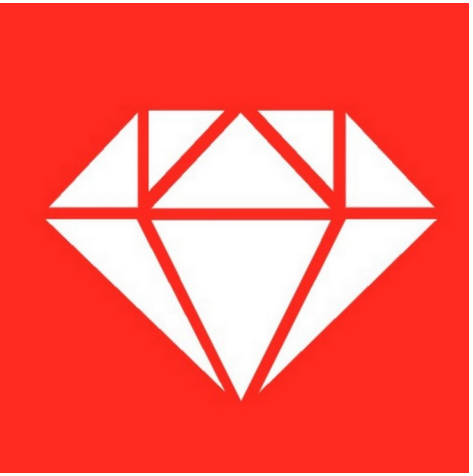
Архитектура уровня динамически типизированного языка



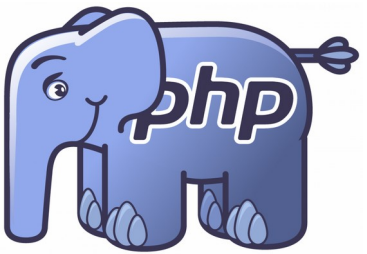
Основные термины и определения

Динами́ческая типиза́ция — приём, используемый в языках программирования и языках спецификации, при котором переменная связывается с типом в момент присваивания значения, а не в момент объявления переменной. Таким образом, в различных участках программы одна и та же переменная может принимать значения разных типов.

Свойства языков с динамической типизацией

Примеры языков с динамической типизацией:

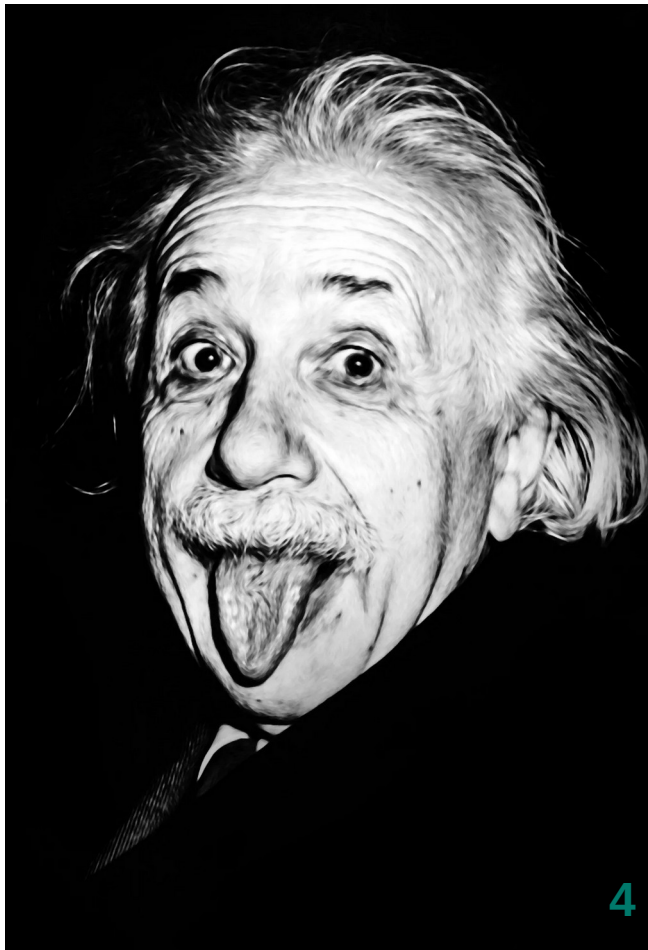
- **Smalltalk**
- **Python**
- **Objective-C**
- **Ruby**
- **PHP**
- **Perl**
- **JavaScript**
- **Лисп**



