ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON

Докладчик: Шеина Татьяна Юрьевна

ПРОГРАММИРОВАНИЕ – ОДИН ИЗ РАЗДЕЛОВ ПРЕДМЕТА ИНФОРМАТИКА

Паскаль – основой язык для обучения школьников программированию в настоящее время

Плюсы языка Паскаль:

- четкая структура программы,
- строгое описание типов переменных,
- строгое соблюдение принципов процедурного программирования,
- используется в качестве одного из языков при сдаче ОГЭ и ЕГЭ по информатике,
- привычен для большинства учителей информатики,
- многие учебники ориентированы на изучение этого языка.

ВИДЫ МОТИВАЦИИ К ИЗУЧЕНИЮ ПРОГРАММИРОВАНИЯ

- использование на уроках различных мультимедийных средств,
- метод проектов,
- переход к объектно-ориентированной парадигме программирования,
- использование роботов и робототехники,
- деловые игры,
- программирование компьютерных игр.

ПРЕИМУЩЕСТВА ЯЗЫКА PYTHON

- Язык достаточно прост в изучении;
- Не требует описания типов и строгой структуры программы;
- Содержит встроенные высокоуровневые структуры (списки, словари, кортежи);
- Включает большое количество дополнительных модулей (библиотек), в том числе модуль Random, используемый для генерации различных случайных значений;
- Используется огромным количеством программистов в самых различных сферах программной разработки;
- Позволяет совмещать процедурный, объектно-ориентированный и функциональный подход к написанию кода.
- Является кроссплатформенным языком, то есть программы на этом языке работают на компьютерах с любой ОС
- Интерпретатор языка является бесплатным
- Можно работать в интерактивном и сценарном режиме, так как является интерпретатором

ОБЩЕЕ ЗНАКОМСТВО С ЯЗЫКОМ PYTHON

Python – мощный и простой в использовании язык программирования, разработанный голландцем Гвидо ван Россумом в 1991 году

Области использования языка:

- Web-программирование (среды для разработки Web-приложений)
- Графика (визуализация научных данных)
- Компьютерные игры (скриптовая поддержка движков)
- В качестве скриптового языка при создании прикладных программ (Gimp, Blender)

ВЫВОД ИНФОРМАЦИИ И ОСНОВНЫЕ АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

print(выражение) – вывод информации

Пример:
print(“Over Game”) или print(‘Over Game’)

Арифметические операции:
+ - сложение
- вычитание
* - умножение
/ - обычное деление
// - целочисленное деление
% - остаток от деления

Пример:
```
print(6*8+10)
        58
print(“55/10=”,55/10)
        55/10=5.5
```

ПОНЯТИЕ ПЕРЕМЕННОЙ. ОПЕРАЦИЯ ПРИСВАИВАНИЯ. ВВОД ИНФОРМАЦИИ

Переменные позволяют хранить данные под именами и через эти имена обращаться к данным. Чтобы занести информацию в переменную, используется оператор присваивания, который имеет следующий вид:

имя_переменной=выражение

Пример

```
name=“Вася”
print(name)
```

Если информацию в переменную необходимо ввести с клавиатуры, то используется функция input:

имя_переменной=input(“Сообщение”)

Пример

```
name=input(“Как тебя зовут?”)
print(“Очень приятно,”, name)
```

ПРЕОБРАЗОВАНИЕ ТИПОВ. ГЕНЕРАТОР СЛУЧАЙНЫХ ЧИСЕЛ

Функции конвертации типов:
int(x) – преобразование к целому типу
float(x) – преобразование к вещественному типу
str(x) – преобразование к строковому типу

Модули – это файлы, содержащие уже готовый код, пригодный для использования в других программах.
import random – подключение библиотеки random
randrange(x) - возвращает случайное целое число в диапазоне от 0 до x-1.
randint(x,y)- возвращает целое число в диапазоне от x до y.

Пример

```
import random
die1=random.randint(1,6)
die2=random.randrange(6)+1
total=die1+die2
print(“Ваш бросок: ”,die1,” и ”,die2,”. В сумме=”, total)
```

Условный оператор

Условный оператор позволяет выбрать для исполнения тот или иной фрагмент кода в зависимости от истинности определенного условия:

```
if условие:
    Блок команд 1
else:
    Блок команд 2
```

В языке Python нет понятия «операторные скобки». Вместо этого используются отступы. То есть строки с одинаковым отступом образуют блок. Для создания условия используются следующие операции отношения:

`==` - равно
`!=` - не равно
`>` - больше
`<` - меньше
`>=` - больше или равно
`<=` - меньше или равно

СЛОЖНЫЙ ОПЕРАТОР ВЕТВЛЕНИЯ

```
if условие 1:
        блок1
elif условие2:
        блок2
elif условие 3:
        блок3
...
else:
        блокN+1
```

Для создания сложных условий используются логические операции and и or.

