Python библиотека math содержит наиболее применяемые математические функции и константы. Все вычисления происходят на множестве вещественных чисел.

math.ceil(X) – округление до ближайшего большего числа.

math.copysign(X, Y) - возвращает число, имеющее модуль такой же, как и у числа X, а знак - как у числа Y.

math.fabs(X) - модуль X.

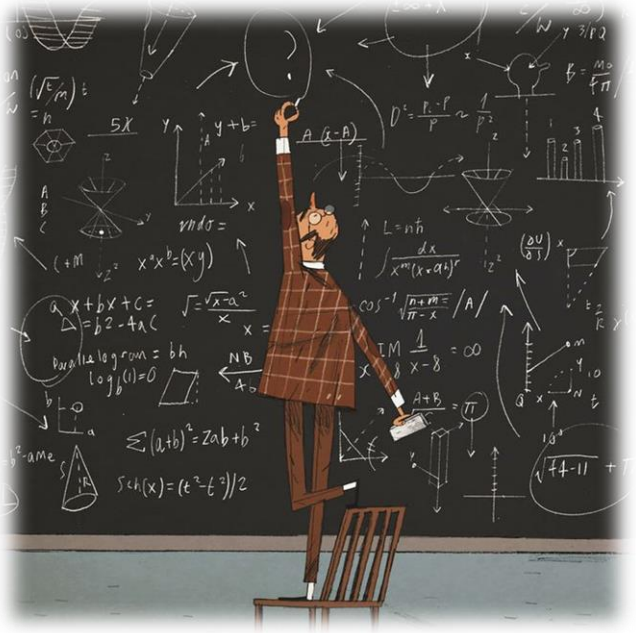
math.factorial(X) - факториал числа X.

math.floor(X) - округление вниз.

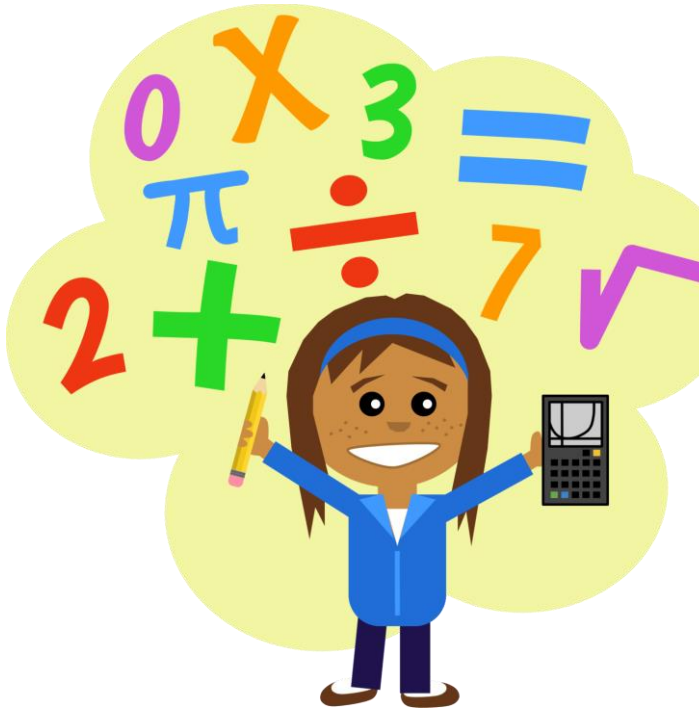
math.fmod(X, Y) - остаток от деления X на Y.

math.frexp(X) - возвращает мантиссу и экспоненту числа.

math.ldexp(X, I) - X * 2^i. Функция, обратная функции math.frexp().



БИБЛИОТЕКА MATH



math.fsum(последовательность) - сумма всех членов последовательности. Эквивалент встроенной функции sum(), но math.fsum() более точна для чисел с плавающей точкой.

math.isfinite(X) - является ли X числом.

math.isinf(X) - является ли X бесконечностью.

math.isnan(X) - является ли X NaN (Not a Number - не число).

math.modf(X) - возвращает дробную и целую часть числа X. Оба числа имеют тот же знак, что и X.

math.trunc(X) - усекает значение X до целого.

math.exp(X) - e^X.

math.expm1(X) - e^X - 1. При X → 0 точнее, чем math.exp(X)-1.

