1. Язык python. Основные определения. Ключевые слова.

**Python**[[7]](https://ru.wikipedia.org/wiki/Python#cite_note-7) — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций:

and as break class continue def global for from accept else if in is not or assert except import while with return true try nonlocal finally pass raise None False True

2. Операторы языка. Виды операторов. Приоритеты операторов.

А) Арифметические операторы (+,-,\*,/,%..)

Б) Операторы сравнения(==,!=,>,<,>=,<=)

В) Операторы присваивания(=,\*=,+=,-=,/=,\*\*=,//=)

Г) Побитовые операторы(&,|,^…)

Д) Логические операторы(and,or,not)

Е) Операторы членства(in,not in)

Ж) Операторы тождественности(is,is not)

Приоритет:

|  |  |
| --- | --- |
| ( ) | Скобки |
| \*\* | Возведение в степень |
| +x, -x, ~x | Унарные плюс, минус и битовое отрицание |
| \*, /, //, % | Умножение, деления, взятие остатка |
| +, - | Сложение и вычитание |
| <<, >> | Битовые сдвиги |
| & | Битовое И |
| ^ | Битовое исключающее ИЛИ (XOR) |
| | | Битовое ИЛИ |
| ==, !=, >, >=, <, <=, is, is not, in, not in | Сравнение, проверка идентичности, проверка вхождения |
| not | Логическое НЕ |
| and | Логическое И |
| or | Логическое ИЛИ |

3. Встроенные функции языка python.

**abs**(x) - Возвращает абсолютную величину (модуль числа).

ord()- числовой код символа

chr()- символ по числовому коду

**complex**([real[, imag]]) - преобразование к комплексному числу.format()

input(),

**float**([X]) - преобразование к числу с плавающей точкой. Если аргумент не указан, возвращается 0.0

**round**(X [, N]) - Округление до N знаков после запятой.

**int**([object], [основание системы счисления]) - преобразование к целому числу.map()

**len**(x) - Возвращает число элементов в указанном объекте.

**max**(iter, [args ...] \* [, key]) - Максимальный элемент последовательности.

**min**(iter, [args ...] \* [, key]) - Минимальный элемент последовательности.

4. Модуль random. Работа со случайными числами.

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

**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.

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

5. Функции ввода и вывода.

input() – возвращает текстовую строку

print(…,sep=’ ’,end=’\n’) - Функция print выводит все элементы, разделяя их значением sep, и завершает вывод значением end.

6. Функция вывода. Форматирование вывода.

print(…,sep=’ ’,end=’\n’) - Функция print выводит все элементы, разделяя их значением sep, и завершает вывод значением end.

print(’{:5d} or {:7.3f} or {:9.3e}’ . format())

|  |  |
| --- | --- |
| '%d', '%i', '%u' | Десятичное число. |
| '%o' | Число в восьмеричной системе счисления. |
| '%x' | Число в шестнадцатеричной системе счисления (буквы в нижнем регистре). |
| '%X' | Число в шестнадцатеричной системе счисления (буквы в верхнем регистре). |
| '%e' | Число с плавающей точкой с экспонентой (экспонента в нижнем регистре). |
| '%E' | Число с плавающей точкой с экспонентой (экспонента в верхнем регистре). |
| '%f', '%F' | Число с плавающей точкой (обычный формат). |
| '%g' | Число с плавающей точкой. с экспонентой (экспонента в нижнем регистре), если она меньше, чем -4 или точности, иначе обычный формат. |
| '%G' | Число с плавающей точкой. с экспонентой (экспонента в верхнем регистре), если она меньше, чем -4 или точности, иначе обычный формат. |
| '%c' | Символ (строка из одного символа или число - код символа). |
| '%r' | Строка (литерал python). |
| '%s' | Строка (как обычно воспринимается пользователем). |
| '%%' | Знак '%'. |

7. Условные операторы. Полные условные операторы. Неполные условные операторы. Тернарный оператор условия. Примеры использования.

if .. else – полный

If – неполный

m = x if x > y else y - Тернарный оператор условия

8. Условные операторы. Множественный выбор. Вложенные операторы условия. Примеры использования.

if….

elif….

else..

if ….:

if..

else

9. Операторы цикла. Оператор цикла с условием. Операторы break и continue. Примеры использования.