ПРИМЕР ОПЕРАТОРА ВЕТВЛЕНИЯ

```
month=input("Введите номер месяца")
month=int(month);
if month==12 or month==1 or month==2:
        print("Зима")
elif month==3 or month==4 or month==5:
        print("Весна")
elif month==6 or month==7 or month==8:
        print("Лето")
elif month==9 or month==10 or month==11:
        print("Осень")
else:
        print("неверный ввод")
```

ЦИКЛ WHILE

Суть цикла заключается в повторении определенного набора действий до тех пор, пока истинно некоторое условие:

```
while условие:
    блок команд
```

Пример

```
print("Ваш герой окружен толпой троллей.")
print("Он достает меч из ножен, готовясь к последней битве в жизни")
health=10
trolls=0
damage=3
while health>0:
    trolls=trolls+1
    health=health-damage
    print("Ваш герой истребляет тролля и получает повреждение на “ ,
    damage," очков")
print("Ваш герой убил ",trolls," троллей и пал на поле боя")
```

ЦИКЛ FOR

В фундаменте цикла for лежит последовательность – упорядоченное множество объектов. Этим цикл for в языке Python очень сильно отличается от многих других языков программирования. Цикл for перебирает все элементы последовательности по порядку и для каждого из них исполняет фрагмент кода, заключенный в теле цикла. После достижения конца последовательности цикл завершается:

```
for имя_переменной in последовательность:
    блок команд
```

Пример (вывод чисел от 0 до 9)

```
for i in range(10):
    print(i,end=" ")
```

Пример

```
letter=input("Введите слово")
for i in letter:
    print(i)
```

СТРОКИ. СРЕЗЫ СТРОК

Механизм индексирования позволяет выбирать не обязательно только один элемент последовательности. Можно создавать копии непрерывных последовательностей. Такие копии называют срезами. Чтобы задать границы среза, надо записать номера начальной и конечной позиций в квадратных скобках через двоеточие. Конечная позиция в срез не входит. Необходимо также учитывать, что нумерация во всех последовательностях начинается с 0.

Пример

```
word="слово”
a=word[1:4]
print(a)
```

Получим «лов».

Разрешается опускать начальную позицию среза. Вместо нее будет рассматриваться самый первый элемент последовательности. Например, word[:4] – это то же самое, что word[0:4]. Разрешается опускать конечную позицию последовательности. В этом случае срез будет заканчиваться последним символов – word[1:]. Чтобы получить полную копию последовательности, надо записать word[:].

КОРТЕЖИ

Кортеж – это еще один тип последовательности. Кортежи способны содержать элементы любой природы. Чтобы создать кортеж – достаточно заключить в круглые скобки последовательность значений, которые разделены запятыми. Даже просто пара скобок рассматривается как пустой кортеж.

Пример

```
inventory=()
inventory=(“меч”,”кольчуга”,”щит”)
```

print(inventory) – вывод кортежа на экран

Для перебора всех элементов кортежа используется цикл for:

```
for item in inventory:
    print(item)
```

Чтобы узнать количество элементов в кортеже, используется функция len():

```
print(len(inventory))
```

Чтобы установить, является ли элемент членом кортежа, используется оператор in:

```
if “целебное снадобье” in inventory:
    print(“Вы ранены, но будете жить”)
else:
   print(“К сожалению, вы не выживете”)
```

Так же к кортежам можно применять срезы:

`print(inventory[0:2])` (меч, кольчуга)

Для объединения двух кортежей используется операция сцепления - `+`.

Для выбора из кортежа случайного значения используется функция `random.choice(имя кортежа):`

```
a=random.choice(inventory)
```

ПРИМЕР «АНАГРАММЫ»

```
import random #подключение библиотеки для генерации случайных чисел
new='' #перемешанное слово
#список слов в виде кортежа
a=("подъезд","автобус","школа","уроки","ёлка","качели","работа",
"больница","молоко","судно")
slovo1=random.choice(a) #выбор случайного элемента из кортежа
slovo2=slovo1 #создание копии выбранного слова для постепенного удаления из
нее букв
while slovo2!='': #пока слово не станет пустым, будем удалять из него по одной букве
и добавлять к новому слову
    i=random.randrange(len(slovo2)) #генерируем номер буквы от 0 до длины слова
минус 1
    new+=slovo2[i] #добавляем букву со сгенерированным номером к новому слову
new
    slovo2=slovo2[:i]+slovo2[i+1:] #удаляем из копии слова выбранную букву путем
сцепления двух срезов строк
print(new) #выводим на экран перемешанное слово
```

```
print('Угадайте слово')
for c in range (3): #даем три попытки угадать слово
    new1=input() #ввод слова пользователем
    if new1==slovo1: #если слово совпало с исходным
        print("Правильно")
        break
    print("Не угадали")
```

Литература

Майкл Доусон - Программируем на Python

Сайт разработчика

http://python.org