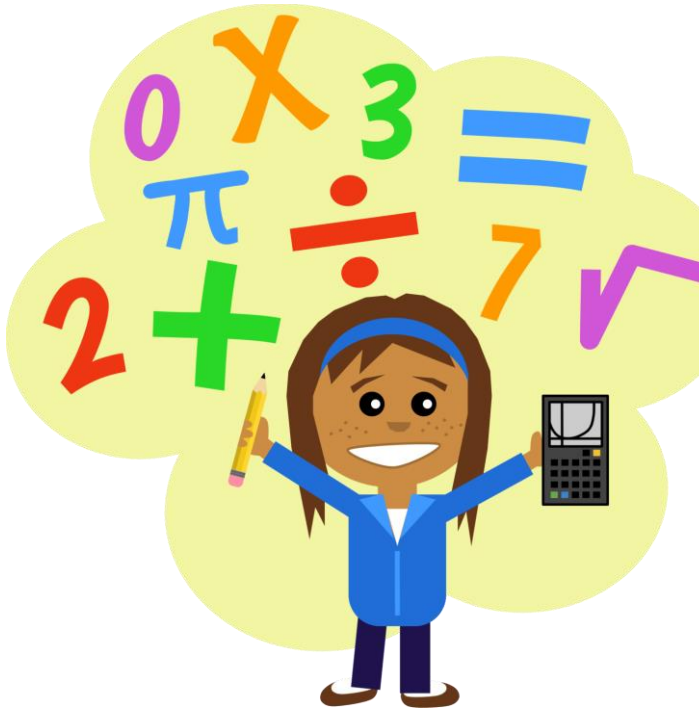
math.log(X, [base]) - логарифм X по основанию base. Если base не указан, вычисляется натуральный логарифм.

math.log1p(X) - натуральный логарифм (1 + X). При X → 0 точнее, чем math.log(1+X).

math.log10(X) - логарифм X по основанию 10.

math.log2(X) - логарифм X по основанию 2.

БИБЛИОТЕКА MATH



math.pow(X, Y) - X^Y.

math.sqrt(X) - квадратный корень из X.

math.acos(X) - арккосинус X. В радианах.

math.asin(X) - арксинус X. В радианах.

math.atan(X) - арктангенс X. В радианах.

math.atan2(Y, X) - арктангенс Y/X. В радианах. С учетом четверти, в которой находится точка (X, Y).

math.cos(X) - косинус X (X указывается в радианах).

math.sin(X) - синус X (X указывается в радианах).

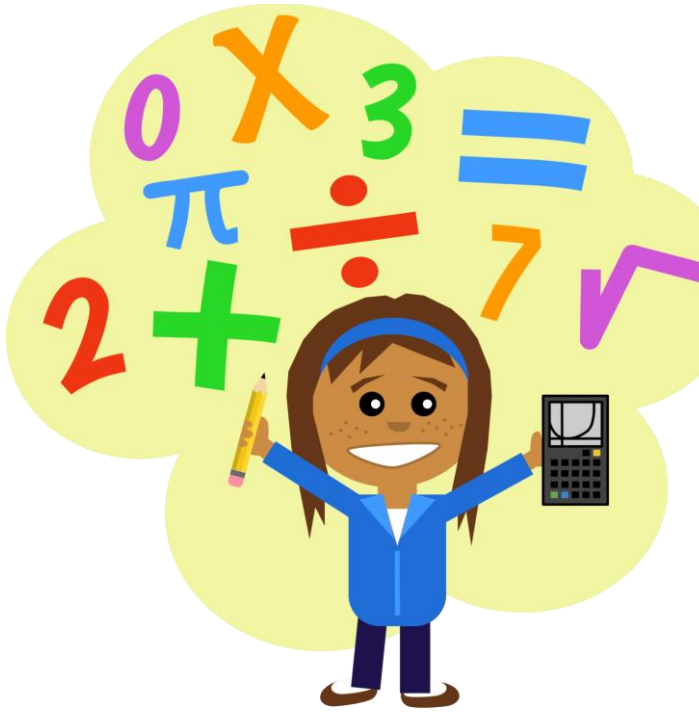
math.tan(X) - тангенс X (X указывается в радианах).

math.hypot(X, Y) - вычисляет гипотенузу треугольника с катетами X и Y (math.sqrt(x * x + y * y)).

math.degrees(X) - конвертирует радианы в градусы.

math.radians(X) - конвертирует градусы в радианы.

