Python_4.1

ВСТРОЕННЫЕ ФУНКЦИИ

Built-in

Встроенные функции (Built-in) - это функции встроенные в интерпретатор Python и для их использования в программах не надо импортировать модули. В интерпретатор Python встроены следующие (Built-in) функции

Ознакомиться со всеми встроенными функциями можно по ссылке

https://docs.python.org/3/library/functions.html

Математические

abc(x) - возвращает абсолютное значение числа **x**;

pow(x, y) - возводит **x** в степень **y**

```
>>>abs(-5/10)
0.5


>>>pow(3, 2)
9
```

round()

round() - возвращает число округляя его дробную часть до значения аргумента;

```
>>> a = 10/3
>>> a
3.3333333333333335
>>> round(a,2)
3.33
>>> round(a)
3
```

```
>>> round(5321, -1)
5320
>>> round(5321, -3)
5000
>>> round(5321, -4)
10000
```

```
>>> round(3.76, 1)
3.8
>>> round(3.72, 1)
3.7
>>> round(3.72)
4
>>> round(3.22)
3
```

Если второй аргумент не задан, то округление идет до целого числа. Есть одна специфическая особенность этой функции. Второй аргумент может быть отрицательным числом. В этом случае округляться начинают единицы, десятки, сотни и т. д., то есть целая часть.

round()

Нередко функцию **round()** используют совместно с функцией **print()**, избегая форматирования вывода:

```
>>> a = 3.45673
>>> print("Number: %.2f" % a)
Number: 3.46
>>> print("Number:", round(a,2))
Number: 3.46
```

int(), divmod()

Если нужно просто избавиться от дробной части без округления, следует воспользоваться функцией **int()**:

```
>>> int(3.78)
3
```

Функция **divmod()** выполняет одновременно деление нацело и нахождение остатка от деления:

```
>>> divmod(10, 3)
(3, 1)
>>> divmod(20, 7)
(2, 6)
```

ord(), char()

ord(x) - возвращает номер символа **x** из таблицы Unicode;

chr(x) – возвращает символ, соответствующий переданному в качестве аргумента целому числу **x** из таблицы Unicode.

```
>>>ord('a')
99
>>>char(100)
d
```

Функции приведения типов

int() – возвращает целое число;

bool() – возвращает логическую интерпретацию переданных данных;

complex() – возвращает комплексное число;

float() – возвращает число с дробной частью;

frozenset() – возвращает неизменяемое множество;

list() – возвращает список;

str() – возвращает строку;

tuple() – возвращает кортеж;

set() – возвращает множество;

globals(), locals()

globals() - возвращает словарь глобальных переменных, текущей области,

locals() - возвращает словарь локальных переменных, текущей области.

locals()

```
x = 5

y = 10

def my_function():
  m = 5
  n = 4
  print(locals())

my_function()
```

{'m': 5, 'n': 4}

globals

```
x = 5

y = 10

def my_function():

  m = 5

  n = 4

  print(globals())

my_function()
```

```
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__':
<_frozen_importlib_external.SourceFileLoader object at 0x7f497d5b6970>, '__spec__': None,
'__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>, '__file__': 'main.py', '__cached__':
None, 'x': 5, 'y': 10, 'my_function': <function my_function at 0x7f497d59b3a0>}
```

Перевод в другую систему счисления

bin() - возвращает строку с числом преобразованным в двоичную форму;

hex() - возвращает строку с числом преобразованным в шестнадцатеричную форму;

oct() - возвращает строку с числом преобразованным в восьмеричную форму;

```
>>>bin(15)
0b1111
>>>oct(15)
0o17
```

Функции проверки

isinstance() - проверяет является ли аргумент экземпляром класса;

issubclass() - проверяет является ли аргумент подклассом;

type() - возвращает тип объекта;

```
class Thefirst:
  pass
class Thesecond(Thefirst):
  pass
print(issubclass(Thesecond, Thefirst))
```

True

sum(), min(), max()

max() - возвращает наибольшее значение;

min() - возвращает наименьшее значение;

sum() – возвращает сумму списка.

```
>>> s = ['1', '12', '120', '22']
>>> max(s)                  # максимальный в лексикографическом порядке'22'
>>> max(s, key=len)         # максимальный по длине. '120'
>>>min('c','a','b')         # 'a'
>>>sum([1, 2, 3, 4, 5]      #15
```

all(), any()

all() - проверяет отсутствие нуля в аргументе; аргумент – любой итерируемый объект, возвращает True или False

any() - проверяет наличие нуля в аргументе; Возвращает True если среди элементов аргумента хотя бы один элемент не равен нулю или False если все элементы аргумента равны нулю или False.

Есть ли в списке четные числа

```
>>> s = [7, 11, 120]
>>> def even(s):
    for element in s:
        if element % 2 == 0:
                return True
    return False
>>> even(s)
True
>>> any(element % 2 == 0 for element in s)  # этот вариант предпочтительнее
True
```

sorted()

sorted() – сортирует итерируемый объект

sorted(iterable, key=None, reverse=False)

key – функция, принимающая один аргумент, применяемая к каждому элементу итерируемого_объекта. Сортируются полученные, после применения функции, значения. Необязательный аргумент.

reverse – в случае если этот именной аргумент равен True, сортировка будет произведена в обратном порядке. Необязательный аргумент.

sorted()

```
>>>a = [3, 2, 1]
>>>sorted(a)
[1, 2, 3]
>>> s = ('K', 'e', 'r', 'E', 'k', 'a', 'A')
>>>sorted(s, reverse=True)
['r', 'k', 'e', 'a', 'K', 'E', 'A']
>>> lst = ["Кот", "Кружка", "Машина", "Клад", "Ор"]
>>> sorted(lst, key=len) # Отсортируем их по длине строки
['Ор', 'Кот', 'Клад', 'Кружка', 'Машина']
```

zip()

zip() - возвращает итератор кортежей состоящих из элементов аргумента, позволяет «склеить» два итерируемых объекта.

```
>>>letters = 'abcd'

>>>numbers = (10, 20, 30)

>>>zipped = zip(letters, numbers)

>>>zipped_list = list(zipped)

>>>zipped_list

[('a',10), ('b', 20), ('c', 30)]
```

с помощью zip можно создать словарь из двух списков

enumerated()

enumerated() - нумерует элементы итерируемого объекта при этом начало нумерации можно задавать произвольно. Создает объект, который генерирует кортежи, состоящие из двух элементов - индекса элемента и самого элемента.

enumerate(iterable, start)

iterable – объект поддерживающий итерирование. Обязательный аргумент.

start – целое число (int) возвращаемое в кортеже с первым элементом итерируемого объекта. По умолчанию аргумент равен нулю. Необязательный аргумент

enumerated()

```
>>> a = [10, 20, 30, 40]
>>> for i in enumerate(a):
...     print(i)
...
(0, 10)
(1, 20)
(2, 30)
(3, 40)
```

reversed()

reversed - разворачивает итерируемый объект

Функция reversed() возвращает обратный итератор, то есть возвращает итератор, который перебирает элементы оригинала в обратном порядке. Функция reversed() не создает копию и не изменяет оригинал последовательности.

```
>>> x = [16, 25, 13, 12, 14, 10]

>>> list(reversed(x))

[10, 14, 12, 13, 25, 16]


>>>x = 'category'

>>>for i in reversed(x):

    print(i, end='')

yrogetac
```

Практика 1

1) Вывести на экран таблицу символов в виде: