**1 семинар**

*# вещественное деление*

6 / 5

*# целочисленное деление (без остатка)*

12 // 5

*# остаток от деления*

6 % 5

*# возведение в степень*

5 \*\* 2

Чтобы запустить **cntr+enter или можно print(a)**

**Alt+enter** запустить+ новая яейка

**Markdown** - можно писать текст

**#+текст** = жирный шрифт

**Int , float , complex type(число)**

*# модуль числа*

abs(-100)

*# Булевые переменные*

*type(True)*

bool

*# True - 1, False - 0*

*bool(0)*

False

*# None - объект для обозначения отсутствия значения*

*type(None)*

NoneType

x = 1

x += 2

x

Быструю справку можно получить, если щёлкнуть по нужной функции и нажать **shift+tab.**

**Логические операции**

* Равно (==)
* Меньше (<)
* Больше (>)
* Не равно (!=)

Результат логических операций — тип bool (True или False).

*# эквивалентно (1 < 2) and (2 < 3)*

1 < 2 < 3

**print(x, end=" ^\_^") или print(x, x, sep='\n', end='!')**

*x = input()*

*x, type(x)*

12

('12', str)

*x = int(input())*

*x, type(x)*

12

(12, int)

**x = input( )**

**print(f'Привет {x}')**

*Полина*

*Привет Полина*

**Записать число 8 подряд 228 раз, потом извлечь из него квадратный корень - что получится?**

x = '8'

y = x\*228

int(y)\*\* (1/2)

**import random**

**random.randint(0, 100)**

информация про принт <https://realpython.com/python-print/>

Функция **len — получить длину строки:**

len("You wil need to work hard hard hard")

У типа str большое количество встроенных строковых методов:

* .find — найти подстроку
* .upper — перевести строку в верхний регистр
* .lower — перевести строку в нижний регистр
* .strip — обрезать по краям поданные символы (по умолчанию пробельные)
* .startswith — вернуть True/False, начинается ли строка с введенных символов
* .endswith — вернуть True/False, заканчивается ли строка введенными символами
* .replace - замена подстроки на другую подстроку

print(prologue[prologue.find('Bran'):]) # напечатает всю строку, начиная с'Bran'

print(prologue[prologue.find('Bran'):prologue.find(' rode')]) # slicing

# replace

statement = '''I take you, Monica, to be my wife.\

I promise to be true to you in good times and in bad, in sickness and in health.\

I will love you and honour you all the days of my life.'''

statement.replace('Monica', 'Rachel')

**s = "abc"**

**s[0] a**

**s[:4]**

**s[-1] последнее число**

**s[1::2] все четные**

**s[::-1] perevernut**

Срез равный всей строке:

s[0:len(s):1]

start, finish, step могут иметь отрицательные значения

* start == -a ↔↔ start == len(s) - a
* finish == -a ↔↔ finish == len(s) - a
* step == -a ↔↔ проходить список в обратном направлении

**Списки**

**a = [1, 2, 3, 2, 0]**

Можно менять элементы списка, можно добавлять новые элементы и удалять старые (.append и .pop).

Два основных строковых метода для взаимодействия списков и строк:

* .split — разделить строку на список строк по определённому сепаратору
* .join — соеденить список строк в одну большую строку по определённому сепаратору

**2 семинар**

https://habr.com/ru/company/ods/blog/322626/

import pandas as pd

df = pd.read\_csv('./diamonds.csv', index\_col=0)

df.head(5)

type(df)

pandas.core.frame.DataFrame

у функции read\_csv, есть ряд важных параметров, которые иногда могут пригодиться

* sep — разделитель данных, по умолчанию ','
* decimal — разделитель числа на целую и дробную часть, по умолчанию'.'
* names — список с названиями колонок
* skiprows — если файл содержит системную информацию, можно просто её пропустить
* index\_col - номер колонки с индексом

если хотим разделить, то ставим запятую после индекса и вводим

**df.head(10**) с начала или с конца **df.tail()**

Посмотрим на размер нашего датасета. Первое число – количество строк (наблюдений), второе – количество столбцов

df.shape

У методы .info() можно передать дополнительные параметры, среди которых:

* verbose: печатать ли информацию о DataFrame полностью (если таблица очень большая, то некоторая информация может потеряться)
* memory\_usage: печатать ли потребление памяти (по умолчанию используется True, но можно поставить либо False, что уберёт потребление памяти, либо 'deep' , что подсчитает потребление памяти более точно)
* null\_counts: подсчитывать ли количество пустых элементов (по умолчанию True)

