Лабораторная работа №1

Установка Python на компьютер

1. Установка Python на компьютер. Скачать питон: https://www.python.org/.
2. Как установить Python на Windows. Пошаговая инструкция по ссылке https://pythonru.com/baza-znanij/skachat-i-ustanovit-python-na-windows-10

1. Синтаксис

Синтаксис языка Python, как и сам язык, очень прост.

- Конец строки является концом инструкции (точка с запятой не требуется).
- Вложенные инструкции объединяются в блоки по величине отступов. Отступ может быть любым, главное, чтобы в пределах одного вложенного блока отступ был одинаков. И про читаемость кода не забывайте. Отступ в 1 пробел, к примеру, не лучшее решение. Используйте 4 пробела (или знак табуляции, на худой конец).
- Вложенные инструкции в Python записываются в соответствии с одним и тем же шаблоном, когда основная инструкция завершается двоеточием, вслед за которым располагается вложенный блок кода, обычно с отступом под строкой основной инструкции.

2. Переменные

Переменные невероятно важны, так как позволяют хранить информацию и использовать её в дальнейшем. На самом старте может быть не понятно, зачем вообще что-то записывать в переменную, если можно просто оперировать значениями без них. Тем не менее, понимание переменных придет немного позже, когда мы начнем создавать более сложные программы, и нам потребуется хранить информацию в каком-либо месте.

Типы переменных в языке Python не объявляются очевидно, тем не менее они здесь присутствуют. Интерпретатор Питона понимает, что записывается в переменную и на основании этого добавляет тип к этой переменной.

В ходе самой программы можно перезаписывать значение переменной, при этом можно указывать новый тип переменной. К примеру, изначально был записан тип float, но потом можно записать другой тип данных:

```
first_num = 23.2 # Тип данных float
first_num = "1" # Тип данных String
```

Тем не менее, если мы попытаемся добавить две переменные с разными типами данных, то это вызовет ошибку.

```
first_num = "Some"
second_num = 21
res = first_num + second_num # Здесь будет ошибка
```

Ниже приводим все существующие типы данных в языке Python:

- `some = 1` Integer - целые числа
- `some = 1.12` Float - числа с плавающей точкой
- `some = "Привет"` String – строки

Целые числа (int)

Числа в Python 3 ничем не отличаются от обычных чисел. Они поддерживают набор самых обычных математических операций:

x + y	Сложение
x - y	Вычитание
x * y	Умножение
x / y	Деление
x // y	Получение целой части от деления
x % y	Остаток от деления
-x	Смена знака числа
abs(x)	Модуль числа
divmod(x, y)	Пара (x // y, x % y)
x ** y	Возведение в степень
pow(x, y[, z])	x^y по модулю (если модуль задан)

Битовые операции

Над целыми числами также можно производить битовые операции

x | y	Побитовое *или*
x ^ y	Побитовое *исключающее или*
x & y	Побитовое *и*
x << n	Битовый сдвиг влево
x >> y	Битовый сдвиг вправо
~x	Инверсия битов

Математические функции в Python

Для работы с математическими функциями нужно импортировать библиотеку **math:**

После этого к функциям из этой библиотеки можно обращаться следующим образом:

math.имя_функции(…)

ceil(x)	Возвращает округленное x как ближайшее целое значение типа int, большее или равное x (округление "вверх").
fabs(x)	Возвращает абсолютное значение (модуль) числа x. В Python есть встроенная функция abs, но она возвращает модуль числа с тем же типом, что число, здесь же всегда float abs (fabs).
factorial(x)	Возвращает факториал целого числа x, если x не целое возбуждается исключение ValueError.
floor(x)	В противоположность ceil(x) возвращает округленное x как ближайшее целое значение типа int, меньшее или равное x (округление "вниз").
frexp(x)	Представляет число в экспоненциальной записи $x = m * 2^n$ и возвращает мантиссу m (действительное число, модуль которого лежит в интервале от 0.5 включительно до 1 не включительно) и порядок n (целое число) как пару чисел (m, n). Если x=0, то возвращает (0.0, 0)
fsum(iterable)	Возвращает float сумму от числовых элементов итерируемого объекта.
isinf(x)	Проверяет, является ли float объект x плюс или минус бесконечностью, результат соответственно True или False.
isnan(x)	Проверяет, является ли float объект x объектом NaN (not a number).
ldexp(x, i)	Возвращает значение $x * 2^i$, то есть осуществляет действие, обратное функции frexp(x).
modf(x)	Возвращает дробную и целую часть float числа. Оба результата сохраняют знак исходного числа x и представлены типом float.
trunc(x)	Возвращает целую часть числа x в виде int объекта.

