**Python**

|  |  |  |
| --- | --- | --- |
| **Модуль** | **Команда** | **Пояснение** |
|  | print(“Hello World”) | вывод сообщения в консоль |
|  | print(‘text’, **end = ‘|’**) | ставит символ в конец строки. По умолчанию это символ переноса строки |
|  | Print(“{:>8d}“.format(some\_int)) | Выравнивает число по правому краю поля шириной 8 символов: \*\*\*\*\*123 |
|  | print(‘text’, ‘text2’, **sep=’ ‘**) | sep – символ, которые разделяет |
|  | print('-' **\* 10**) | вывести ---------- |
|  | word = input(“Enter some text: “) | ввод |
|  | import time | импорт модуля. |
|  | from time import sleep | из модуля импортируем конкретную функцию |
|  | sleep(2) | **приостановка** программы на 2 сек |
|  | If \_\_name\_\_==“\_\_main\_\_“: | True если прог-а запускается  False если импортируется |
|  | dir(random) | показывает все функции модуля |
|  | help(random.randint) | показывает описание данной функции |
|  | def foo()  “““reference“““  help(foo) | >>> reference |
|  | Python3 test.py 1 2 3  print(sys.argv) | [„test.py“, „1“, „2“, „3“] |
|  | x = 10 if y > 3 else x = 20 | тернарный оператор |
|  | type(obj) | возвращает **тип** объекта |
|  | def func(arg): | объявление **функции** |
|  | def func(name**:str**) **-> set:**  //аргумент str, возвращение set | добавление **аннотации**, чтобы дать *информацию* об использовании типов |
|  | def func(arg, arg2**:str=’name’**) | функция принимает два аргумента. Однако, если второй аргумент пропущен, ему присваивается значение ‘name’ |
|  | def func(…):  “”” subscribe**.**””” | описание того, что делает функция. Точка в конце |
|  | def func(par1=arg1, par2=arg2) | присваивание аргументов **по ключу** |
|  | bool(condition) | возвращает True или False от условия |
|  | bool(obj) | возвращает True, если объект не пуст |
|  | Try:  a = 1/0  except **ZeroDivisionError**:  print(«divine to 0») | Исключения. Существуют другие ошибки, например **KeyboardInterrupt** |
|  | input = open(‘test.txt’) | открыть существующий файл |
|  | for every in input | считывает строки из файла |
|  | cpntents = input.read() | прочитать файл целиком |
|  | output = open(‘test.txt’, ‘a’) | создает файл. ‘a’ - append |
|  | output = open(‘test.txt’, ‘w’) | создать файл для записи |
|  | output = open(‘test.txt’, ‘x’) | создать файл для записи. Если файл *уже существует,* то не сработает |
|  | print(‘some text’, ‘another text’, file = output) | записать текст в файл |
|  | output.close() |  |
|  | with open(‘test.txt’) as input: | работа с файлом внутри **блока.** Close не нужен |
|  | If file.endswith(“.wav“) | Расширение файла |
|  | dir(obj) | возвращает список от объекта |
| elif == else if for i in [1,2,3]: for i in range(5) | | |
|  | range(5) | возвращает список [0,1,2,3,4] |
|  | range(2, 6) | возвращает список [2,3,4,5] |
|  | range(start, stop, step) | возвращает список |
|  | range(7,0,-2) | возвращает список [7,5,3,1] |
|  | ls = [None] \* 5 | [None, None, None, None, None] |
|  | len(mas) | возвращает количество элементов списка |
|  | mas[start:stop:step] | возвращает выбранные только выбранные элементы списка. Аргументы могут быть пропущены. Исходный список не меняется |
|  | mas[-1-i] == mas[len(mas)-i] | обращение к элементу списка с конца |
|  | mas1 = mas[middle:] | получить первую половину списка |
|  | mas1 = mas[:middle] | получить вторую половину списка |
|  | mas.append(n) | добавить элемент в конец списка |
|  | mas.remove(a) | удаляет элемент a из списка |
|  | mas.pop(i) | возвращает и удаляет элемент по индексу. Если аргумента нет, выбирается последний индекс |
|  | mas.extend([1,2,3]) | добавляет к списку новые элементы |
|  | mas.insert(i, a) | вставляет элемент a на место индекса i |
