Лабораторная работа №2. Базовые операторы языка. Срезы.

Синтаксис инструкции if

Сначала записывается часть if с условным выражением, далее могут следовать одна или более необязательных частей elif, и, наконец, необязательная часть else. Общая форма записи условной инструкции if выглядит следующим образом:

```
if test1:

    state1

elif test2:

    state2

else:

    state3
```

Простой пример (напечатает 'true', так как 1 - истина):

```
>>> if 1:

...     print('true')

... else:

...     print('false')

...

true
```

Чуть более сложный пример (его результат будет зависеть от того, что ввёл пользователь):

```
a = int(input())
```

```
if a < -5:

    print('Low')

elif -5 <= a <= 5:

    print('Mid')

else:

    print('High')
```

Конструкция с несколькими elif может также служить отличной заменой конструкции switch - case в других языках программирования.

Проверка истинности в Python

- Любое число, не равное 0, или непустой объект - истина.
- Числа, равные 0, пустые объекты и значение None - ложь
- Операции сравнения применяются к структурам данных рекурсивно
- Операции сравнения возвращают True или False
- Логические операторы and и or возвращают истинный или ложный объект-операнд

Логические операторы:

```
X and Y
```

Истина, если оба значения X и Y истинны.

```
X or Y
```

Истина, если хотя бы одно из значений X или Y истинно.

```
not X
```

Истина, если X ложно.

Трехместное выражение if/else

Следующая инструкция:

```
if X:

    A = Y

else:

    A = Z
```

довольно короткая, но, тем не менее, занимает целых 4 строки. Специально для таких случаев и было придумано выражение if/else:

```
A = Y if X else Z
```

В данной инструкции интерпретатор выполнит выражение Y, если X истинно, в противном случае выполнится выражение Z.

```
>>> A = 't' if 'spam' else 'f'

>>> A

't'
```

Цикл while

While - один из самых универсальных циклов в Python, поэтому довольно медленный. Выполняет тело цикла до тех пор, пока условие цикла истинно.

```
>>> i = 5

>>> while i < 15:

...     print(i)

...     i = i + 2
```

```
...

5

7

9

11

13
```

Цикл for

Цикл for уже чуточку сложнее, чуть менее универсальный, но выполняется гораздо быстрее цикла while. Этот цикл проходится по любому итерируемому объекту (например строке или списку), и во время каждого прохода выполняет тело цикла.

```
>>> for i in 'hello world':

...     print(i * 2, end='')

...

hheellllloo  wwoorrlldd
```

Оператор continue

Оператор continue начинает следующий проход цикла, минуя оставшееся тело цикла (for или while)

```
>>> for i in 'hello world':

...     if i == 'o':

...         continue

...     print(i * 2, end='')
```

```
...

hheellll  wwrrlldd
```

Оператор break

Оператор break досрочно прерывает цикл.

```
>>> for i in 'hello world':

...     if i == 'o':

...         break

...     print(i * 2, end='')

...

hheellll
```

Волшебное слово else

Слово else, примененное в цикле for или while, проверяет, был ли произведен выход из цикла инструкцией break, или же "естественным" образом. Блок инструкций внутри else выполнится только в том случае, если выход из цикла произошел без помощи break.

```
>>> for i in 'hello world':

...     if i == 'a':

...         break

... else:

...     print('Буквы a в строке нет')

...
```

```
Буквы a в строке нет
```

Взятие элемента по индексу

Как и в других языках программирования, взятие по индексу:

```
>>> a = [1, 3, 8, 7]

>>> a[0]

1

>>> a[3]

7

>>> a[4]

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

IndexError: list index out of range
```

Как и во многих других языках, нумерация элементов начинается с нуля. При попытке доступа к несуществующему индексу возникает исключение IndexError.

В данном примере переменная a являлась списком, однако взять элемент по индексу можно и у других типов: строк, кортежей.

В Python также поддерживаются отрицательные индексы, при этом нумерация идёт с конца, например:

```
>>> a = [1, 3, 8, 7]

>>> a[-1]

7

>>> a[-4]
```

```
1

>>> a[-5]

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

IndexError: list index out of range
```

Срезы

В Python, кроме индексов, существуют ещё и **срезы**.

item[START:STOP:STEP] - берёт срез от номера START, до STOP (не включая его), с шагом STEP. По умолчанию START = 0, STOP = длине объекта, STEP = 1. Соответственно, какие-нибудь (а возможно, и все) параметры могут быть опущены.

```
>>> a = [1, 3, 8, 7]

>>> a[:]

[1, 3, 8, 7]

>>> a[1:]

[3, 8, 7]

>>> a[:3]

[1, 3, 8]

>>> a[::2]

[1, 8]
```

Также все эти параметры могут быть и отрицательными:

```
>>> a = [1, 3, 8, 7]
```

```
>>> a[::-1]

[7, 8, 3, 1]

>>> a[:-2]

[1, 3]

>>> a[-2::-1]

[8, 3, 1]

>>> a[1:4:-1]

[]
```

