ИНФОРМАТИКА

1 курс

**Типы данных в Python**

1. none - неопределенное значение переменной
2. bool - логические переменные
3. Числа
   1. *int* – целое число
   2. *float* – число с плавающей точкой
   3. *complex* – комплексное число
4. Списки
   1. *list* – список
   2. *tuple* – кортеж
   3. *range* – диапазон
5. Строки
   1. *str*
6. Бинарные списки
   1. *bytes* – байты
   2. *bytearray* – массивы байт
   3. *memoryview* – специальные объекты для доступа к внутренним данным объекта через protocol buffer
7. Множества
   1. *set* – множество
   2. *frozenset* – неизменяемое множество
8. Словари
   1. *dict* – словарь

* К неизменяемым (immutable) типам относятся: целые числа (int), числа с плавающей точкой (float), комплексные числа (complex), логические переменные (bool), кортежи (tuple), строки (str) и неизменяемые множества (frozen set).
* К изменяемым (mutable) типам относятся: списки (list), множества (set), словари (dict).

**Встроенные функции, выполняющие преобразование типов**

bool(x) 🡪…………………………………………………….преобразование к типу bool, использующая стандартную процедуру проверки истинности. Если х является ложным или опущен, возвращает значение False, в противном случае она возвращает True.

bytearray([источник [, кодировка [ошибки]]]) 🡪………….преобразование к bytearray. Bytearray - изменяемая последовательность целых чисел в диапазоне 0≤X<256. Вызванная без аргументов, возвращает пустой массив байт.

bytes([источник [, кодировка [ошибки]]]) 🡪……………...возвращает объект типа bytes, который является неизменяемой последовательностью целых чисел в диапазоне 0≤X<256. Аргументы конструктора интерпретируются как для bytearray().

complex([real[, imag]]) 🡪…………………………………….преобразование к комплексному числу.

dict([object]) 🡪………………………………………………..преобразование к словарю.

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

frozenset([последовательность]) 🡪……………………….....возвращает неизменяемое множество.

int([object], [основание системы счисления]) 🡪…………....преобразование к целому числу.

list([object]) 🡪создает список.

memoryview([object]) 🡪……………………………………….создает объект memoryview.

object() 🡪……………………………………………………….возвращает безликий объект, являющийся базовым для всех объектов.

range([start=0], stop, [step=1]) 🡪……………………………....арифметическая прогрессия от start до stop с шагом step.

set([object]) 🡪…………………………………………………..создает множество.

slice([start=0], stop, [step=1]) 🡪………………………………..объект среза от start до stop с шагом step.

str([object], [кодировка], [ошибки]) 🡪……………………..…строковое представление объекта. Использует метод \_\_str\_\_.

tuple(obj) 🡪………………………………………………..……преобразование к кортежу.

**Другие встроенные функции**

abs(x) 🡪………………………………...Возвращает абсолютную величину (модуль числа).

all(последовательность) 🡪…………....Возвращает True, если все элементы истинные (или, если последовательность пуста).

any(последовательность) 🡪 ………….Возвращает True, если хотя бы один элемент - истина. Для пустой последовательности возвращает False.

ascii(object) 🡪 …………………………Как repr(), возвращает строку, содержащую представление объекта, но заменяет не-ASCII символы на экранированные последовательности.

bin(x) 🡪…………………………………Преобразование целого числа в двоичную строку.

callable(x) 🡪…………………………….Возвращает True для объекта, поддерживающего вызов (как функции).

chr(x) 🡪…………………………………Возвращает односимвольную строку, код символа которой равен x.

classmethod(x) 🡪………………………..Представляет указанную функцию методом класса.

compile(source, filename, mode, flags=0, dont\_inherit=False) 🡪Компиляция в программный код, который впоследствии может выполниться функцией eval или exec. Строка не должна содержать символов возврата каретки или нулевые байты.

delattr(object, name) 🡪…………………..Удаляет атрибут с именем 'name'.