Циклы позволяют повторять некоторое действие в зависимости от соблюдения некоторого условия.

while условное\_выражение:

   инструкции

После ключевого слова while указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций, который идет далее.

Оператор continue переводит цикл на следующую итерацию, пропуская всё "оставшееся" тело цикла.

Оператор break завершает цикл.

10. Операторы цикла. Оператор цикла с итератором. Примеры использования.

Циклы позволяют повторять некоторое действие в зависимости от соблюдения некоторого условия.

for переменная in последовательность

range(a,b,h)

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

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

Основные функции модуля

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

**math.floor**(X) - округление вниз.

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

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

**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.pi** - pi = 3,1415926...

**math.e** - e = 2,718281...

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

**math.pow**(X, Y) - XY.

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

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

**math.factorial**(X) - факториал числа X.

12. Строки. Основные функции и методы работы со строками. Пример программы.

Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

len(S) - Длина строки

S.split() - Разбиение строки по разделителю

S.isdigit() - Состоит ли строка из цифр

S.isalpha() - Состоит ли строка из букв

S.islower() - Состоит ли строка из символов в нижнем регистре

S.isupper() - Состоит ли строка из символов в верхнем регистре

S.upper() - Преобразование строки к верхнему регистру

S.lower() - Преобразование строки к нижнему регистру

S.strip() - Удаление пробельных символов в начале и в конце строки

S[i:j:step] - Извлечение среза

13. Списки. Создание списков. Генераторы списков.

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

X = list()

Список можно создать и при помощи литерала:

И еще один способ создать список - это **генераторы списков**

Генератор списков - способ построить новый список, применяя выражение к каждому элементу последовательности. Генераторы списков очень похожи на цикл for.

L = input('Введите массив в одну строку: ').split()

Y = [k for k in range(7)]

Y = [k for k in range(7) if k % 3]

L = [L\*2 for L in ‘IU7’]

14. 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).

L.append(x) - Добавляет элемент x в конец списка

L.extend(s) – Расширяет список L, добавляя в конец все элементы списка s

L.reverse() - Разворачивает список

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

L.insert(i, x) - Вставляет на i-ый элемент значение x

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

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

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

15. 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).

min(A) - Наименьший элемент списка A

max(A) - Наибольший элемент списка A

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

sum(A) - Возвращает сумму элементов в списке A

16. Списки. Использование срезов при обработке списков.

|  |  |
| --- | --- |
| R = x[:] | создает копию списка: [1,2,3,4,5] |
| R1 = x[0:3] | формируется список c 0-го элемента х по : x[0], x[1], x[2]: [1,2,3] |
| R2 = x[1:] | выбираются все элементы, кроме первого: [2,3,4,5] |
| R3 = x[:3] | первые 3 элемента массива: [1,2,3] |
| R4 = x[:-1] | все элементы, кроме последнего: [1,2,3,4] |
| R5 = x[-3:] | три последних элемента: [3,4,5] |
| R6 = x[1:-2] | отсутствуют первый элементы и два последних: [2,3] |

17. Списки. Сортировка элементов списка.

Множество в python - "контейнер", содержащий не повторяющиеся элементы в случайном порядке.

**Словари в Python** - неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда ещё называют ассоциативными массивами или хеш-таблицами.

. Сортировать можно при помощи функции sort(). Также существуют разные методы сортировок:

· Сортировка выбором

· Пузырьковая сортировка

· Пузырьковая сортировка с флагом

· Метод шейкер-сортировки

· Сортировка вставками

· Сортировка вставками с барьером

· Сортировка вставками с бинарным поиском

· Сортировка Шелла

· Быстрая сортировка

18. Списки. Сортировка. Сортировка вставками. Сортировка выбором.

**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**

s1=[4,8,2,5,9,1]

**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**

**selectionsort(s1)**

19. Списки. Сортировка вставками. Метод простых вставок. Метод вставок с бинарным поиском. Метод Шелла.

**Сортировка вставками**

1. часть массива, расположенная левее текущего элемента, считается отсортированной
2. в отсортированной части массива ищется место для вставки текущего элемента
3. вставка элемента в найденную позицию

s5=[10,-4,-3,-2,-1,4,8,2,9,5,1]

def insertionsort(arr):

for i in range(1,len(arr)):

cur=arr[i]

j=i-1

while j>=0 and cur<arr[j]:

arr[j+1]=arr[j]

j=j-1

arr[j+1]=cur

print(i,':',arr[:i],'|',arr[i:])

return arr

insertionsort(s5)

Метод Шелла:

Сортировка вставками, осуществляемая с определенным шагом

s7=[4,8,2,9,5,1]

def Shellsort(arr):

inc=len(arr)//2

while inc:

for i,el in enumerate(arr):

while i>=inc and arr[i-inc]>el:

arr[i]=arr[i-inc]

i-=inc

arr[i]=el

print(i,':',\*arr,'|',inc)

if inc==2:

inc=1

else:

inc=int(inc\*5.0/11)

return arr

Shellsort(s7)

Метод пузырька:

1. сравниваются 2 соседних элемента
2. если нарушен порядок, то они обмениваются

s2=[4,8,2,9,5,1]

**def bubblesort(arr):**

**n=len(arr)**

**for i in range(n-1):**

**for j in range(n-1-i):**

**if arr[j]>arr[j+1]:**

**arr[j],arr[j+1]=arr[j+1],arr[j]**

**print(i,':',\*arr)**

**return arr**

**bubblesort(s2)**

**Метод Вставками с бинарным поиском:**

**s6=[4,8,2,9,5,1]**

**def insertionsortwithbinsearch(arr):**

**for i in range(1,len(arr)):**

**cur=arr[i]**

**lo=0**

**hi=i**

**if lo==hi:lo+=1**

**else:**

**while lo<hi:**

**mid=(lo+hi)//2**

**if cur<arr[mid]:hi=mid**

**else:lo=mid+1**

**j=i**

**while j>lo and j>0:**

**arr[j]=arr[j-1]**

**j=j-1**

**arr[lo]=cur**

**print(i,':',\*arr)**

**return arr**

**insertionsortwithbinsearch(s6)**

20. Списки. Сортировка. Пузырьковые методы сортировки. Сортировка пузырьком. Сортировка пузырьком с барьером. Метод шейкер-сортировки.

Шейкер-сортировка:

Это пузырьковая сортировка, работающая в 2 прохода

1. прямой
2. обратный

s4=[10,-4,-3,-2,-1,4,8,2,9,5,1]

**def shakersort(arr):**

**left=0**

**right=len(arr)-1**

**print(left,right)**

**while left<right:**

**r\_new=left**

**for i in range(left,right):**

**if arr[i]>arr[i+1]:**

**arr[i],arr[i+1]=arr[i+1],arr[i]**

**r\_new=i**

**right=r\_new**

**print(\*arr,'|',left,right)**

**l\_new=right**

**for i in range(right-1,left-1,-1):**

**if arr[i]>arr[i+1]:**

**arr[i],arr[i+1]=arr[i+1],arr[i]**

**l\_new=i**

**left=l\_new**

**print(\*arr,'|',left,right)**

**return arr**

**shakersort(s4)**

21. Списки. Сортировка. Метод быстрой сортировки.

Быстрая сортировка:

import random as r  
L = [5,4,3,2,10,2,0]  
def quicksort(arr,start=0,end=None):  
 if len(arr)==0:  
 return arr  
 pind=r.randint(start,end-1)  
 pivot=arr[pind]  
 left=[x for x in arr if x<pivot]  
 right=[x for x in arr if x>pivot]

equal=[x for x in arr if x==pivot]  
 print('pivot: ',pind,pivot)  
 print('left: ',left)  
 print('right: ',right)  
 return quicksort(left,0,len(left))+equal+quicksort(right,0,len(right))  
a=quicksort(L,0,len(L))  
print(a)

22. Множества. Определение. Основные операции.

Множество в python - "контейнер", содержащий не повторяющиеся элементы в случайном порядке.

a = set() – создание множества

Операции:

* len(s) - число элементов в множестве (размер множества).
* x in s - принадлежит ли x множеству s.
* **set.isdisjoint**(other) - истина, если set и other не имеют общих элементов.
* **set == other** - все элементы set принадлежат other, все элементы other принадлежат set.
* **set.issubset**(other) или **set <= other** - все элементы set принадлежат other.
* **set.union**(other, ...) или **set | other | ...** - объединение нескольких множеств.
* **set.intersection**(other, ...) или **set & other & ...** - пересечение.
* **set.difference**(other, ...) или **set - other - ...** - множество из всех элементов set, не принадлежащие ни одному из other.
* **set.copy**() - копия множества.
* **set.update**(other, ...); set |= other | ... - объединение.
* **set.intersection\_update**(other, ...); set &= other & ... - пересечение.
* **set.difference\_update**(other, ...); set -= other | ... - вычитание.
* **set.symmetric\_difference\_update**(other); set ^= other - множество из элементов, встречающихся в одном множестве, но не встречающиеся в обоих.
* **set.add**(elem) - добавляет элемент в множество.
* **set.remove**(elem) - удаляет элемент из множества. KeyError, если такого элемента не существует.
* **set.discard**(elem) - удаляет элемент, если он находится в множестве.
* **set.pop**() - удаляет первый элемент из множества. Так как множества не упорядочены, нельзя точно сказать, какой элемент будет первым.
* **set.clear**() - очистка множества.

Set() – изменяемый кортеж, frozenset() – неизменяемый.

23. Словари. Понятие ключей и значений. Создание словарей. Основные методы словарей.

**Словари в Python** - неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда ещё называют ассоциативными массивами или хеш-таблицами.

Создавать словари можно при помощи ‘{}’, ‘dict’. Также можно использовать метод fromkeys (d = dict.fromkeys([‘a’,’b’]). Можно также использовать генераторы: d = {a: a \*\*2 for a in range(7)}

Методы:

**dict.clear**() - очищает словарь.

**dict.copy**() - возвращает копию словаря.

classmethod **dict.fromkeys**(seq[, value]) - создает словарь с ключами из seq и значением value (по умолчанию None).

**dict.get**(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None).

**dict.items**() - возвращает пары (ключ, значение).

**dict.keys**() - возвращает ключи в словаре.

**dict.pop**(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).

**dict.popitem**() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены.

**dict.setdefault**(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ с значением default (по умолчанию None).

**dict.update**([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).

**dict.values**() - возвращает значения в словаре.

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

# Создание матрицы

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

a = [[0] \* m] \* n]

a = [[0] \* m for I in range(n)]

# Ввод двумерного массива

n = int(input())

a = []

for i in range(n):

a.append([int(j) for j in input().split()])

ИЛИ:

n = int(input())

a = []

for i in range(n):

row = input().split()

for i in range(len(row)):

row[i] = int(row[i])

a.append(row)

ИЛИ при помощи ГЕНЕРАТОРА:

n = int(input())

a = [[int(j) for j in input().split()] for i in range(n)]

# Вывод двумерного массива:

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

for i in range(len(a)):

for j in range(len(a[i])):

print(a[i][j], end=' ')

print()

ИЛИ:

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

for row in a:

for elem in row:

print(elem, end=' ')

print()

25. Матрицы. Квадратные матрицы. Обработка верхне- и нижнетреугольных матриц. Работа с диагональными элементами матрицы.

i > j – элементы расположены ниже главной диагонали;  
i < j – элементы расположены выше главной диагонали;  
i + j< n+1– элементы расположены над побочной диагональю;  
i + j> n+1– элементы расположены под побочной диагональю;

26. Функции. Создание функции. Аргументы функции. Возвращаемое значение.

