1. Электронно-вычислительная машина. Устройство ЭВМ. Программа. Исходный текст, исполняемый файл.

Электронно-вычислительная машина- основной вид реализации компьютером, который технически выполнен на электронных элементах

Исполняемый файл-файл, содержащий программу в виде, в котором она может быть исполнена компьютером( то есть в машинном коде).Получение исполняемых файлов требует выполнения компиляции .

Исходный текст программы – синтаксическая единица, которая соответствует правилам определенного языка программирования, состоящая из определений и операторов или инструкций, необходимых для определенной функции , задачи или решения проблемы.

1. Схемы алгоритмов
2. Языки программирования. Классификация.

Язык программирования – формальный язык, предназначенный для записи компьютерных программ. ЯП определяет набор лексических, синтаксических и семантических правил, определяющих действия , которые выполнит ЭВМ под её управлением.

Способы классификации :

* По уровню абстракции от аппаратной части
  + Низкоуровневые
  + Высокоуровневые
* По способу выполнения готовой программы
  + Компилируемые
  + Интерпретируемые
* По используемой парадигме программирования
  + Императивные/процедурные языки
  + Аппликативные/функциональные языки
  + Языки системы правил/декларативные языки
  + Объектно-ориентированные языки

4.Язык Python. Структура программы. Лексемы языка.

Питон- высокоуровневый язык программирования общего назначения. Интепретируемый, Является полностью объектно-ориентированным .

Программа

Модули  
 операторы  
 выражения  
 объекты

Символы алфавита любого яп образуют лексемы. По умолчанию- кодировка UTF-8.

Лексема – это минимальная единица языка, имеющая самостоятельный смысл . Лексемы формируют базовый словарь языка, понятный компилятору.

Существует 5 видов лексем:

* Ключевые слова
* Идентификаторы
* Литералы
* Операции
* Знаки пунктуации

5. Типы данных языка Python. Классификация. Скалярные типы данных. Приведение типов.

Основные типы данных Питон.

* Числа
* Строки
* Списки
* Словари
* Кортежи
* Файлы
* Множества
* Прочие основные типы
* Типы программных единиц
* Типы, связанные с реализацией

Способы классификации типов данных :

* Простые и сложные(составные)(массивы, записи, файлы)
* Скалярные и нескалярные (агрегатные типы данных)
* Самостоятельные и зависимые(например, ссылки)

**Скалярные (неделимые).**

* Числа (целое, вещественное).
* Логический тип.
* NoneType.

Приведение типов данных бывает явное и неявное.

Неявное это когда питон сам преобразует тип данных без участия пользователя, например 123+3.14 в результате сложения int и float мы получим float

Явное это когда пользователь сам заменяет тип данных объекта на требуемый .  
Для этого существуют функции int(),float(),str() и т.д.

6.Операции над скалярными типами данных. Приоритеты операций.

Операции над числами (сумма, разность, изменения знака числа, умножение, деление, деление нацело или с остатком, взятие в модуль, сравнение)

NoneType обозначающее нейтральное или «нулевое» поведение. Наиболее часто используется для защитного программирования - «если что-то не None, можно продолжать работу программы».

Логический тип данных: not and or операции соответствующие логическим таблицам истинности.

Приоритеты операций: возведение в степень, умножение деление взятие остатка, сложение вычитание, побитовое И, побитовое исключающее ИЛИ, побитовое ИЛИ.

7.Функции ввода и вывода.

Ввод функция input([prompt])

n=input(‘Введите натуральное число: ’)

Вывод функция print(\*objects, sep=’’,end=’\n’,file=sys.stdout,flush=False)

8.Функция вывода. Форматирование вывода.

Вывод функция print(\*objects, sep=’’,end=’\n’,file=sys.stdout,flush=False)

9.Оператор присваивания. Множественное присваивание.

Оператор присваивания предназначен для связывания имен со значениями и для изменения атрибутов или элементов изменяемых объектов. Оператор присваивания связывает переменную с объектом. Обозначается «=»

Обычная ( каноническая ) форма : name = value

Множественное присваивание: a,b,c=value1,value2,value3

list1 = [5, 3, 1]

a, b, c = list1

10.Условный оператор. Полные условные операторы. Неполные условные

операторы. Тернарный оператор условия. Примеры использования.