В последнем примере получился пустой список, так как START < STOP, а STEP отрицательный. То же самое произойдёт, если диапазон значений окажется за пределами объекта:

```
>>> a = [1, 3, 8, 7]

>>> a[10:20]

[]
```

Также с помощью срезов можно не только извлекать элементы, но и добавлять и удалять элементы (разумеется, только для изменяемых последовательностей).

```
>>> a = [1, 3, 8, 7]

>>> a[1:3] = [0, 0, 0]

>>> a

[1, 0, 0, 0, 7]

>>> del a[:-3]
```

```
>>> a

[0, 0, 7]
```

Модуль math – один из наиважнейших в Python. Этот модуль предоставляет обширный функционал для работы с числами.

math.ceil(X) – округление до ближайшего большего числа.

math.copysign(X, Y) - возвращает число, имеющее модуль такой же, как и у числа X, а знак - как у числа Y.

math.fabs(X) - модуль X.

math.factorial(X) - факториал числа X.

math.floor(X) - округление вниз.

math.fmod(X, Y) - остаток от деления X на Y.

math.frexp(X) - возвращает мантиссу и экспоненту числа.

math.ldexp(X, I) - X * 2^i. Функция, обратная функции math.frexp().

math.fsum(последовательность) - сумма всех членов последовательности. Эквивалент встроенной функции sum(), но math.fsum() более точна для чисел с плавающей точкой.

math.isfinite(X) - является ли X числом.

math.isinf(X) - является ли X бесконечностью.

math.isnan(X) - является ли X NaN (Not a Number - не число).

math.modf(X) - возвращает дробную и целую часть числа X. Оба числа имеют тот же знак, что и X.

math.trunc(X) - усекает значение X до целого.

math.exp(X) - e^x.

math.expm1(X) - e^x - 1. При X → 0 точнее, чем math.exp(X)-1.

math.log(X, [base]) - логарифм X по основанию base. Если base не указан, вычисляется натуральный логарифм.

math.log1p(X) - натуральный логарифм (1 + X). При X → 0 точнее, чем math.log(1+X).

math.log10(X) - логарифм X по основанию 10.

math.log2(X) - логарифм X по основанию 2. Новое в Python 3.3.

math.pow(X, Y) - X^Y.

math.sqrt(X) - квадратный корень из X.

math.acos(X) - арккосинус X. В радианах.

math.asin(X) - арксинус X. В радианах.

math.atan(X) - арктангенс X. В радианах.

math.atan2(Y, X) - арктангенс Y/X. В радианах. С учетом четверти, в которой находится точка (X, Y).

math.cos(X) - косинус X (X указывается в радианах).

math.sin(X) - синус X (X указывается в радианах).

math.tan(X) - тангенс X (X указывается в радианах).

math.hypot(X, Y) - вычисляет гипотенузу треугольника с катетами X и Y (math.sqrt(x * x + y * y)).

math.degrees(X) - конвертирует радианы в градусы.

math.radians(X) - конвертирует градусы в радианы.

math.cosh(X) - вычисляет гиперболический косинус.

math.sinh(X) - вычисляет гиперболический синус.

math.tanh(X) - вычисляет гиперболический тангенс.

math.acosh(X) - вычисляет обратный гиперболический косинус.

math.asinh(X) - вычисляет обратный гиперболический синус.

math.atanh(X) - вычисляет обратный гиперболический тангенс.

math.erf(X) - функция ошибок.

math.erfc(X) - дополнительная функция ошибок (1 - math.erf(X)).

math.gamma(X) - гамма-функция X.

math.lgamma(X) - натуральный логарифм гамма-функции X.

math.pi - pi = 3,1415926...

math.e - e = 2,718281...

Задание

Задание 1. Вычислите сумму ряда.

1.

$$y(x)=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots\right),|x|>1;$$

2.

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots,|x|<\infty$$

3.

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{4!}-\ldots,|x|<\infty$$

4.

$$y(x)=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}...,x>1;$$

5.

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)}=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+...,|x|<=1;$$

6.

$$y(x)=\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+...,|x|>1$$

7.

$$y(x)=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+...,x<-1$$

8.

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-...,|x|<\infty$$

9.

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}-...,|x|<\infty$$

10.

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}-...,|x|<\infty$$

11.

$$y(x)=2\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}=2\left(\frac{x-1}{x+1}+\frac{(x-1)^3}{3(x+1)^3}+\frac{(x-1)^5}{5(x+1)^5}+...\right),x>0;$$

Задание 2. Пользователь вводит с клавиатуры любую фразу (от 10 символов). Используя срезы выведите на экран:

а) первые 4 символа,

б) последние 4 символа,

в) символ посередине,

г) символы с 3-го по 8-й.

Задание 3. Статистика по списку из 6 элементов.

Пользователь вводит с клавиатуры последовательно (через запятые) 6 целых десятичных чисел – элементов списка. Требуется:

а) вывести на экран 4-й элемент,

б) вывести все элементы в обратном порядке,

в) рассчитать и вывести на экран сумму и среднее арифметическое.