Степенные и логарифмические функции

exp(x)	Возвращает e^x.
log(x[, base])	При передаче функции одного аргумента x, возвращает натуральный логарифм x (логарифм по основанию e = 2.7182...). При передаче двух аргументов, второй берется как основание логарифма.
log10(x)	Возвращает десятичный логарифм x.
pow(x, y)	Возвращает x в степени y. В отличие от операции ** приводит оба аргумента к типу float.
sqrt(x)	Квадратный корень (square root) из x.

Тригонометрические функции

acos(x)	Возвращает арккосинус x, в радианах.
asin(x)	Возвращает арксинус x, в радианах.
atan(x)	Возвращает арктангенс x, в радианах.
atan2(y, x)	Возвращает $atan(y/x)$, в радианах. Результат лежит в интервале [-π, π]. Вектор, конец, которого задается точкой (x, y) образует угол с положительным направлением оси x. Поэтому эта функция имеет более общее назначение, чем предыдущая. Например и atan(1), и atan2(1, 1) дадут в результате pi/4, но atan2(-1, -1) это уже -3*pi/4.
cos(x)	Возвращает косинус x, где x выражен в радианах.
hyp(x, y)	Возвращает sqrt(x**2+y**2). Удобно для вычисления гипотенузы (hyp) и длины вектора.
sin(x)	Возвращает синус x, где x выражен в радианах.
tan(x)	Возвращает тангенс x, где x выражен в радианах.

3. Условные операторы

Условные операторы помогают проверить несколько значений и выполнить код в зависимости от итога проверки. Вы можете создавать вложенные условные операторы, которые будут дополнительно проверять данные. Также можно проверять несколько условий в одном операторе. К примеру, чтобы проверить сразу два условия в одном операторе потребуется использовать операцию and (логическое и):

```
a = 2
if a != 0 and a != 1:
  print ("Проверка сработала")
```

Эта проверка сработает лишь в том случае, если оба условия окажутся как истинными (true).

Также можно проверять при помощи логического или or. В этом случае если хотя бы одно условие является верным (true), то код внутри условия будет выполнен:

```
a = 1.1
if a != 1.1 or a > 0:
  print ("Проверка сработала")
```

if <условие1>**:** <оператор1> [**elif** <условие2>**:** <оператор2>]* [**else:** <оператор3>]	Оператор "если". Часть в квадратных скобках является необязательной. Следующий за скобками символ "*" означает, что заключенная в скобки часть может быть записана неоднократно одна за другой. Здесь, при истинности <условия1> будет выполнен <оператор1> и проигнорированы ветки elif и else. В противном случае, если истинно <условие2>, то выполняется <оператор2>, ветка else игнорируется. Иначе выполняется <оператор3>.

Индивидуальные задания

В качестве *f(x)* использовать x^2. Переменные вводятся с клавиатуры. Необходимо проверить все ветви алгоритма.