Условный оператор:

If выржение1:  
 Блок1  
elif выражение2:  
 Блок2  
elif выражение3:  
 Блок2  
else:  
 Блок4

Полный условный оператор:

If x>y:  
 print(‘x больше y’)  
elif x==y:  
 print(‘х равен у’)  
else:  
 print(‘х меньше у’)

неполный условный оператор:

m=x

If x<y:  
 m=y

Тернарные операторы наиболее широко известны в Python как условные выражения. Эти операторы возвращают что-то в зависимости от того, является ли условие истиной или ложью. Они стали частью языка с версии 2.4. Имеет самый низкий приоритет среди всех операций языка

X if c else y

Значение if условие else значение

num=int(input('введите целое число: '))  
n=''  
n="Число четное" if num % 2 == 0 else "число нечетное"  
print(num,n)

11.Условные операторы. Множественный выбор. Вложенные операторы условия.

Примеры использования.

Условный оператор:

If выржение1:  
 Блок1  
elif выражение2:  
 Блок2  
elif выражение3:  
 Блок2  
else:  
 Блок4

Множественный выбор

ball = int(input())

if ball < 50 : print(2)

else :

    if ball < 75 : print(3)

    else :

        if ball < 95 : print(4)

        else : print(5)

Вложенные операторы условия.

if A > 10:

**if A > 100:**

**print("У вас очень много денег.")**

**else:**

**print("У вас достаточно денег.")**

else:

print("У вас маловато денег.")

12.Операторы цикла. Цикл с условием. Операторы break и continue. Примеры

использования.

Цикл-разновидность управляющей конструкции в высокоуровневых ЯП , предназначенная для организации многократного исполнения набора инструкций .

Разновидности:

* бесконечный цикл
* цикл с предусловием
* цикл с постусловием
* цикл со счетчиком

В питоне существует for и while

Цикл с условием while

while условие:  
 операторы 1  
else:   
 операторы 2

Оператор break переходит за пределы ближайшего заключающего цикла (после всего оператора цикла)

Оператор continue переходит в начало ближайшего заключающего цикла (в строку заголовка цикла)

13.Операторы цикла. Цикл с итератором. Функция range(). Примеры

использования.

Цикл-разновидность управляющей конструкции в высокоуровневых ЯП , предназначенная для организации многократного исполнения набора инструкций .

Разновидности:

* бесконечный цикл
* цикл с предусловием
* цикл с постусловием
* цикл со счетчиком

В питоне существует for и while

Цикл for применим к любому итерируемому объекту

For переменная in объект:  
 операторы1  
else:  
 операторы2

Функция range()

Range(start=0,stop,step=1)  
Порождает серию целых чисел start<=n<stop с шагом step

14.Изменяемые и неизменяемые типы данных.

Типы данных бывают изменяемые и неизменяемые. К неизменяемым относятся целые числа (int), числа с плавающей запятой (float), булевы значения (bool), строки (str), кортежи (tuple). К изменяемым — списки (list), множества (set), байтовые массивы (byte arrays) и словари (dict).

15.Списки. Основные функции, методы, операторы для работы со списками.

Срезы.

16.Списки. Создание списков. Списковые включения.

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

Чтобы использовать списки, их нужно создать. Создать список можно несколькими способами.   
numbers=[1,1,2,3,5,8,13,21]  
a=list()  
а=[] пустой список

Списковые включения в Python являются краткими синтаксическими конструкциями. Их можно использовать для создания списков из других списков, применяя функции к каждому элементу в списке.

Списковые включения (list comprehension)

a=[i\*i for i in range(10)]  
a=[i for I in range(10) if i%2==0]

17.Списки. Основные методы для работы с элементами списка. Добавление

элемента, вставки, удаление, поиск.

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

list.append(x)-добавление элемента в конец списка  
insert(i,x)- Вставляет на i-ый элемент значение x

list.remove(x)- Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует

list.pop([i])- Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент

list.index(x, [start [, end]])- Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)

list.count(x)- Возвращает количество элементов со значением x

18.Списки. Основные операции со списками. Поиск минимального элемента. Поиск

максимального элемента. Нахождение количества элементов. Нахождение

суммы и произведения элементов.

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

Для работы со списками используются следующие основные операции:

+ – конкатенация списков;

\* – дублирование списка;

len – определение длины списка;

del – удаление элемента с списка;

присваивание по индексу;

использование среза для доступа к элементам списка и изменения списка.

min(list)-поиск минимального элемента

max(list)-поиск максимального элемента

len(list)-длина списка или же количество элементов

sum(list)-сумма элементов списка

prod(list)- произведение элементов списка from math import prod

19.Списки. Использование срезов при обработке списков.

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

[start:stop:step] - возвращает элементы списка, начиная с индекса start до stop с шагом step.

a[:] - все элементы списка  
a[5:] - с элемента с индексом 5 до конца   
a[:2] - элементы 0 и 1  
a[::-1] - разворачивание списка

20.Кортежи. Основные функции, методы, операторы для работы с кортежами.

Кортеж-неизменяемая последовательность.

Операторы: in, not in

Создание: a, b = tuple(), (1, 2, 3)

Операторы - аналогично спискам (+, \*)

Функции - аналогично спискам (all, enumerate, len, max, min, reversed, sorted, sum)

Методы - index(x) и count(x)

21.Словари. Понятие ключей и значений. Создание словарей. Основные функции,

методы, операторы для работы со словарями.

Словари-неупорядоченные коллекции произвольных объектов с доступом по ключу . В других языках-ассоциативные массивы, хэш-таблицы.

Элементы записываются в виде key:value. Ключи должны быть хэшируемыми и сравнимыми . Ключи словаря могут быть любым неизменяемым типом данных (числа, строки, кортежи и тд).

Создание:   
a, b = dict(), {‘odd’: 1, ‘even’: 2}   
c = dict(short='dict', long='dictionary')  
d = dict([(1, 1), (2, 4)])  
e = {a: a \*\* 2 for a in range(7)}

f = dict.fromkeys(['a', 'b'], 100)

Операторы : del, in, not in, |, |= (с 3.9).

Функции - те же, но применяются к ключам. Ключи должны быть либо одного типа данных, либо целые.

Методы словарей:

* clear() - очищает словарь (удаляет все элементы)
* copy() - создаёт “мелкую” копию
* fromkeys(iterable[, value]) - создаёт словарь на основе ключей и значении по умолчанию.
* метод класса
* get(key[, default]) - возвращает значение по ключу либо default либо None.
* items() - возвращает отображение содержимого
* keys() - возвращает отображение ключей
* pop(key[, default]) - удаляет значение из словаря и возвращает его, либо возвращает default,
* либо порождает исключение KeyError
* popitem() - возвращает последнюю добавленную в словарь пару либо порождает исключение KeyError
* setdefault(key[, default]) - значение по умолчанию для метода get на случай отсутствия ключа
* update([other]) - обновляет значение по другому словарю, кортежу и т.п.
* values() - возвращает отображение значений

22.Множества. Основные функции, методы, операторы для работы с множествами.

Множество в python - "контейнер", содержащий не повторяющиеся элементы в случайном порядке. Элементы должны быть хэшируемыми .

Единственное отличие set от frozenset заключается в том, что set - изменяемый тип данных, а frozenset - нет. Примерно похожая ситуация с [списками](https://pythonworld.ru/tipy-dannyx-v-python/spiski-list-funkcii-i-metody-spiskov.html) и [кортежами](https://pythonworld.ru/tipy-dannyx-v-python/kortezhi-tuple.html).

Создание: a, b = set(), {1,2,3}

Операторы: in, not in, <=, <, >=, >, |, &, -, ^.

Функции: len(s)

Только для set: |=, &=, -=, ^=

Методы для множеств :

isdisjoint(other) - нет пересечения   
issubset(other) - является подмножеством, <=   
issuperset(other) - является надмножеством, >=   
union(\*others)  
intersection(\*others)  
difference(\*others)  
symmetric\_difference(other)  
copy()

23.Строки. Основные функции, методы, операторы для работы со строками.

Срезы.

Строка - тип данных, значениями которого является произвольная последовательность символов. Обычно реализуется как массив символов.