dir([object]) 🡪……………………………Список имен объекта, а если объект не указан, список имен в текущей локальной области видимости.

divmod(a, b) 🡪……………………………Возвращает частное и остаток от деления a на b.

enumerate(iterable, start=0) 🡪…………….Возвращает итератор, при каждом проходе предоставляющем кортеж из номера и соответствующего члена последовательности.

eval(expression, globals=None, locals=None) 🡪Выполняет строку программного кода.

exec(object[, globals[, locals]]) 🡪…………Выполняет программный код на Python.

filter(function, iterable) 🡪…………………Возвращает итератор из тех элементов, для которых function возвращает истину.

format(value[,format\_spec]) 🡪…………….Форматирование (обычно форматирование строки).

getattr(object, name ,[default]) 🡪…………..извлекает атрибут объекта или default.

globals()🡪…………………………………..Словарь глобальных имен.

hasattr(object, name) 🡪……………………..Имеет ли объект атрибут с именем 'name'.

hash(x) 🡪……………………………………Возвращает хеш указанного объекта.

help([object]) 🡪……………………………..Вызов встроенной справочной системы.

hex(х) 🡪……………………………………..Преобразование целого числа в шестнадцатеричную строку.

id(object) 🡪……………………………….....Возвращает "адрес" объекта. Это целое число, которое гарантированно будет уникальным и постоянным для данного объекта в течение срока его существования.

input([prompt]) 🡪……………………………Возвращает введенную пользователем строку. Prompt - подсказка пользователю.

isinstance(object, ClassInfo) 🡪………………Истина, если объект является экземпляром ClassInfo или его подклассом. Если объект не является объектом данного типа, функция всегда возвращает ложь.

issubclass(класс, ClassInfo) 🡪…………….....Истина, если класс является подклассом ClassInfo. Класс считается подклассом себя.

iter(x) 🡪………………………………………Возвращает объект итератора.

len(x) 🡪……………………………………….Возвращает число элементов в указанном объекте.

locals() 🡪……………………………………...Словарь локальных имен.

map(function, iterator) 🡪……………………..Итератор, получившийся после применения к каждому элементу последовательности функции function.

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

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

next(x) 🡪……………………………………….Возвращает следующий элемент итератора.

oct(х) 🡪………………………………………...Преобразование целого числа в восьмеричную строку.

open(file, mode='r', buffering=None, encoding=None, errors=None, newline=None, closefd=True) 🡪Открывает файл и возвращает соответствующий поток.

ord(с) 🡪…………………………………………Код символа.

pow(x, y[, r]) - ( x \*\* y ) % r.

reversed(object) 🡪………………………………Итератор из развернутого объекта.

repr(obj) 🡪………………………………………Представление объекта.

print([object, ...], \*, sep=" ", end='\n', file=sys.stdout) 🡪Печать.

property(fget=None, fset=None, fdel=None, doc=None)

round(X [, N]) 🡪…………………………………Округление до N знаков после запятой.

setattr(объект, имя, значение) 🡪……………….Устанавливает атрибут объекта.

sorted(iterable[, key][, reverse]) 🡪……………….Отсортированный список.

staticmethod(function) 🡪…………………………Статический метод для функции.

sum(iter, start=0) 🡪………………………………Сумма членов последовательности.

super([тип [, объект или тип]]) 🡪……………….Доступ к родительскому классу.

type(object) 🡪…………………………………….Возвращает тип объекта.

type(name, bases, dict) 🡪…………………………Возвращает новый экземпляр класса name.

vars([object]) 🡪…………………………………...Словарь из атрибутов объекта. По умолчанию - словарь локальных имен.

zip(\*iters) 🡪……………………………………….Итератор, возвращающий кортежи, состоящие из соответствующих элементов аргументов-последовательностей.

**Функция или метод строк**

S = 'str'; S = "str"; S = '''str'''; S = """str"""🡪………..Литералы строк

S = "s\np\ta\nbbb"🡪…………………………………Экранированные последовательности

S = r"C:\temp\new"🡪………………………………..Неформатированные строки (подавляют экранирование)

S = b"byte"🡪…………………………………………Строка байтов

S1 + S2🡪……………………………………………..Конкатенация (сложение строк)

S1 \* 3🡪………………………………………………Повторение строки

S[i] 🡪…………………………………………………Обращение по индексу

S[i:j:step] 🡪…………………………………………..Извлечение среза

len(S) 🡪………………………………………………Длина строки

S.find(str, [start],[end]) 🡪…………………………….Поиск подстроки в строке. Возвращает номер первого вхождения или -1

S.rfind(str, [start],[end]) 🡪……………………………Поиск подстроки в строке. Возвращает номер последнего вхождения или -1

S.index(str, [start],[end]) 🡪…………………………...Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError

S.rindex(str, [start],[end]) 🡪…………………………..Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError

S.replace(шаблон, замена,[maxcount]) 🡪…………...Замена шаблона на замену. maxcount ограничивает количество замен

S.split(символ) 🡪……………………………………...Разбиение строки по разделителю

S.isdigit()🡪…………………………………………….Состоит ли строка из цифр

S.isalpha()🡪……………………………………………Состоит ли строка из букв

S.isalnum()🡪…………………………………………...Состоит ли строка из цифр или букв

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

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

S.isspace()🡪…………………………………………….Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая строка" ('\n'), "перевод каретки" ('\r'), "горизонтальная табуляция" ('\t') и "вертикальная табуляция" ('\v'))

S.istitle()🡪………………………………………………Начинаются ли слова в строке с заглавной буквы

S.upper()🡪………………………………………………Преобразование строки к верхнему регистру

S.lower()🡪…………………………………………...….Преобразование строки к нижнему регистру

S.startswith(str) 🡪……………………………………….Начинается ли строка S с шаблона str

S.endswith(str) 🡪………………………………………..Заканчивается ли строка S шаблоном str

S.join(список) 🡪………………………………………...Сборка строки из списка с разделителем S

ord(символ) 🡪…………………………………………...Символ в его код ASCII

chr(число) 🡪……………………………………………..Код ASCII в символ

S.capitalize()🡪……………………………………………Переводит первый символ строки в верхний регистр, а все остальные в нижний

S.center(width, [fill]) 🡪…………………………………..Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)

S.count(str, [start],[end]) 🡪……………………………….Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)

S.expandtabs([tabsize]) 🡪…………………………………Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам

S.lstrip([chars]) 🡪………………………………………….Удаление пробельных символов в начале строки

S.rstrip([chars]) 🡪………………………………………….Удаление пробельных символов в конце строки

S.strip([chars]) 🡪…………………………………………..Удаление пробельных символов в начале и в конце строки

S.partition(шаблон) 🡪…………………………………….Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки

S.rpartition(sep) 🡪…………………………………………Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку

S.swapcase()🡪……………………………………………..Переводит символы нижнего регистра в верхний, а верхнего – в нижний

S.title()🡪……………………………………………………Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний

S.zfill(width) 🡪……………………………………………..Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями

S.ljust(width, fillchar=" ")🡪………………………………..Делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar

S.rjust(width, fillchar=" ")🡪………………………………...Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar

S.format(\*args, \*\*kwargs) 🡪……………………………….Форматирование строки

**Методы списков**

list.append(x) 🡪………………………………………...….….Добавляет элемент в конец списка

list.extend(L) 🡪………………………………………...……..Расширяет список list, добавляя в конец все элементы списка L

list.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

list.sort([key=функция]) 🡪………………………….………...Сортирует список на основе функции

list.reverse()🡪…………………………………………….……Разворачивает список

list.copy()🡪…………………………………………….………Поверхностная копия списка

list.clear()🡪…………………………………………….………Очищает список

**СЛОВАРИ**

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 со 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.