**df.info()**

**df.info(**memory\_usage=’deep’**)**

Можно вывести только тип данных в каждой колонке

**df.dtypes**

Метод describe показывает основные статистические характеристики данных по каждому числовому признаку (типы int64 и float64):

* число непропущенных значений
* среднее, стандартное отклонение
* минимум, максимум
* медиану, 0.25 и 0.75 квартили

**df.describe()**

Чтобы посмотреть статистику по нечисловым признакам (например, по строчным (object) или булевым (bool) данным), нужно явно указать интересующие нас типы в параметре include метода describe

**df.describe(include = ['object'])**

Из таблицы мы можем "достать" отдельный столбец, который будет уже другого специального типа pd.Series

Это можно сделать несколькими способами

* df['col']
* df.col – если название колонки на латинице без пробелов

**df['cut'] BUT**

**df[['cut',’karat’]]**

type(df.cut)

pandas.core.series.Series

df['cut'].mode()

df['carat'].median()

df['carat'].mean()

df['carat'].min()

df['carat'].max()

Например, посмотрим на среднюю цену бриллианта в зависимости от цвета.

**df.groupby('color')['price'].mean().sort\_values()**

**df.groupby('color')['price'].agg(['mean','median'])**

Например, можно посмотреть на среднюю цену бриллианта в зависимости от цвета (color) и качества (cut)

**df.pivot\_table(columns='color', index='cut', values='price', aggfunc='mean')**

**df['cut'].value\_counts(normalize=True) the same df['cut'].value\_counts() / df.shape(0)**

Просто посмотреть уникальные категории в колонке можно с помощью метода .unique()

**df['cut'].unique()**

А просто количество уникальных категорий (или значений) можно посмотреть с помощью метода .nunique()

**df['cut'].nunique()**

Визуализация

**df['cut'].value\_counts(normalize=True).plot(kind='bar')**

Добавим в таблицу новую колонку объем бриллианта (volume), используя длину, высоту и глубину брилланта

**df['volume'] = df['x']\*df['y']\*df['z']**

А колонки с длинной, высотой и глубиной удалим с помощью метода .drop()

* укажем колонки, которые нужно удалить – список drop\_cols
* axis=1 – означает, что удаляем колонки, а не строчки
* inplace=True – означает, что удаляем колонки в текущем датасете, а не выводим новый

**drop\_cols = ['x', 'y', 'z']**

**df.drop(drop\_cols, axis=1, inplace=True)** если не сделать инплэйс, то не удалиться, а просто выведет без колонки

**df\_save = df.copy()**  если хотим сохранить

Переведем также караты в миллиграммы (умножив массу в каратах на 200)

**df['carat'] = df['carat']\*200**

Переименуем колонку

**df.rename(columns={'carat':'weight'}, inplace=True)**

**Питонтьютор**

print('Как вас зовут?')

name = input() # считываем строку и кладём её в переменную name

print('Здравствуйте, ' + name + '!')

**Про электронные часы**

n = int(input())

n = n % (24\*60)

h = n // 60

m = n % 60

print(h,m)

**Следующее и предыдущее число**

a = int(input())

b = a+1

c = a-1

print('The next number for the number '+str(a)+' is '+str(b)+'')

print('The previous number for the number '+str(a)+' is '+str(c)+'')

**Шнурки**

a = int(input())

b = int(input())

L = int(input())

N = int(input())

print(2 \* L + (2 \* N - 1) \* a + 2 \* (N - 1) \* b)

if Условие:

Блок инструкций 1

else:

Блок инструкций 2

x = int(input())

y = int(input())

**В какой четверти находится**

if x > 0:

if y > 0: # x > 0, y > 0

print("Первая четверть")

else: # x > 0, y < 0

print("Четвертая четверть")

else:

if y > 0: # x < 0, y > 0

print("Вторая четверть")

else: # x < 0, y < 0

print("Третья четверть")

**x = int(input())**

**y = int(input())**

**if x > 0 and y > 0:**

**print("Первая четверть")**

**elif x > 0 and y < 0:**

**print("Четвертая четверть")**

**elif y > 0:**

**print("Вторая четверть")**

**else:**

**print("Третья четверть")**

**Про шахматы**

x1 = int(input())

y1 = int(input())

x2 = int(input())

y2 = int(input())

if (x1 + y1 + x2 + y2) % 2 == 0:

    print('YES')

else:

    print('NO')