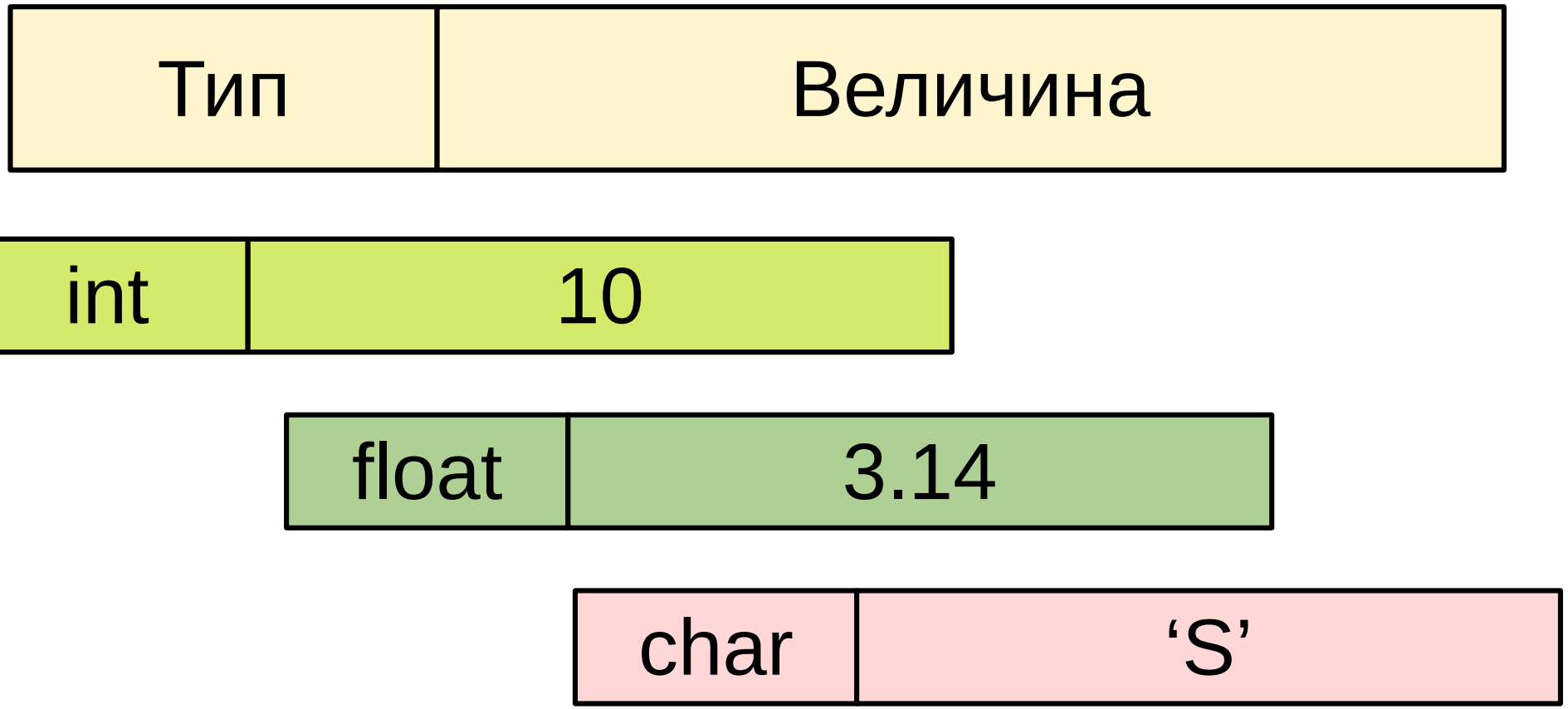
Динамическая типизация упрощает написание программ для работы с меняющимся окружением, при работе с данными переменных типов; при этом отсутствие информации о типе на этапе компиляции повышает вероятность ошибок в исполняемых модулях.

