Документация к языку программирования AlaVis.

1. Основы языка программирования. Синтаксис. Операторы ввода/вывода. Переменные и типизация данных. Преобразование данных. Математические операции с переменными. Сравнительный метод. Цикличный метод. Дополнительные методы.

1.1 Основы языка программирования. Синтаксис.
AlaVis- это язык программирования построенный на базе популярного языка Python. Предназначен для решения простых задач, для работы с внутренностями компьютера. Синтаксис языка очень прост в исполнении. Интерпретатор считывает построчно код и выполняет все последовательно (*сверху вниз по файлу* **.av**).

1.2 Операторы ввода/вывода.
Основными операторами языка являются операторы ввода и вывода какие либо данных:

- **<<#** оператор ввода данных
- **#>>** оператор вывода данных

```
#>> Hello, world!
```

1.3 Переменные и типизация данных. Преобразование данных.
Переменная - это как ярлык или контейнер, который хранит определенное значение или данные в программе.

Данные в **AlaVis**могут быть следующими типами:

- **int** целочисленные данные
- **float** вещественные данные
- **string** - строковые данные
- **bool** - булевые данные (*True или False*)

Пример создания переменной (*тип данных определяется автоматически как string*):

```
<<# num1 123
```

В данном примере мы создали переменную с именем **num1** и строковым значением по-умолчанию 123.

Явное указание типа данных при создании переменной:

```
<<# num1 int 123
```

В данном примере мы создали переменную с именем **num1** и целочисленным значением 123.

Метод **typed** - используется для проверки типа данных у заданной переменной.

Пример:

```
typed num1
```

Вывод:

```
typed num1 - int
```

Метод **formed** - используется для изменения (обновления) типа данных у заданной переменной. (например поменять string в float)

Пример:

```
formed num1 int
```

Вывод: нету

1.4 Математические операции с переменными.

Основными операциями над переменными являются:
+ сложение
- вычитание
* умножение
/ деление
% остаток от деления

Все математические операции записываются в квадратных скобках […]
Пример (*разность двух чисел типа int*):

```
<<# num1 int 123
<<# num2 int 12
<<# num3 int [ num1 - num2 ]
#>> num3
```

1.4 Сравнительный метод.

Сравнительный метод comp - используется для сравнения двух переменных (на равенство, больше или меньше и др).

Основные конструкции:
comp_equal - истина если две переменные равны.
comp_not_equal - истина если две переменные не равны.
comp_less - истина если переменная1 < переменной2.
comp_more - истина если переменная1 > переменной2.
comp_eless - истина если переменная1 <= переменной2.
comp_emore - истина если переменная1 >= переменной2.

конструкция завершение comp **::** - означает конец блока сравнительных операций.

(*иными словами - блок сравнительных операций продолжается до тех пор, пока не встретится специальный символ **::**, который означает конец блока.*)

```
<<# num1 int 123
<<# num2 int 12
<<# num3 int [ num1 - num2 ]
#>> num3

comp_equal num1 num2
typed num3
::
```

1.5 Цикличный метод.
Цикл в программировании - это конструкция, позволяющая выполнять определенный блок кода несколько раз. Это полезно, когда вам нужно повторять одни и те же действия несколько раз, например, для обработки множества данных или выполнения однотипных операций.

В **AlaVis** есть один единственный (*и простой*) цикл **loop**. (*работает примерно также как типичный while цикл*)

1.6 Дополнительные методы.
max - максимальное значение из..

```
<<# n1 float 5
<<# n2 float 0
<<# n3 float 1
<<# n float 0
max n n1 n2 n3
#>> n
```

min - минимальное значение из…

```
<<# n1 float 5
<<# n2 float 0
<<# n3 float 1
<<# n float 0
min n n1 n2 n3
#>> n
```

sqrt - квадратный корень от числа (auto - float)
пример - корень из числа 4

```
sqrt n 4
#>> n
```

pow - возведение числа в степень (auto - float)
пример - число 3 в степень 2. т.е 3^2

```
pow n 3 2
#>> n
```

abs - модуль от числа (auto - float)

```
abs n -5
#>> n
```

gcd - наибольший общий делитель (НОД) (auto - int)

```
gcd n n1 n2 n3
#>> n
```

lcm - наименьшее общее кратное (НОК) (auto - int)

```
lcm n n1 n2 n3
#>> n
```

del - удалить переменную

```
del n1
```

date - текущая дата
time - текущее время
x10x2 - перевод из 10-й системы счисления в 2-ю

```
<<# a 123

x10x2 a
#>> a
```

x10x8 - перевод из 10-й системы счисления в 8-ю

```
x10x8 a
#>> a
```

x10x16 - перевод из 10-й системы счисления в 16-ю

```
x10x16 a
#>> a
```

x2x10 - перевод из 2-й в 10-ю
x2x8 - перевод из 2-й системы счисления в 8-ю
x2x16 - перевод из 2-й в 16-ю

2. Модули. Подключение модулей. Работа с модулями.

2.1 Модули. Подключение модулей.

Модуль **MATH** - используется для расширенных математических операций.
Модуль **FILE** - используется для работы с файлами и директориями.
Модуль **SYSTEM** - используется для управления ОС.
Модуль **MEMORY** - используется для управления памятью компьютера.

Подключение модуля (*примеры*):

1. `/connect/ ? FILE`
2. `/connect/ ? MATH`

2.2 Модуль MATH.

Основные методы модуля:
sin - синус угла
cos - косинус угла
tan - тангенс угла
ctg - котангенс угла
arctan - арктангенс угла
arcctg - арккотангенс угла
pi - полная запись числа ПИ
integral - вычисление интеграла от выражения по dx
double_integral - двойной интеграл
factorial - факториал числа
log - логарифм
exp - экспонента
hypot - вычисление гипотенузы (теорема Пифагора)
round - округление числа

2.3 Модуль FILE.

open - открытие файлового потока (закрытие автоматическое)
read - чтение файла
write - запись в файл

awrite - дозапись в файл
fi - хранение считанных данных из файла
savefi - сохранение из **fi** в любую другую созданную переменную.

2.3 Модуль SYSTEM.
info - информация о данном компьютере
dir - текущая директория (путь)
change dir new_dir - смена директории
create folder name1 - создать папку в текущей директории
rename folder name1 new_name - переименовать папку
remove folder name1 - удалить папку

2.4 Модуль MEMORY.
disk - вывести информацию о всех дисках на компьютере