tan(x)	Тангенс угла, задаваемого в радианах
asin(x)	Арксинус, возвращает значение в радианах
acos(x)	Арккосинус, возвращает значение в радианах
atan(x)	Арктангенс, возвращает значение в радианах
atan2(y, x)	Полярный угол (в радианах) точки с координатами (х, у).
hypot(a, b)	Длина гипотенузы прямоугольного треугольника с катетами а и b.
degrees(x)	Преобразует угол, заданный в радианах, в градусы.
radians(x)	Преобразует угол, заданный в градусах, в радианы.
pi	Константа π

Некоторые из перечисленных функций (int, round, abs) являются стандартными и не требуют подключения модуля math для использования.

## Оборудование и материалы.

Персональный компьютер, среда разработки Python.

# Указания по технике безопасности:

Соответствуют технике безопасности по работе с компьютерной техникой.

#### Задания

Для начала вернемся к стандартной библиотеке и рассмотрим некоторые особенности и модули.

# Математические функции

Python имеет множество математических функций в стандартной библиотеке math. Просто введите import math, чтобы получить к ним доступ из своих программ.

Она содержит такие константы, как рі и е:

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.e
2.718281828459045
```

В основном код состоит из функций, рассмотрим наиболее полезные из них.

Функция fabs() возвращает абсолютное значение своего аргумента:

```
>>> math.fabs(98.6)
98.6
>>> math.fabs(-271.1)
271.1
```

Получаем округление вниз (floor()) и вверх (ceil()) некоторого числа:

```
>>> math.floor(98.6)
98
>>> math.floor(-271.1)
-272
>>> math.ceil(98.6)
99
>>> math.ceil(-271.1)
-271
```

Высчитываем факториал (в математике это выглядит как n!) с помощью функции factorial():

```
>>> math.factorial(0)
      >>> math.factorial(1)
      1
      >>> math.factorial(2)
      2
      >>> math.factorial(3)
      >>> math.factorial(10)
      3628800
     Получаем натуральный логарифм аргумента с помощью функции log():
      >>> math.log(1.0)
      0.0
      >>> math.log(math.e)
      1.0
     Если вы хотите задать другое основание логарифма, передайте его как второй
аргумент:
      >>> math.log(8, 2)
      3.0
     Функция pow() возвращает противоположный результат, возводя число в степень:
      >>> math.pow(2, 3)
      8.0
     Python имеет также встроенный оператор экспоненты **, делающий то же самое,
но он не преобразует автоматически результат к числу с плавающей точкой, если
основание и степень были целыми числами:
      >>> 2**3
      >>> 2.0**3
      8.0
     Получаем квадратный корень с помощью функции sqrt():
      >>> math.sqrt(100.0)
      10.0
     Если вы попытаетесь задать отрицательный аргумент, то получите:
      >>> math.sgrt(-100.0)
      Traceback (most recent call last):
        File "<stdin>". line 1. in <module>
      ValueError: math domain error
     Здесь имеются также обычные тригонометрические функции, я просто приведу их
```

Здесь имеются также обычные тригонометрические функции, я просто приведу их названия: sin(), cos(), tan(), asin(), acos(), atan() и atan2(). Если вы помните теорему Пифагора (или хотя бы можете произнести ее название быстро три раза, не начав

плеваться), библиотека math предоставит вам также функцию hypot(), которая рассчитывает гипотенузу на основании длины двух катетов:

Если вы не доверяете таким функциям, можете сделать все самостоятельно:

```
>>> math.sqrt(x*x + y*y)
5.0
>>> math.sqrt(x**2 + y**2)
5.0
```

Последний набор функций преобразует угловые координаты:

```
>>> math.radians(180.0)
3.141592653589793
>>> math.degrees(math.pi)
180.0
```

# Работа с комплексными числами

Комплексные числа полностью поддерживаются основой языка Python в традиционной нотации с мнимой и действительной частями:

```
>>> # действительное число
... 5
5
>>> # мнимое число
... 8ј
8ј
>>> # мнимое число
... 3 + 2ј
(3+2j)
```

Поскольку мнимое число i (в Python оно записывается как 1j) определено как квадратный корень из -1, мы можем выполнить следующее:

```
>>> 1j * 1j
(-1+0j)
>>> (7 + 1j) * 1j
(-1+7j)
```

Некоторые математические функции для комплексных чисел содержатся в стандартном модуле cmath.

#### Paccчитываем точное значение чисел с плавающей точкой с помощью decimal

Числа с плавающей точкой в вычислительной технике не похожи на настоящие числа, которые мы изучали в школе. Из-за того что компьютеры разрабатывались для

бинарной математики, числа, не являющиеся равными степени двойки, зачастую не могут быть выражены точно:

```
>>> x = 10.0 / 3.0
>>> x
3.3333333333333333333
```

Эй, что это за пятерка в конце? На ее месте должна быть тройка. С помощью модуля decimal (http://docs.python.org/3/library/decimal.html) вы можете записывать числа с желаемым уровнем точности. Это особенно важно для расчетов денежных сумм. Сумма в валюте Соединенных Штатов не может быть меньше 1 цента (сотая часть доллара), поэтому, если мы подсчитываем количество долларов и центов, нам нужно считать все до копеечки. Если мы попробуем представить доллары и центы как значения с плавающей точкой вроде 19,99 или 0,06, то потеряем некоторую часть точности в последних битах еще до начала вычислений. Как с этим справиться? Легко. Нам нужно использовать модуль decimal:

```
>>> from decimal import Decimal
>>> price = Decimal('19.99')
>>> tax = Decimal('0.06')
>>> total = price + (price * tax)
>>> total
Decimal('21.1894')
```

Мы записали цену и налог как строковые значения, чтобы сохранить их точность. При вычислении переменной total мы обработали все значащие части цента, но хотим получить ближайшее целое значение цента:

```
>>> penny = Decimal('0.01')
>>> total.quantize(penny)
Decimal('21.19')
```

Такие же результаты можно получить с помощью старых добрых чисел с плавающей точкой и округлений, но не всегда. Вы можете умножить сумму на 100 и использовать при подсчетах количество центов, но и при таком подходе можете пострадать.

# Выполняем вычисления для рациональных чисел с помощью модуля fractions

Вы можете представлять числа как числитель, разделенный на знаменатель, с помощью модуля стандартной библиотеки fractions. Рассмотрим пример простой операции умножения 1/3 на 2/3:

```
>>> from fractions import Fraction
>>> Fraction(1, 3) * Fraction(2, 3)
Fraction(2, 9)
```

Аргументы с плавающей точкой могут быть неточными, поэтому вы можете использовать модуль Decimal вместе с модулем Fraction:

Получим наибольший общий делитель для двух чисел с помощью функции gcd:

```
>>> import fractions
>>> fractions.gcd(24, 16)
8
```

#### Контрольные вопросы

- 1. Каково назначение модуля math? Его основные функции?
- 2. Функции округления, особенности их использования.
- 3. Константы модуля math.
- 4. Как осуществляется работа с комплексными числами в Python?
- 5. Использование модуля decimal.
- 6. Использование модуля fractions.

# Список литературы, рекомендуемый к использованию по данной теме:

- 1. Орлов, С. А. Программная инженерия. Технологии разработки программного обеспечения: учебник / С.А. Орлов. 5-е изд., обновл. и доп. СПб.: Питер, 2017. 640 с.
- 2. Гагарина, Л. Г. Современные проблемы информатики и вычислительной техники : [учеб. пособие] / Л.Г. Гагарина, А.А. Петров. М. : Форум, 2016. 368 с.
- 3. Шкляр, М. Ф. Основы научных исследований : учеб. пособие / М.Ф. Шкляр. 6-е изд. М. : Дашков и Ко, 2016. 208 с.
- 4. Михеева, Е. В. Информационные технологии в профессиональной деятельности : учеб. пособие / Е.В. Михеева. 14-е изд., стер. М. : Академия, 2016. 384 с.
- 5. Любанович Билл. Простой Python. Современный стиль программирования. СПб.: Питер, 2016. 480 с.

# Лабораторная работа 18. Модуль для научных расчетов NumPy. Работа с массивами.

# Цель работы:

Изучить особенности создания массивов в NumPy. Создание массива с помощью функций zeros(), ones() и random().Изменение формы массива с помощью метода reshape(). Получение элемента с помощью конструкции [].Математика массивов. Линейная алгебра.

# Компетенции:

Код	Формулировка:
ОК-5	быть готовым работать с информацией в различных формах,
	использовать для ее получения, обработки, передачи, хранения и
	защиты современные компьютерные технологии
ОПК-5	способностью использовать основные приемы обработки и
	представления экспериментальных данных
ОПК-6	способность осуществлять поиск, хранение, обработку и анализ
	информации из различных источников и баз данных, представлять ее в
	требуемом формате с использованием информационных, компьютерных
	и сетевых технологий