Позиции языков с динамической типизацией

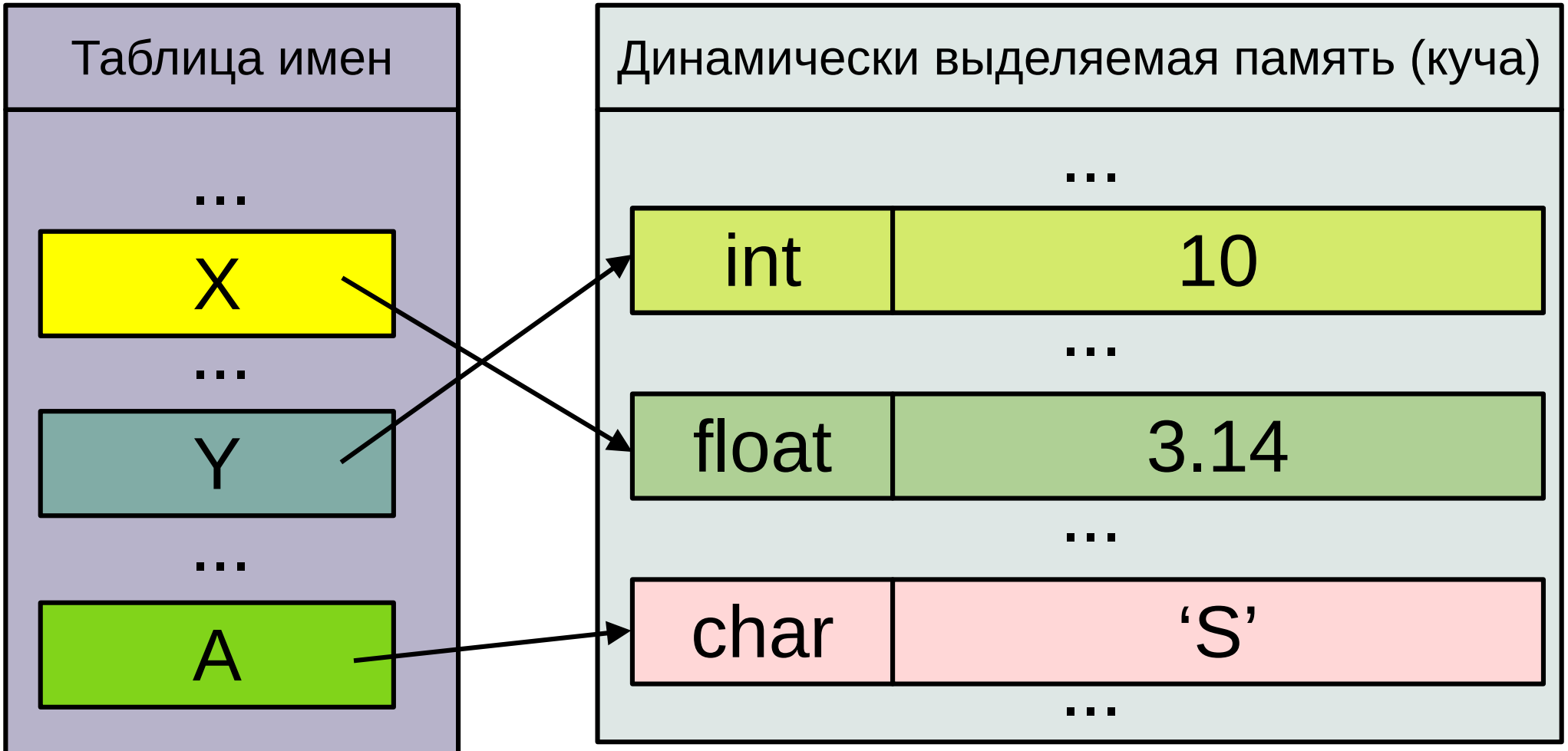
Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	↑	Python	11.67%	+1.20%
3	**2**	↓	**Java**	**11.12%**	**-2.37%**
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%
8	14	⇈	Assembly language	2.42%	+1.12%
9	8	↓	PHP	1.85%	-0.64%
10	10		SQL	1.80%	+0.04%
11	22	⇈	Classic Visual Basic	1.52%	+0.77%
12	17	⇈	Groovy	1.46%	+0.48%
13	15	↑	Ruby	1.27%	+0.03%
14	11	↓	Go	1.13%	-0.33%
15	12	↓	Swift	1.07%	-0.31%



Организация значения при динамической типизации



Обращение к величинам через указатели



Прямое использование динамической типизации

```
if __name__ == '__main__':
    if len(sys.argv) == 3:
        inputFileName  = sys.argv[1]
        outputFileName = sys.argv[2]
    elif len(sys.argv) == 2:
        inputFileName  = sys.argv[1]
        outputFileName = "output.txt"
    elif len(sys.argv) == 1:
        print("Incorrect command line! You must write: python main <inputFileName> [<outputFileName>]")
        exit()

    ifile = open(inputFileName)
    str = ifile.read()
    ifile.close()

    # Формирование массива строк, содержащего чистые данные в виде массива строк символов.
    strArray = str.replace("\n", " ").split(" ")

    print('==> Start')
    cont = []
    figNum = container.Read(cont, strArray)

    # Тестовый вывод содержимого контейнера
    print("Container as list of lists {}\n".format(cont))

    container.Print(cont)

    ofile = open(outputFileName, 'w')
    container.Write(cont, ofile)
    ofile.close()
    print('==> Finish')
```