БИБЛИОТЕКА MATH



math.cosh(X) - вычисляет гиперболический косинус.

math.sinh(X) - вычисляет гиперболический синус.

math.tanh(X) - вычисляет гиперболический тангенс.

math.acosh(X) - вычисляет обратный гиперболический косинус.

math.asinh(X) - вычисляет обратный гиперболический синус.

math.atanh(X) - вычисляет обратный гиперболический тангенс.

math.erf(X) - функция ошибок.

math.erfc(X) - дополнительная функция ошибок (1 - math.erf(X)).

math.gamma(X) - гамма-функция X.

math.lgamma(X) - натуральный логарифм гамма-функции X.

math.pi - pi = 3,1415926...

math.e - e = 2,718281...

МОДУЛЬ RANDOM

Модуль random предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.

random.randrange(start, stop, step) - возвращает случайно выбранное число из последовательности.

random.randint(A, B) - случайное целое число N, A ≤ N ≤ B.

random.choice(sequence) - случайный элемент непустой последовательности.

random.shuffle(sequence, [rand]) - перемешивает последовательность (изменяется сама последовательность). Поэтому функция не работает для неизменяемых объектов.

random.sample(population, k) - список длиной k из последовательности population.

random.random() - случайное число от 0 до 1.

random.uniform(A, B) - случайное число с плавающей точкой, A ≤ N ≤ B (или B ≤ N ≤ A).

МОДУЛЬ RANDOM

random.triangular(low, high, mode) - случайное число с плавающей точкой, low ≤ N ≤ high. Mode - распределение.

random.betavariate(alpha, beta) - бета-распределение. alpha>0, beta>0. Возвращает от 0 до 1.

random.expovariate(lambd) - экспоненциальное распределение. lambd равен 1/среднее желаемое. Lambd должен быть отличным от нуля. Возвращаемые значения от 0 до плюс бесконечности, если lambd положительно, и от минус бесконечности до 0, если lambd отрицательный.

random.gammavariate(alpha, beta) - гамма-распределение. Условия на параметры alpha>0 и beta>0.

random.gauss(значение, стандартное отклонение) - распределение Гаусса.

random.lognormvariate(mu, sigma) - логарифм нормального распределения. Если взять натуральный логарифм этого распределения, то вы получите нормальное распределение со средним mu и стандартным отклонением sigma. mu может иметь любое значение, и sigma должна быть больше нуля.

random.normalvariate(mu, sigma) - нормальное распределение. mu - среднее значение, sigma - стандартное отклонение.

random.vonmisesvariate(mu, kappa) - mu - средний угол, выраженный в радианах от 0 до 2π, и kappa - параметр концентрации, который должен быть больше или равен нулю. Если каппа равна нулю, это распределение сводится к случайному углу в диапазоне от 0 до 2π.

random.paretovariate(alpha) - распределение Парето.

random.weibullvariate(alpha, beta) - распределение Вейбулла.

ЗАДАНИЕ ДЛЯ САМОКОНТРОЛЯ

Необходимо смоделировать следующую ситуацию:

В сказочном мире, располагающемся на квадратной площади 10x10 клеток проживают 4 народа, которые борются между собой. В начальный момент времени в мире присутствует по 10 представителей каждого народа, расположенных в случайных координатах. Помимо координат, у каждого представителя есть дата рождения, меньше сегодняшней даты и случайное значение силы от 0 до 100. Далее каждый представитель начинает двигаться в одном из четырех направлений, выбранном случайно. При этом он попадает в одну из ситуаций:

1) Клетка, куда он перемещается, свободна, тогда координаты особи просто меняются на новые.
2) Представитель стоит у края, тогда он просто остается на месте
3) Клетка занята представителем того же народа, тогда оба представителя остаются на месте, а их сила возрастает на единицу
4) Клетка занята представителем другого народа, тогда происходит сражение у кого округленное значение (сила + π * полных лет) больше. Проигравшая особь меняет свой народ на народ победившего

Учесть 2 момента:

- Каждую итерацию в мире сменяется сезон, всего их 4 и в соответствующий сезон сила соответствующего народа временно возрастает в 1.5 раза.
- Каждую итерацию последовательность передвижения меняется.

Победивший народ – народ с наибольшим количеством представителей после 100 итераций.

ЗАДАНИЕ ДЛЯ САМОКОНТРОЛЯ

Подсказки по реализации:

- Каждого представителя задайте словарем с соответствующими полями, а всех представителей храните в списке.
- Для изменения последовательности передвижения, просто перемешивайте ваш список при помощи **shuffle** и проходитесь по получившемуся списку.
- Сменяйте сезоны при помощи **itertools.cycle**

СПАСИБО ЗА ВНИМАНИЕ!