1. $$a=\begin{cases}(f(x)+y)^2-\sqrt{f(x)y}, & xy\rangle 0\\ (f(x)+y)^2+\sqrt{|\mathrm{f(x)\ y}|}, & xy\langle 0\\ (f(x)+y)^2+1, & xy=0.\end{cases}$$

2. $$b=\begin{cases}\ln(f(x))+(f(x)^2+y)^3, & x/y\rangle 0\\ \ln|f(x)/y|+(f(x)+y)^3, & x/y\langle 0\\ (f(x)^2+y)^3, & x=0\\ 0, & y=0.\end{cases}$$

3. $$c=\begin{cases}f(x)^2+y^2+\sin(y), & x-y=0\\ (f(x)-y)^2+\cos(y), & x-y\rangle 0\\ (y-f(x))^2+tg(y), & x-y\langle 0.\end{cases}$$

4. $$d=\begin{cases}(f(x)-y)^3+arctg(f(x)), & x\rangle y\\ (y-f(x))^3+arctg(f(x)), & y\rangle x\\ (y+f(x))^3+0.5, & y=x.\end{cases}$$

5. $$e=\begin{cases}i\sqrt{f(x)}, & i-\textit{нечетное},\ x\rangle 0\\ i/2\sqrt{|f(x)|}, & i-\textit{четное},\ x\langle 0\\ \sqrt{|if(x)|}, & \textit{иначе}.\end{cases}$$

6. $$g=\begin{cases}e^{f(x)-|b|}, & 0.5\langle xb\langle 10\\ \sqrt{|f(x)+b|}, & 0.1\langle xb\langle 0.5\\ 2f(x)^2, & \textit{иначе}.\end{cases}$$

7. $$s=\begin{cases}e^{f(x)}, & 1\langle xb\langle 10\\ \sqrt{|f(x)+4*b|}, & 12\langle xb\langle 40\\ bf(x)^2, & \textit{иначе}.\end{cases}$$

8. $$j=\begin{cases}\sin(5f(x)+3m|f(x)|), & -1\langle m\langle x\\ \cos(3f(x)+5m|f(x)|), & x\rangle m\\ (f(x)+m)^2, & x=m.\end{cases}$$

9. $$l=\begin{cases}2f(x)^3+3p^2, & x\rangle |p|\\ |f(x)-p|, & 3\langle x\langle |p|\\ (f(x)-p)^2, & x=|p|.\end{cases}$$

10. $$k=\begin{cases}\ln(|f(x)|+|q|), & |xq|\rangle 10\\ e^{f(x)+q}, & |xq|\langle 10\\ f(x)+q, & |xq|=10\end{cases}$$

11. $$g=\begin{cases}e^{f(x)-|b|}, & 0.5\langle xb\langle 10\\ \sqrt{|f(x)+b|}, & 0.1\langle xb\langle 0.5\\ 2f(x)^2, & \textit{иначе}.\end{cases}$$

12. $$s=\begin{cases}e^{f(x)}, & 1\langle xb\langle 10\\ \sqrt{|f(x)+4*b|}, & 12\langle xb\langle 40\\ bf(x)^2, & \textit{иначе}.\end{cases}$$

13. $$g=\begin{cases}e^{f(x)-|b|}, & 0.5\langle xb\langle 10\\ \sqrt{|f(x)+b|}, & 0.1\langle xb\langle 0.5\\ 2f(x)^2, & \textit{иначе}.\end{cases}$$

14. $$b=\begin{cases}\ln(f(x))+(f(x)^2+y)^3, & x/y\rangle 0\\ \ln|f(x)/y|+(f(x)+y)^3, & x/y\langle 0\\ (f(x)^2+y)^3, & x=0\\ 0, & y=0.\end{cases}$$

Дополнительная литература:

Жуков, Р. А. Язык программирования Python: практикум: учебное пособие / Р.А. Жуков. — Москва : ИНФРА-М, 2021. — 216 с. + Доп. материалы [Электронный ресурс]. — (Высшее образование: Бакалавриат). — DOI 10.12737/textbook_5cb5ca35aaa7f5.89424805. - ISBN 978-5-16-016971-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/1412168 (дата обращения: 18.01.2022). – Режим доступа: по подписке.

! Глава 2 п.п. 2.1; Глава 3 п.п 3.3