Образное представление прямоугольника

```
['rectangle', 3, 4]
```

```
def Read(rect, strArray, i):
    # должно быть как минимум два непрочитанных значения в массиве
    if i >= len(strArray) - 1:
        return 0
    rect.append("rectangle")
    rect.append(int(strArray[i]))
    rect.append(int(strArray[i+1]))
    i += 2
    #print("Rectangle: x = ", rect[1], " y = ", rect[2])
    return i

def Print(rect):
    print("Rectangle: x = ", rect[1], " y = ", rect[2], ", Perimeter = ", Perimeter(rect))
    pass

def Write(rect, ostream):
    ostream.write("Rectangle: x = {}  y = {}, Perimeter = {}".format(rect[1], rect[2], Perimeter(rect)))
    pass

def Perimeter(rect):
    return 2.0 * (rect[1] + rect[2])
    pass
```

Образное представление треугольника

```
['triangle', 2, 2, 1]
```

```
def Read(trian, strArray, i):
    # должно быть как минимум три непрочитанных значения в массиве
    if i >= len(strArray) - 2:
        return 0
    trian.append("triangle")
    trian.append(int(strArray[i]))
    trian.append(int(strArray[i+1]))
    trian.append(int(strArray[i+2]))
    i += 3
    #print("Triangle: a = ", trian[1], " b = ", trian[2], "c = ", trian[3])
    return i

def Print(trian):
    print("Triangle: a = ", trian[1], " b = ", trian[2], "c = ", trian[3], ", Perimeter = ", Perimeter(trian))
    pass

def Write(trian, ostream):
    ostream.write("Triangle: a = {}  b = {}  c = {}, Perimeter = {}".format \
                  (trian[1], trian[2], trian[3], Perimeter(trian)))
    pass

def Perimeter(trian):
    return float(trian[1] + trian[2] + trian[3])
    pass
```

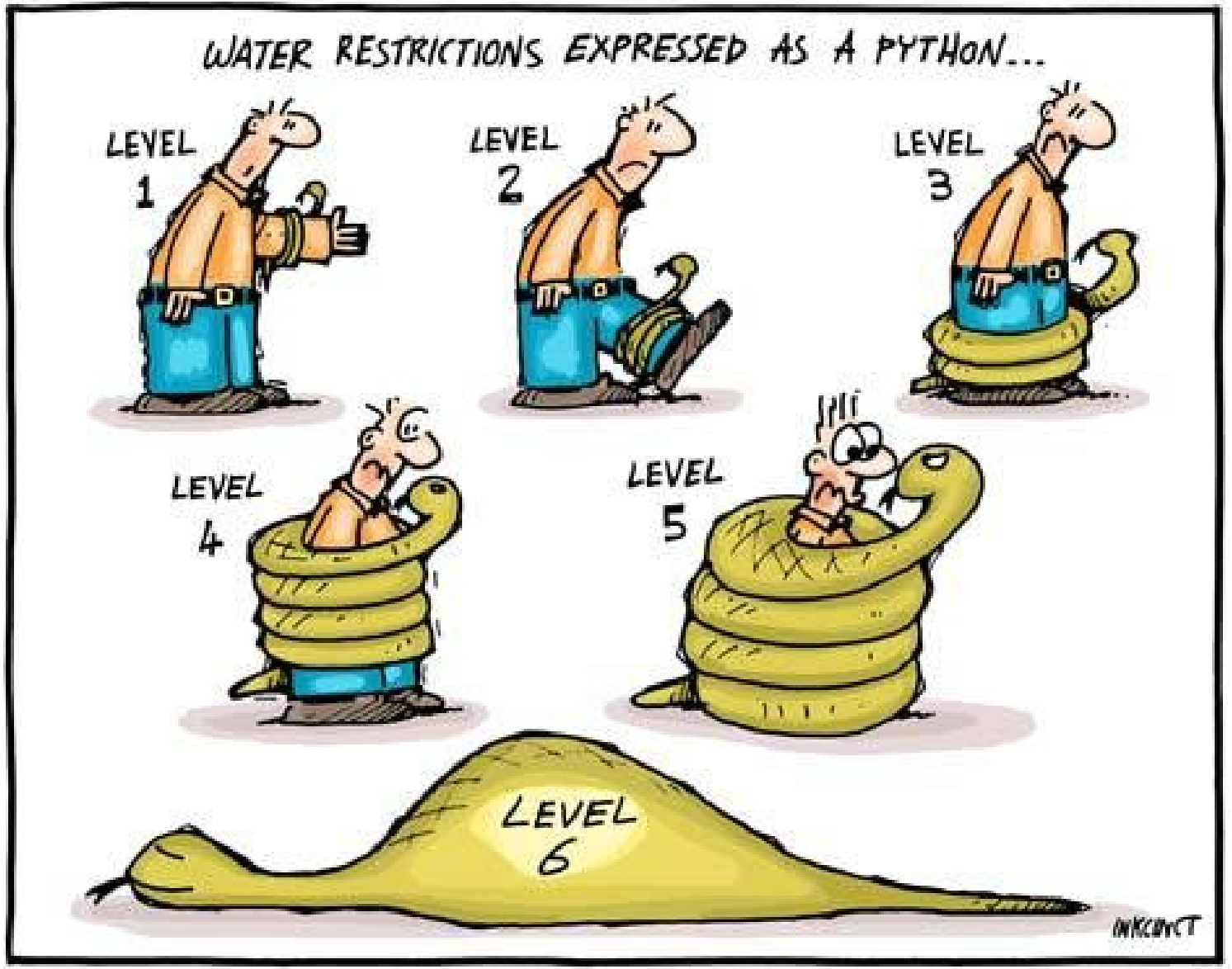
Загрузка данных в контейер

```
def Read(cont, strArray):
    arrayLen = len(strArray)
    i = 0   # Индекс, задающий текущую строку в массиве
    figNum = 0  # Счетчк фигур
    while i < arrayLen:
        str = strArray[i]
        key = int(str)   # преобразование в целое
        #print("key = ", key)    # тестовый вывод ключа фигуры
        if key == 1: # признак прямоугольника
            i += 1
            rect = []   # прямоугольник - это список с признаком и значениями
            i = rectangle.Read(rect, strArray, i)   # Заполнение фигуры значением и получение следующей позиции
            cont.append(rect)      # прямоугольник поступает в контейнер
        elif key == 2: # признак треугольника
            i += 1
            trian = []   # прямоугольник - это список с признаком и значениями
            i = triangle.Read(trian, strArray, i)   # Заполнение фигуры значением и получение следующей позиции
            cont.append(trian)      # треугольник поступает в контейнер
        else:
            # что-то пошло не так. Должен быть известный признак
            # Возврат количества прочитанных фигур
            return figNum
        # Количество фигур увеличивается на 1
        if i == 0:
            return figNum
        figNum += 1
    return figNum
```

Контейер как список списков (образных фигур)

```
def Print(cont):
    print("Container stores", len(cont), "shapes:")
    for shape in cont:
        if shape[0] == "rectangle":
            rectangle.Print(shape)
        elif shape[0] == "triangle":
            triangle.Print(shape)
        else:
            print("Incorrect figure ", shape)
    print("Summa of Perimeters  = ", Perimeter(cont))
    pass

def Write(cont, ostream):
    ostream.write("Container stores {} shapes:\n".format(len(cont)))
    for shape in cont:
        if shape[0] == "rectangle":
            rectangle.Write(shape, ostream)
        elif shape[0] == "triangle":
            triangle.Write(shape, ostream)
        else:
            ostream.write("Incorrect figure ")
            ostream.write(shape)
        ostream.write("\n")
    ostream.write("Summa of Perimeters  = {}\n".format(Perimeter(cont)))
    pass

def Perimeter(cont):
    perim = 0.0
    for shape in cont:
        if shape[0] == "rectangle":
            perim += rectangle.Perimeter(shape)
        elif shape[0] == "triangle":
            perim += triangle.Perimeter(shape)
        else:
            perim += 0.0
    return perim
```

```
==> Start
Container as list of lists [['rectangle', 3, 4], ['trian
gle', 1, 1, 1], ['rectangle', 30, 40], ['triangle', 2, 2
, 1], ['rectangle', 13, 14], ['triangle', 10, 10, 10], [
'rectangle', 330, 49], ['triangle', 3, 4, 5]]

Container stores 8 shapes:
Rectangle: x =  3  y =  4 , Perimeter =  14.0
Triangle: a =  1  b =  1 c =  1 , Perimeter =  3.0
Rectangle: x =  30  y =  40 , Perimeter =  140.0
Triangle: a =  2  b =  2 c =  1 , Perimeter =  5.0
Rectangle: x =  13  y =  14 , Perimeter =  54.0
Triangle: a =  10  b =  10 c =  10 , Perimeter =  30.0
Rectangle: x =  330  y =  49 , Perimeter =  758.0
Triangle: a =  3  b =  4 c =  5 , Perimeter =  12.0
Summa of Perimeters  =  1016.0
==> Finish
```


WATER RESTRICTIONS EXPRESSED AS A PYTHON...
LEVEL 1
LEVEL 2
LEVEL 3
LEVEL 4
LEVEL 5
LEVEL 6

Динамическая типизации и ОО подход

```
if __name__ == '__main__':
    if len(sys.argv) == 3:
        inputFileName  = sys.argv[1]
        outputFileName = sys.argv[2]
    elif len(sys.argv) == 2:
        inputFileName  = sys.argv[1]
        outputFileName = "output.txt"
    elif len(sys.argv) == 1:
        print("Incorrect command line! You must write: python main <inputFileName> [<outputFileName>]")
        exit()

    # Чтение исходного файла, содержащего данные, разделенные пробелами и переводами строки
    ifile = open(inputFileName)
    str = ifile.read()
    ifile.close()

    # Формирование массива строк, содержащего чистые данные в виде массива строк символов.
    strArray = str.replace("\n", " ").split(" ")

    print('==> Start')
    container = Container()
    figNum = ReadStrArray(container, strArray)
    container.Print()

    ofile = open(outputFileName, 'w')
    container.Write(ofile)
    ofile.close()

    print('==> Finish')
```

Функция формирующая из строки фигуры

```
def ReadStrArray(container, strArray):
    arrayLen = len(strArray)
    i = 0   # Индекс, задающий текущую строку в массиве
    figNum = 0
    while i < arrayLen:
        str = strArray[i]
        key = int(str)   # преобразование в целое
        #print("key = ", key)
        if key == 1: # признак прямоугольника
            i += 1
            shape = Rectangle()
            i = shape.ReadStrArray(strArray, i) # чтение прямоугольника с возвратом позиции за ним
        elif key == 2: # признак треугольника
            i += 1
            shape = Triangle()
            i = shape.ReadStrArray(strArray, i) # чтение треугольника с возвратом позиции за ним
        else:
            # что-то пошло не так. Должен быть известный признак
            # Возврат количества прочитанных фигур
            return figNum
        # Количество пробелами фигур увеличивается на 1
        if i == 0:
            return figNum
        figNum += 1
        container.store.append(shape)
    return figNum
```

Базовый класс, обеспечивающий полиморфизм

```
class Shape:
    def ReadStrArray(self, strArray, i):
        pass

    def Print(self):
        pass

    def Write(self, ostream):
        pass

    def Perimeter(self):
        pass
```

Производный класс, задающий прямоугольник

```
class Rectangle(Shape):
    def __init__(self):
        self.x = 0
        self.y = 0

    def ReadStrArray(self, strArray, i):
        # должно быт как минимум два непрочитанных значения в массиве
        if i >= len(strArray) - 1:
            return 0
        self.x = int(strArray[i])
        self.y = int(strArray[i+1])
        i += 2
        #print("Rectangle: x = ", self.x, " y = ", self.y)
        return i

    def Print(self):
        print("Rectangle: x = ", self.x, " y = ", self.y, ", Perimeter = ", self.Perimeter())
        pass

    def Write(self, ostream):
        ostream.write("Rectangle: x = {}  y = {}, Perimeter = {}".format(self.x, self.y, self.Perimeter()))
        pass

    def Perimeter(self):
        return 2.0 * (self.x + self.y)
        pass
```

Производный класс, задающий треугольник

```
class Triangle(Shape):
    def __init__(self):
        self.a = 0
        self.b = 0
        self.c = 0

    def ReadStrArray(self, strArray, i):
        # должно быт как минимум три непрочитанных значения в массиве
        if i >= len(strArray) - 2:
            return 0
        self.a = int(strArray[i])
        self.b = int(strArray[i+1])
        self.c = int(strArray[i+2])
        i += 3
        #print("Triangle: a = ", self.a, " b = ", self.b, "c = ", self.c)
        return i

    def Print(self):
        print("Triangle: a = ", self.a, " b = ", self.b, "c = ", self.c, ", Perimeter = ", self.Perimeter())
        pass

    def Write(self, ostream):
        ostream.write("Triangle: a = {}  b = {}  c = {}, Perimeter = {}".format \
                    (self.a, self.b, self.c, self.Perimeter()))
        pass

    def Perimeter(self):
        return float(self.a + self.b + self.c)
        pass
```

Класс, задающий контейнер контейнер

```
class Container:
    def __init__(self):
        self.store = []

    def Print(self):
        print("Container is store", len(self.store), "shapes:")
        for shape in self.store:
            shape.Print()
        print("Summa of Perimeters  = ", self.Perimeter())
        pass

    def Write(self, ostream):
        ostream.write("Container is store {} shapes:\n".format(len(self.store)))
        for shape in self.store:
            shape.Write(ostream)
            ostream.write("\n")
        ostream.write("Summa of Perimeters  = {}\n".format(self.Perimeter()))
        pass

    def Perimeter(self):
        perim = 0.0
        for shape in self.store:
            perim += shape.Perimeter()
        return perim
```

Базовый класс, с «забытым» методом вычисления периметра?

```
class Shape:
    def ReadStrArray(self, strArray, i):
        pass

    def Print(self):
        pass

    def Write(self, ostream):
        pass
```

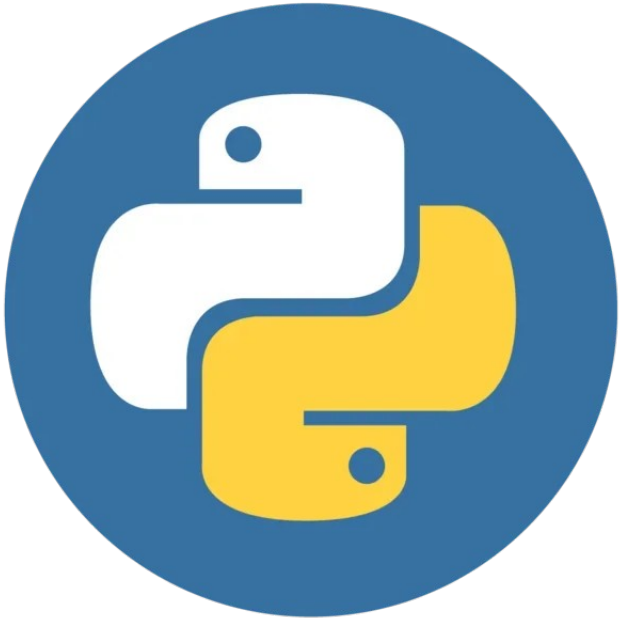
Базовый класс, без “виртуальных” методов?