Функции. Создание функции. Аргументы функции. Возвращаемое значение.

При помощи def():

def add(x, y):

return x + y

27. Функции. Lambda-функции. Рекурсивные функции.

Lambda-функции.

Func = lambda x, y: x + y

Func (1,2)

(lambda x, y: x + y)(1,2)

28. Файлы. Текстовые файлы. Открытие файла. Режимы доступа к файлу.

F = open(‘text.txt’, ‘r’)

Режимы:

'r’ – открытие на чтение

‘w’ – открытие на запись

‘x’ – открытие на запись, если файла не существует

‘a’ – открытие на дозапись

‘b’ – открытие в двоичном режиме

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

# Чтение файла

F = open(‘text.txt’)

f=F.read()

print(f)

# Запись в файл

L = [str(i) + str(i-1) for I in range(20)]

F = open(‘text.txt’, ‘w’)

For index in L:

F.write(index + ‘\n’)

F.close()

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

F = open(‘text.txt’,’r’)

For I in F:

Print(i)

31. Файлы. Бинарные файлы. Сериализация данных. Модуль pickle.

Файлы. Бинарные файлы. Сериализация данных. Модуль pickle.

File = open(‘text.txt’,’wb’)

Dict1 = {‘0’: kek, ‘1’:mem}

dump( Dict1,file) – записывает сериализованный объект в файл

File = open(‘text.txt’,’rb’)

load(file) – загружает объект из файла

32. Модуль numpy. Обработка массивов с использованием данного модуля.

A = np.array([20,30,40,50])

B = np.arange(4)

A + B 20,31,42,53

A – B 20,29,38,47

A \* B 0 30 80 150

A / B inf,30. , 20. , 16.6666667

A \*\* B 1, 30, 1600 , 125000

A % B 0,0,0,2

A + 1 21,31,41,51

A < 35 True True False False

np.cos(A) 0.408.. 0.154… -0.666… 0.964…

A = np.array([[1,2,3],[4,5,6]])

np.sum(A) 21

A.sum() 21

A.min 1

A.max 6

A.min(axis=0) array([1,2,3]) - Наименьшее число в каждом столбце

A.max(axis=1) array([1,4]) - Наименьшее число в каждой строке

Есть срезы.

A = np.array([[1,2],[3,4]])

B = np.array([[5,6],[7,8]])

np.vstack((a,b))

array([[1,2],

[3,4],

[5,6],

[7,8]])

np.hstack((a,b))

array([[1,2,5,6],

[3,4,7,8]])

column\_stack((a,b)) – объединяет одномерные массивы в качестве двумерных столбцов

row\_stack((a,b)) – в качестве строк

33. Модуль numpy. Работа с числами и вычислениями.

numpy - библиотека языка Python ориентирована на работу с многомерными массивами и матрицами, с полиномами и с другими объектами.

34. Модуль matplotlib. Построение графиков в декартовой системе координат. Управление областью рисования.

Пакет matplotlib является основным для визуализации расчетных данных

Для рисования графика используем модуль pyplot из пакета

import matplotlib.pyplot as plt

X=[0,6,11,17,24,31,38,40]

Y=[0,2,13,10,16,12,20,17]

plt.plot(X,Y)

plt.show()

35. Модуль matplotlib. Построение гистограмм и круговых диаграмм.

Гистограмм:

Import matplotlib.pyplot as plt

Import numpy as np

X=np.random.normal(0,3,1000) #np.random.randn(1000)

plt.hist(X,25) #по умолчанию 10

plt.show()

круговых диаграмм:

import matplotlib.pyplot as plt

x=[6,12,20,7,5,5]

languages=[‘Matlab’,’Java’,’Python’,’C’,’C++’,’Other’]

plt.figure(figsize=(10,10))

explode=[0,0,0,0.1,0,0]

plt.pie(x,labels=languages,explode=explode,autopct=’%1.1f%%’,shadow=False)

plt.title(‘Circle diagram’)

plt.show()