Конкатенация +  
Дублирование \*  
Длина строки len()  
Доступ по индексу s[‘ Номер индекса ’]  
Извлечение среза:

Оператор извлечения среза: [X:Y]. X – начало среза, а Y – окончание;

символ с номером Y в срез не входит. По умолчанию первый индекс равен 0, а второй - длине строки.

in, not in

Методы работы со строками

●  capitalize() - переводит первую букву в верхний регистр

●  casefold() - один из способов перевода в нижний регистр

●  center(width[,fillchar]) - центрирует строку, дополняя пробелами с двух сторон

●  count(sub[,start[,end]]) - считает неперекрываюшиеся вхождения подстроки в

строку

●  encode(encoding=’utf-8’,erros=’strict’) - кодирование в заданную кодировку

●  endswith(suffix[,start[,end]]) - проверка на окончание одним из суффиксов

●  expandtabs(tabsize=8) - замена табуляций на пробелы

●  find(sub[,start[,end]]) - поиск подстроки в строке

●  format()

index(sub[,start[,end]]) - аналогично find, но порождает исключение ValueError, если вхождений нет

●  isalnum() - если все символы буквенно-цифровые и строка не пустая

●  isalpha() - если все символы - буквенные и строка не пустая

●  isascii() - если все символы из таблицы ASCII

●  isdecimal() - если символы цифровые в 10-й с/с и строка не пустая

●  isdigit() - если символы цифровые в 10-й с/с и строка не пустая

●  isidentifier() - является корректным идентификатором

●  islower() - все символы в нижнем регистре и строка не пустая

●  isnumeric() - все символы являются “числовыми” и строка не пустая

Методы работы со строками, часть 3

●  isprintable() - все символы “печатные” или строка пустая

●  isspace() - все символы “пробельные” и строка не пустая

●  istitle() - все символы в верхнем регистре и строка не пустая

●  isupper() - все символы в верхнем регистре и строка не пустая

●  join(iterable) - конкатенирует строки

●  ljust(width[, fillchar]) - дополняет пробелами справа до заданной ширины

●  lower() - переводит в нижний регистр

●  lstrip([chars]) - удаляет символы слева

●  partition(sep) - разделяет строку на части по первому вхождению разделителя, возвращает кортеж из 3-х элементов

Методы работы со строками, часть 4

●  removeprefix(prefix) - удаляет префикс

●  removesuffix(suffix) - удаляет суффикс

●  replace(old, new[, count]) - заменяет все вхождения подстроки

●  rfind(sub[,start[,end]]) - ищет подстроку справа

●  rindex(sub[,start[,end]]) - ищет подстроку справа с исключением

●  rjust(width[,fillchar]) - дополняет пробелами слева до ширины

●  rpartition(sep) - делит по разделителю, поиск справа

●  rsplit(sep = None, maxsplit = -1) - возвращает список слов (частей), поиск справа

●  rstrip([chars]) - удаляет завершающие символы

●  split(sep=None, maxsplit=-1) - возвращает список слов (частей) по разделителю

page16image44582976

Методы работы со строками, часть 5

●  splitlines([keepends]) - делит на части по переводам строк

●  startswith(prefix[,start[,end]]) - если начинается с префикса

●  strip([chars]) - удаляет символы и из начала, и с конца

●  swapcase() - меняет регистр

●  title() - переводит первые буквы слов в верхний регистр

●  translate(table) - преобразование символов по таблице

●  upper() - переводит в верхний регистр

●  zfill() - дополняет строку нулями слева

24.Матрицы. Создание матрицы. Ввод и вывод матрицы. Выполнение операций с

элементами матрицы.

Матрицы

В математике - таблица чисел

В программировании - массив массивов (двумерный массив)

a = [[1, 2, 3], [4, 5, 6]]  
N x M:

N - количество строк, M - количество столбцов

Обращение к элементу: a[i][j] # i - строка, j - столбец

Создание матриц

a = [[0] \* m] \* n # неправильно!!! создастся n ссылок на 1-ю строку

a = [[0] \* m for i in range(n)] # правильно

Ввод-вывод матриц

|  |
| --- |
| n = int(input(‘Введите количество строк матрицы: ’))  m = int(input(‘Введите количество столбцов матрицы: ’))  a = []  for i in range(n):  a.append([])  for j in range(m):  a[i].append(int(input(‘Введите {}-й элемент {}-й строки: ’.format(i+1, j+1)))) |

25.Матрицы. Квадратные матрицы. Обработка верхне- и нижнетреугольных

матриц. Работа с диагональными элементами матрицы.

В математике - таблица чисел

В программировании - массив массивов (двумерный массив)

Квадратная матрицы-размером NxN

Элементы расположенные на главной диагонали называются  **диагональными элементами.**

Изображение выглядит как текст

Автоматически созданное описание

26.Отладка программы. Способы отладки.

* Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

При отладке требуется:

-узнавать текущие значения переменных

-выяснять, по какому пути выполнялась программа

Способы отладки:

-Использование отладочной печати

-Использование отладчика

\*\* Простейший инструмент отладки - отладочная печать (вывод)

27.Подпрограммы. Функции. Создание функции. Аргументы функции.

Возвращаемое значение.

Подпрограмма - поименованная или иным образом идентифицированная

отдельная функционально-независимая часть компьютерной программы (функции / процедуры)

Параметры подпрограммы - переменные, которые вызывающая программа

передаёт подпрограмме

Формальные параметры - те, которые объявлены при описании подпрограммы

Фактические параметры - те, которые передаются в подпрограмму при её вызове

Функция в python - объект, принимающий аргументы и возвращающий значение. Обычно функция определяется с помощью инструкции def.

Определим простейшую функцию:

**def** add(x, y):

**return** x + y

Инструкция return говорит, что нужно вернуть значение. В нашем случае функция возвращает сумму x и y.

1. Функции. Области видимости.

Функция в python - объект, принимающий аргументы и возвращающий значение. Обычно функция определяется с помощью инструкции def.

Область видимости - часть программы, в пределах которой идентификатор

остаётся связан с сущностью, которой он был назначен при объявлении

\*\* В Python пространство имён определяется по местоположению

присваивания этому имени какого-либо значения

Основы областей видимости в Python:

-имена, присвоенные внутри def, "видны" только в коде внутри этого

оператора; ссылаться на них извне функции нельзя

-имена внутри def не конфликтуют с переменными за пределами def

Области видимости:

1. Глобальная - если переменная объявлена за пределами всех def,

то она является "глобальной" в целом файле

2. Локальная - переменная, объявленная внутри def, будет локальной в своей функции

3. Нелокальная - переменная, объявленная внутри def, включающем другие def

4. Встроенная (built-in)

\*\* Оператор global делает имя внутри функции глобальным

\*\* Оператор nonlocal делает имя внутри функции нелокальным

1. Функции. Завершение работы функции. Рекурсивные функции. Прямая и   
   косвенная рекурсия.

Функция в python - объект, принимающий аргументы и возвращающий значение. Обычно функция определяется с помощью инструкции def.

Рекурсия - вызов подпрограммы из неё же самой:

-непосредственно - простая рекурсия

-через другие подпрограммы - косвенная рекурсия

\*\* Тело рекурсивной подпрограммы должно иметь не меньше двух альтернативных (условных) ветвей, хотя бы одна из которых должна быть терминальной

Виды рекурсивных вызовов:

-по количеству вызовов:

-линейная - в теле функции присутствует только один вызов самой себя

-нелинейная - в теле присутствует несколько вызовов

-по месту расположения рекурсивного вызова:

-головная - рекурсивный вызов расположен ближе к началу тела функции

-хвостовая (концевая) - рекурсивный вызов является последним оператором функции

Видов рекурсии существует несколько. Если функция вызывает себя непосредственно, то мы имеем дело с *прямой рекурсией*. Если же функция вызывает внутри себя другую, которая когда-то вызовет первую, то это уже *косвенная рекурсия*.

1. Функции высшего порядка. Замыкания.

Функция высшего порядка- та, которая принимает в качестве аргументов другие функции или возвращает другие функции.

Замыкания (closure) в программировании - функция первого класса, в теле которой  
присутствуют ссылки на переменные, объявленные вне тела этой функции в окружающем коде и не являющиеся её параметрами

1. lambda-функции.

Оператор lambda создаёт и возвращает объект функции, которой будет вызываться позднее, не присваивая ему имени

lambda аргумент1, аргумент2, ...: выражение, использующее аргументы

Используется для сокращения кода в тех местах, где включение оператора def не разрешено синтаксисом

1. Аннотации.

Аннотации - способ добавлять произвольные метаданные к аргументам функциии возвращаемому значению

1. Функции map, filter, reduce, zip.

Функция map - возвращает итератор, применяющий функцию к каждому элементу  
итерируемого объекта

Функция filter - применяет функцию к элементам итерируемого объекта кумулятивно (накопительно): сначала - к первым двум элементам (либо к отдельно заданному начальному значению и первому элементу), далее - к промежуточному результату и очередному значению

Функция zip - Соединяет элементы итерируемых объектов в кортежи

Функция reduce принимает 2 аргумента: функцию и последовательность. reduce() последовательно применяет функцию-аргумент к элементам списка, возвращает единичное значение. Обратите внимание в Python 2.x функция reduce доступна как встроенная, в то время, как в Python 3 она была перемещена в модуль functools.