|  | mas = list(string) | переделывает строку в массив |
|  | str = ‘ ‘.join(mas) | переделывает список в строку. ‘ ‘ - **разделитель** |
|  | mas = str.split(‘ ‘) | разделяет строку на элементы массива.  ‘ ‘ - **разделитель** |
|  | mystr.startswith('hello') -> bool | проверяет начало строки |
|  | mas1 = mas | копирует **ссылку** на список. Изменение одного списка приводит к изменению др-го |
|  | mas1 = copy(mas) | копирует **элементы** списка в другой список |
|  | mas = sorted(mas) | сортирует список |
|  | map = {“name”: “vova”, “age”: 3} | создание словаря |
|  | if “name” in map: | проверка на наличие ключа в списке |
|  | map.setdefault(“num”, 0) | если такого ключа нет, присваивает ему 0 |
|  | for k,v in found.items() | возвращает в цикле пару: ключ-значение (в том же порядке, как при добавлении) |
|  | max(my\_dict, key=my\_dict.get) | получить ключ макс. значения |
|  | Dict1.update(dict2) | Объединение словарей. |
|  | Dict.keys() | Получить список ключей |
|  | myset = {a, b, c, d} // == myset = set{“abcd”} | создание множества |
|  | if letter in myset | проверка на вхождение в множество |
|  | myset = set(“something”) | переводит строку в множество |
|  | myset.add(new\_elem) | добавление нового |
|  | newset = set1.union(set2) | объединяет два множества |
|  | newset = set1.difference(set2) | разность двух множеств |
|  | newset = set1.intersection(set2) | пересечение двух множеств |
|  | str = sorted(list(myset)) | превращает множество в отсорт. список |
|  | cort = (a, b, c, d) | создание кортежа (**неизменяемого** списка) |
|  | for i, element in **enumerate**(array) | Цикл по индексу и элементу |
| datetime | datetime.today().minute | возвращает текущую минуту |
|  | datetime.date.today() | возвращает текущую дату |
|  | datetime.date.today().day/month/year | возвращает текущий день/месяц/год |
|  | datetime.date.isoformat(datetime.date.today()) | дата в формате 2015-10-12 |
| time | time.strftime(“%I:%M %p %A”) | выводит время “05:16 pm Friday” |
| random | randint(a, b) | выводит случайное число в диапазоне a b |
|  | num = complex(re, im) | преобразование к комплекс. числу |
|  | k,m = divmod(a,b) | возвращает частное и остаток от делен |
|  | round(a, k) | округление до k знаков после запятой |
|  | new\_list = sorted(list, key, reverse = True) | отсортировать список |
|  | x // y | получение целой части от деления |
|  | x % y | остаток от деления |
|  | x \*\* y | возведение в степень |
| serial | ser = serial.Serial("COM3", 9600) | подключение COM порта |
|  | if ser.isOpen(): | если порт подключен |
|  | ser.write(b’1’) | вывод информации в порт |
|  | if isinstance(smth, **int**): | если smth имеет тип **int**, то |
| time | time.sleep(sec) | приостановить программу |
| os | system(‘sh bash.sh’) | Команда для command line |
|  | def \_method()  self.\_method() | Приватный метод класса |
|  | \_variable | Приватный аттрибут класса |
|  | def \_func() | Функция недоступна за пределом данного файла |
|  | a = [1, 2, 3]  b = [a, 4, 5] # *[[1, 2, 3], 4, 5]*  b = [a\*, 4, 5] # *[1, 2, 3, 4, 5]* | распаковка |
|  | def \_\_private\_method(self): | приватный метод класса |
| В Python при передаче в функцию: **list, dict, ndarray, set** фактически передается ссылка на объект => при изменении объекта в функции, он изменяется за пределами (после) функции. Чтобы это избежать, можно создать копию объекта, например, **cp\_ls = list(ls)** | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Абстрактный класс –** это класс-родитель, который не предполагает создание его экземпляров | | |
| **\_\_call\_\_** - метод класса, который вызывается при вызове экземпляра | | |
| **https://pyformat.info/** | | |