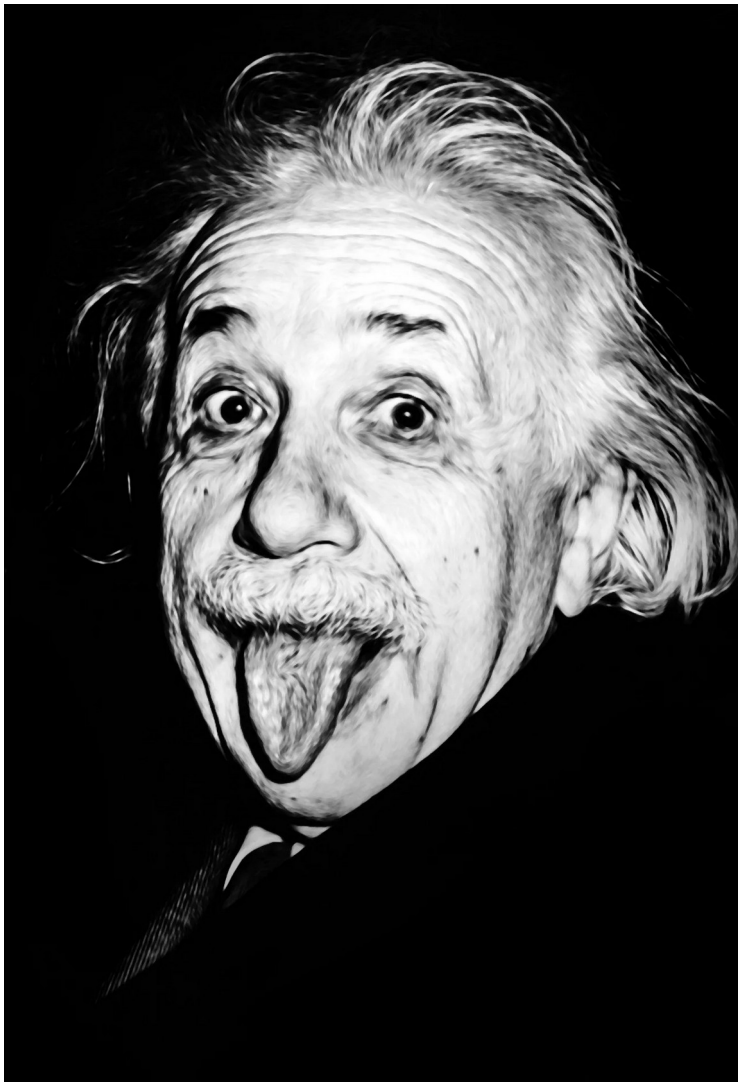
```
class Shape:
    pass
```

Методы без базового класса?

```
class Rectangle:
    def __init__(self):
        self.x = 0
        self.y = 0

    def ReadStrArray(self, strArray, i):
        # должно быт как минимум два непрочитанных значения в массиве
        if i >= len(strArray) - 1:
            return 0
        self.x = int(strArray[i])
        self.y = int(strArray[i+1])
        i += 2
        #print("Rectangle: x = ", self.x, " y = ", self.y)
        return i

    def Print(self):
        print("Rectangle: x = ", self.x, " y = ", self.y, ", Perimeter = ", self.Perimeter())
        pass

    def Write(self, ostream):
        ostream.write("Rectangle: x = {}  y = {}, Perimeter = {}".format(self.x, self.y, self.Perimeter()))
        pass

    def Perimeter(self):
        return 2.0 * (self.x + self.y)
        pass
```

Бестиповое поведение?





Использование явной проверки типа для селекции данных

```
def OnlyRectanglePerimeter(self):
    perim = 0.0
    for shape in self.store:
        #print(type(shape))
        if isinstance(shape, rectangle.Rectangle):
            perim += shape.Perimeter()
    return perim

def OnlyTrianglePerimeter(self):
    perim = 0.0
    for shape in self.store:
        #print(type(shape))
        if isinstance(shape, triangle.Triangle):
            perim += shape.Perimeter()
    return perim
```

```
print("Only triangle perimeter = ", container.OnlyTrianglePerimeter());
print("Only rectangle perimeter = ", container.OnlyRectanglePerimeter());
```

Список источников

Inside The Python Virtual Machine
https://leanpub.com/insidethepythonvirtualmachine/read
Внутри виртуальной машины Python. Части 1, 2 (перевод текста выше)
https://habr.com/ru/post/501338/
https://habr.com/ru/post/501920/

Лекция 14: Устройство интерпретатора языка Python
https://intuit.ru/studies/courses/49/49/lecture/27084

Youtube

Злата Обуховская - Цикл "Что внутри у Питона"
https://www.youtube.com/playlist?list=PLv_zOGKKxVpi6BSAuySAtX5KyCa50PSCz