1. Декораторы.

Декоратор - это функция, которая позволяет обернуть другую функцию

для расширения её функциональности без непосредственного изменения её кода

1. Знак “\_“.

-хранение значения последнего выражения в интерпретаторе

-игнорирование некоторых значений (при разыменовании кортежей и т.д.)

-задание специальных значений для имён переменных или функций

(\_name, \_\_name, \_\_name\_\_)

1. Модули. Способы подключения.

Модуль Python - отдельный файл с кодом, который можно повторно использовать

в других программах

Преимущества модулей:

-многократное использование кода

-разбиение пространства имён системы

-реализация разделяемых служб или данных

Подключить модуль можно с помощью инструкции import. После 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.

1. Модуль math. Основные функции модуля. Примеры использования функций.

Модуль math – один из наиважнейших в Python. Этот модуль предоставляет обширный функционал для работы с числами.

**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 \* 2i. Функция, обратная функции math.frexp().

**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.

1. Модуль time.

Предоставляет функции для работы со временем

Sleep(secs)- задержка в секундах  
time()-время эпохи Юникс, с 01.01.1970 00:00+00

1. Модуль random. Работа со случайными числами.

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

**random.seed**([X], version=2) - инициализация генератора случайных чисел. Если X не указан, используется системное время.

**random.getstate**() - внутреннее состояние генератора.

**random.setstate**(state) - восстанавливает внутреннее состояние генератора. Параметр state должен быть получен функцией getstate().

**random.getrandbits**(N) - возвращает N случайных бит.

**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.

1. Модуль copy. Способы копирования объектов различных типов. “Глубокая” и

“мелкая” копии.

Данный модуль предоставляет общие (поверхностная и глубокая) операции копирования.

**copy.copy**(x) - возвращает поверхностную копию x.

**copy.deepcopy**(x) - возвращает полную копию x.

Разница между поверхностным и глубоким копированием существенна только для составных объектов, содержащих изменяемые объекты (например, список списков, или словарь, в качестве значений которого - списки или словари):

* **Поверхностная копия** создает новый составной объект, и затем (по мере возможности) вставляет в него ссылки на объекты, находящиеся в оригинале.
* **Глубокая копия** создает новый составной объект, и затем рекурсивно вставляет в него копии объектов, находящихся в оригинале.

1. Исключения.

Исключения - тип данных, позволяющий классифицировать ошибки и обрабатывать их.

\*\* В процессе обработки одного исключения может произойти (или быть

выброшено через raise) другое исключение, так, что обработка исключений

будет выполняться по цепочке

1. Файлы. Программная обработка файлов. Понятие дескриптора. Виды файлов.

Файл - поименованное место на диске.

Файловый дескриптор - целое число, которое присваивается операционной системе каждому потоку ввода-вывода при его создании.

Виды файлов:  
-текстовые файл  
-структурированные (типизированные) форматы  
-бинарные файлы

1. Файлы. Режимы доступа к файлам.

Режимы доступа к файлам

|  |  |
| --- | --- |
| Режим | Обозначение |
| 'r' | открытие на чтение (является значением по умолчанию). |
| 'w' | открытие на запись, содержимое файла удаляется, если файла не существует, создается новый. |
| 'x' | открытие на запись, если файла не существует, иначе исключение. |
| 'a' | открытие на дозапись, информация добавляется в конец файла. |
| 'b' | открытие в двоичном режиме. |
| 't' | открытие в текстовом режиме (является значением по умолчанию). |
| '+' | открытие на чтение и запись |

1. Файлы. Текстовые файлы. Основные методы для работы.

Текстовый файл-компьютерный файл, содержащий текстовые данные.

f.close  
f.read()  
f.readline()  
f.readlines()  
f.write()  
f.writelines(lines)  
f.truncate(size)  
f,seek(offset)  
f.tell()

1. Файлы. Текстовые файлы. Чтение файла. Запись в файл. Поиск в файле.

f = open(‘in.txt’)  
content = f.read() # 1  
lines\_list = f.readlines() # 2   
for i in (range(5)): # 3   
 string = f.readline() for s in f: # 4   
 print(s)  
f.close()

f = open(‘in.txt’, ‘w’)   
f.write(‘abcd’) # 1   
f.writelines([‘1’,’abcd’]) # 2   
print(‘def’, file=f)   
f.close()

поиск в файле: либо считываем построчно и ищем либо считываем все данные в один список и ищем.

