

# Лекция 6. Основные встроенные модули

## Содержание

1. Модуль `random`
2. Модуль `math`
3. Модуль `locale`
4. Модуль `decimal`

# Модуль `random`

Модуль `random` управляет генерацией случайных чисел. Его основные функции:

- **`random()`**: генерирует случайное число от 0.0 до 1.0
- **`randint()`**: возвращает случайное число из определенного диапазона
- **`randrange()`**: возвращает случайное число из определенного набора чисел
- **`shuffle()`**: перемешивает список
- **`choice()`**: возвращает случайный элемент списка

Функция **random()** возвращает случайное число с плавающей точкой в промежутке от 0.0 до 1.0.

```
import random
```

```
number = random.random() # значение от 0.0 до 1.0
```

```
print(number)
```

```
number = random.random() * 100 # значение от 0.0 до 100.0
```

```
print(number)
```

Функция **randint(min, max)** возвращает случайное целое число в промежутке между двумя значениями min и max.

```
import random
```

```
number = random.randint(20, 35) # значение от 20 до 35  
print(number)
```

Функция **randrange()** возвращает случайное целое число из определенного набора чисел. Она имеет три формы:

- `randrange(stop)` : в качестве набора чисел, из которых происходит извлечение случайного значения, будет использоваться диапазон от 0 до числа `stop`
- `randrange(start, stop)` : набор чисел представляет диапазон от числа `start` до числа `stop`
- `randrange(start, stop, step)` : набор чисел представляет диапазон от числа `start` до числа `stop`, при этом каждое число в диапазоне отличается от предыдущего на шаг `step`

```
import random
```

```
number = random.randrange(10) # значение от 0 до 10
```

```
print(number)
```

```
number = random.randrange(2, 10) # значение в диапазоне  
2, 3, 4, 5, 6, 7, 8, 9, 10
```

```
print(number)
```

```
number = random.randrange(2, 10, 2) # значение в  
диапазоне 2, 4, 6, 8, 10
```

```
print(number)
```

## Работа со списком

Для работы со списками в модуле `random` определены две функции: функция **`shuffle()`** перемешивает список случайным образом, а функция **`choice()`** возвращает один случайный элемент из списка:

```
import random
numbers = [1, 2, 3, 4, 5, 6, 7, 8]
random.shuffle(numbers)
print(numbers)          # 1
random_number = random.choice(numbers)
print(random_number)
```

```
[7, 4, 3, 6, 5, 1, 8, 2]
```

```
8
```

# Модуль **math**

Встроенный модуль **math** в Python предоставляет набор функций для выполнения математических, тригонометрических и логарифмических операций. Некоторые из основных функций модуля:

- **pow(num, power)**: возведение числа `num` в степень `power`
- **sqrt(num)**: квадратный корень числа `num`
- **ceil(num)**: округление числа до ближайшего наибольшего целого
- **floor(num)**: округление числа до ближайшего наименьшего целого
- **factorial(num)**: факториал числа
- **degrees(rad)**: перевод из радиан в градусы
- **radians(grad)**: перевод из градусов в радианы
- **cos(rad)**: косинус угла в радианах
- **sin(rad)**: синус угла в радианах
- **tan(rad)**: тангенс угла в радианах
- **acos(rad)**: арккосинус угла в радианах
- **asin(rad)**: арксинус угла в радианах
- **atan(rad)**: арктангенс угла в радианах
- **log(n, base)**: логарифм числа `n` по основанию `base`
- **log10(n)**: десятичный логарифм числа `n`

```
import math
# возведение числа 2 в степень 3
n1 = math.pow(2, 3)
print(n1) # 8
# ту же самую операцию можно выполнить так
n2 = 2**3
print(n2)
# возведение в квадрат
print(math.sqrt(9)) # 3
# ближайшее наибольшее целое число
print(math.ceil(4.56)) # 5
# ближайшее наименьшее целое число
print(math.floor(4.56)) # 4
# перевод из радиан в градусы
print(math.degrees(3.14159)) # 180
```

```
# перевод из градусов в радианы
print(math.radians(180))    # 3.1415.....
# КОСИНУС
print(math.cos(math.radians(60))) # 0.5
# СИНОС
print(math.sin(math.radians(90))) # 1.0
# ТАНГЕНС
print(math.tan(math.radians(0)))  # 0.0
print(math.log(8,2))             # 3.0
print(math.log10(100))           # 2.0
```

```
import math
radius = 30
# площадь круга с радиусом 30
area = math.pi * math.pow(radius, 2)
print(area)

# натуральный логарифм числа 10
number = math.log(10, math.e)
print(number)
```

# Модуль locale

# англосаксонская система

1,234.567

# европейская система

1.234,567

И для решения проблемы форматирования под определенную культуру в Python имеется встроенный модуль **locale**.