**Про короля**

x1 = int(input())

y1 = int(input())

x2 = int(input())

y2 = int(input())

if abs(x1 - x2) <= 1 and abs(y1 - y2) <= 1:

    print('YES')

else:

    print('NO')

**Библиотека**import math

math.ceil(x) - округление к ближайшему целому

или from math import ceil

или from math import \*

x = 7 / 2

y = ceil(x)

print(y)

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| Округление | |
| **int(x)** | Округляет число в сторону нуля. Это стандартная функция, для ее использования не нужно подключать модуль math. |
| **round(x)** | Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа. |
| **round(x, n)** | Округляет число x до n знаков после точки. Это стандартная функция, для ее использования не нужно подключать модуль math. |
| **floor(x)** | Округляет число вниз («пол»), при этом floor(1.5) == 1, floor(-1.5) == -2 |
| **ceil(x)** | Округляет число вверх («потолок»), при этом ceil(1.5) == 2, ceil(-1.5) == -1 |
| **abs(x)** | Модуль (абсолютная величина). Это — стандартная функция. |
| Корни, логарифмы | |
| **sqrt(x)** | Квадратный корень. Использование: sqrt(x) |
| **log(x)** | Натуральный логарифм. При вызове в виде log(x, b) возвращает логарифм по основанию b. |
| **e** | Основание натуральных логарифмов e = 2,71828... |
| Тригонометрия | |
| **sin(x)** | Синус угла, задаваемого в радианах |
| **cos(x)** | Косинус угла, задаваемого в радианах |
| **tan(x)** | Тангенс угла, задаваемого в радианах |
| **asin(x)** | Арксинус, возвращает значение в радианах |
| **acos(x)** | Арккосинус, возвращает значение в радианах |
| **atan(x)** | Арктангенс, возвращает значение в радианах |
| **atan2(y, x)** | Полярный угол (в радианах) точки с координатами (x, y). |
|  |  |
| **degrees(x)** | Преобразует угол, заданный в радианах, в градусы. |
| **radians(x)** | Преобразует угол, заданный в градусах, в радианы. |
| **pi** | Константа *π* = 3.1415... |

**Семинар 3**

a = input('Введите первое число: ')

b = input('Введите второе число: ')

if a < b:

min\_n = a

else:

min\_n = b

print('Минимум равен', min\_n)

**Цикл while**

i = 1

while i <= 10:

print(i)

i += 1

**Break** внутри цикла позволяет прервать его выполнение и сразу же перейти к коду, который идет после цикла (либо завершить программу).

**Про оценки**

i = 1

while i <= 5:

mark = int(input("Введите оценку: "))

if mark < 4:

print('YES')

break

i += 1

else: # else находится на том же уровне отступа, что и while, поэтому относится именно к циклу, а не к условию внутри цикла

print ('NO')

Оператор **continue** позволяет сразу же перейти на новую итерацию цикла, не выполняя код, который написан внутри цикла ниже его.

i = 1

retakes = 0

while i <= 5:

mark = int(input("Введите оценку: "))

i += 1

if mark >= 4: # если пересдачи нет, сразу же идем проверять переменную i, без увеличения переменной retakes

continue

retakes += 1

print("Итого пересдач:", retakes)

**print(list(myRange))**

**print(tuple(myRange))**

print(list(range(1, 5)))

print(list(range(1, 10, 2)))

print(list(range(-5, -1)))

print(list(range(-5, -10, -2)))

[1, 2, 3, 4]

[1, 3, 5, 7, 9]

[-5, -4, -3, -2]

[-5, -7, -9]

**Цикл for**

students = ['Ivan Ivanov', 'Tatiana Sidorova', 'Maria Smirnova']

i = 0

while i < len(students):

print(students[i])

i += 1

**students = ['Ivan Ivanov', 'Tatiana Sidorova', 'Maria Smirnova']**

**for student in students:**

**print(student)**

students = ['Ivan Ivanov', 'Tatiana Sidorova', 'Maria Smirnova']

for i in range(len(students)): # здесь не надо преобразовывать range в список или кортеж

print(students[i])

<http://localhost:8888/notebooks/Desktop/sem03_for/sem03_193.ipynb> полезно

<http://localhost:8888/notebooks/Desktop/sem03_for/sem03_194.ipynb>

https://pythontutor.ru/lessons/inout\_and\_arithmetic\_operations/