1. Файлы. Текстовые файлы. Итерационное чтение содержимого файла.

Итерационное чтение- конструкция with open() as f либо с помощью функции range()

1. Файлы. Бинарные файлы. Основные методы. Сериализация данных.

Бинарные файлы в отличие от текстовых хранят информацию в виде набора байт. Для работы с ними в Python необходим встроенный модуль pickle.

[Сериализация](https://ru.wikipedia.org/wiki/Сериализация) — процесс перевода какой-либо структуры данных в последовательность битов.

1. Файлы. Оператор with. Исключения.

Оператор with имеет конструкцию with open(‘filename’) as ‘name’: действия   
Этот вид более питоновский. И закрывать файл не надо.

Исключения - тип данных, позволяющий классифицировать ошибки и обрабатывать их.

\*\* В процессе обработки одного исключения может произойти (или быть

выброшено через raise) другое исключение, так, что обработка исключений

будет выполняться по цепочке

1. Типы данных bytes и bytearray. Байтовые строки. Конвертация различных типов

в байтовые строки и обратно.

bytes and bytearray - классы для представления бинарных данных, "байтовые строки"

\*\* Набор операторов и методов похож на аналогичный у обычных строк

1. Модуль struct.

Модуль struct – формирует двоичные структуры данных из переменных базовых типов данных, и распаковывают их обратно.

Функции:

●  pack(format, v1, v2, ...)

●  pack\_into(format, buffer, offset, v1, v2, ...)

●  unpack(format, buffer)

●  unpack\_from(format, /, buffer, offset=0)

●  iter\_unpack(format, buffer)

●  calcsize(format)

1. Модуль os. Основные функции.

Модуль os предоставляет множество функций для работы с операционной системой, причём их поведение, как правило, не зависит от ОС, поэтому программы остаются переносимыми.

●  chdir(path) смена текущей директории.

●  getcwd() - текущая рабочая директория.

●  fspath(path)

●  getenv(key), putenv(key, value)

●  getegid(), geteuid(),getlogin(), getpid()... текущий id процесса.

●  set...()....

●  fdopen, close, open, ...

●  stat(path)

●  listdir(path = ‘.’) - список файлов и директорий в папке.

● mkdir(path)- создаёт директорию. OSError, если директория существует.

* **os.remove**(path, \*, dir\_fd=None) - удаляет путь к файлу.

● removedirs(name)  
● **os.rename**(src, dst, \*, src\_dir\_fd=None, dst\_dir\_fd=None) - переименовывает файл или директорию из src в dst.

● **os.replace**(src, dst, \*, src\_dir\_fd=None, dst\_dir\_fd=None) - переименовывает из src в dst с принудительной заменой.

**os.rmdir**(path, \*, dir\_fd=None) - удаляет пустую директорию.

● scandir(path=’.’)  
● truncate(path, length)   
● system(command)  
● times()  
● kill(pid, sig)

1. Генераторы.
2. Модуль numpy. Обработка массивов с использованием данного модуля.
3. Модуль numpy. Работа с числами и вычислениями.
4. Модуль matplotlib. Построение графиков в декартовой системе координат.

Управление областью рисования.

1. Модуль matplotlib. Построение гистограмм и круговых диаграмм.
2. Списки. Сортировка. Сортировка вставками. Сортировка выбором.

1)выбирается наименьший элемент и ставится в позицию 1,  
2)из оставшихся выбирается наименьший и ставится в позицию 2,   
3)продолжать до конца списка.

def selectionsort(arr):  
 for i in range(len(arr)):   
 minind = i  
 for j in range(i+1,len(arr)):   
 if arr[j] < arr[minind]: minind = j   
 arr[i],arr[minind] = arr[minind],arr[i]   
 print(i,' : ', \*arr)   
 return arr

1. Списки. Сортировка вставками. Метод простых вставок. Метод вставок с

бинарным поиском. Вставки с барьером. Метод Шелла.

1. Списки. Сортировка. Обменные методы сортировки. Сортировка пузырьком.

Сортировка пузырьком с флагом. Метод шейкер-сортировки.

1. Списки. Сортировка. Метод быстрой сортировки.