Для установки локальной культуры в модуле locale определена функция **setlocale()**. Она принимает два параметра:

```
1setlocale(category, locale)
```

Первый параметр указывает на категорию, к которой применяется функция - к числам, валютам или и числам, и валютам. В качестве значения для параметра мы можем передавать одну из следующих констант:

- **LC\_ALL**: применяет локализацию ко всем категориям - к форматированию чисел, валют, дат и т.д.
- **LC\_NUMERIC**: применяет локализацию к числам
- **LC\_MONETARY**: применяет локализацию к валютам
- **LC\_TIME**: применяет локализацию к датам и времени
- **LC\_STYPE**: применяет локализацию при переводе символов в верхний или нижний регистр
- **LC\_COLLATE**: применяет локаль при сравнении строк

Второй параметр функции `setlocale` указывает на локальную культуру, которую надо использовать. На ОС Windows можно использовать код страны по ISO из двух символов, например, для США - "us", для Германии - "de", для России - "ru". Но на MacOS необходимо указывать код языка и код страны, например, для английского в США - "en\_US", для немецкого в Германии - "de\_DE", для русского в России - "ru\_RU". По умолчанию фактически используется культура "en\_US".

Непосредственно для форматирования чисел и валют модуль `locale` предоставляет две функции:

- `currency(num)` : форматирует валюту
- `format(str, num)` : подставляет число `num` вместо плейсхолдера в строку `str`

Применяются следующие плейсхолдеры:

- `d`: для целых чисел
- `f`: для чисел с плавающей точкой
- `e`: для экспоненциальной записи чисел

Перед каждым плейсхолдером ставится знак процента `%`, например:

`"%d"`

Применим локализацию чисел и валют в немецкой культуре:

```
import locale
locale.setlocale(locale.LC_ALL, "de")           # для Windows
# locale.setlocale(locale.LC_ALL, "de_DE")     # для MacOS
number = 12345.6789
formatted = locale.format("%f", number)
print(formatted)           # 12345,678900
formatted = locale.format("%.2f", number)
print(formatted)          # 12345,68
formatted = locale.format("%d", number)
print(formatted)          # 12345
formatted = locale.format("%e", number)
print(formatted)          # 1,234568e+04
money = 234.678
formatted = locale.currency(money)
print(formatted)          # 234,68 €
```

# Модуль `decimal`

При работе с числами с плавающей точкой (то есть `float`) мы сталкиваемся с тем, что в результате вычислений мы получаем не совсем верный результат:

```
number = 0.1 + 0.1 + 0.1  
print(number)          # 0.30000000000000004
```

Проблему может решить использование функции **`round()`**, которая округлит число. Однако есть и другой способ, который заключается в использовании встроенного модуля **`decimal`**.

```
from decimal import Decimal

number = Decimal("0.1")
number = number + number + number
print(number)          # 0.3
```

# Округление чисел

```
from decimal import Decimal
```

```
number = Decimal("0.444")  
number = number.quantize(Decimal("1.00"))  
print(number)           # 0.44
```

```
number = Decimal("0.555678")  
print(number.quantize(Decimal("1.00")))           # 0.56
```

```
number = Decimal("0.999")  
print(number.quantize(Decimal("1.00")))           # 1.00
```

```
from decimal import Decimal, ROUND_HALF_EVEN
```

```
number = Decimal("10.025")
```

```
print(number.quantize(Decimal("1.00"), ROUND_HALF_EVEN)) # 10.02
```

```
number = Decimal("10.035")
```

```
print(number.quantize(Decimal("1.00"), ROUND_HALF_EVEN)) # 10.04
```

```
number = Decimal("10.026")  
print(number.quantize(Decimal("1.00"), ROUND_HALF_DOWN))      # 10.03
```

```
number = Decimal("10.025")  
print(number.quantize(Decimal("1.00"), ROUND_HALF_DOWN))      # 10.02
```

```
number = Decimal("10.021")  
print(number.quantize(Decimal("1.00"), ROUND_CEILING))      # 10.03
```

```
number = Decimal("10.025")  
print(number.quantize(Decimal("1.00"), ROUND_CEILING))      # 10.03
```

```
number = Decimal("10.021")  
print(number.quantize(Decimal("1.00"), ROUND_FLOOR))           # 10.02
```

```
number = Decimal("10.025")  
print(number.quantize(Decimal("1.00"), ROUND_FLOOR))           # 10.02
```