**Основы Python**

**Про выполнение команд**

* Shift + Enter - выполнить код в ячейке и перейти на следующую
* Ctrl + Enter - выполнить и остаться

**Немного полезностей и про markdown:**

* перевести ячейку из кода в комментарии можно с помощью 'Esc + m' или просто 'm', если вы не еще не успели сделать ее активной
* добавить новую ячейку можно аналогичным образом через 'b'
* Jupyter понимает эти команды и в русской раскладке
* все это можно сделать и через верхнюю панель, но это дольше
* список горячих клавиш Ctrl+Shift+P (Command+Shift+P для мака)

**Арифметические операции**

print(1 **+** 1)

2

print(2 **-** 1, 5 **+** 6)

1 11

1 **+** 1 *# для вывода в jupyter notebook не обязательно использовать print*

2

1 **+** 1

2 **-** 1

1

print(1 **+** 1)

print(2 **-** 1)

2

1

5 **\*** 6

30

деление

10 **/** 4

2.5

Целочисленное деление

10 **//** 4

2

Возведение в степень

2 **\*\*** 5

32

Остаток от деления

4 **%** 3

1

(5 **-** 3) **+** 2 **\*** 6

14

**Переменные**

a **=** 10 **+** 20

b **=** a **\*** 30

c **=** a **/** b

print('Ответ: ', c)

Ответ: 0.03333333333333333

x **=** 10

print(x)

10

**Строки**

my\_string\_1 **=** 'Hello World'

my\_string\_2 **=** "Hello World!"

my\_string\_2

'Hello World!'

**Операторы сравнения**

2 **>** 1

True

10 **<** 9

False

10 **>** 10

False

10 **>=** 10

True

7 **==** 7

True

15 **!=** 15

False

5 **>** 4 **<** 10

True

*# сравнение строк*

print('c' **>** 'a')

True

print('b' **>** 'B')

True

print('abc' **>** 'aba')

True

len('3') **<** len('7')

False

**Условные конструкции**

x**=** 7

**if** x **%** 2 **==** 0:

print(x, '- четное число')

x **=** 5

**if** x **%** 2 **==** 0:

print(x, '- четное число')

**else**:

print(x, '- нечетное число')

5 - нечетное число

*# Пишем программу для определения куда направить призывника*

height **=** 180

**if** height **<** 170:

print('В танкисты')

​

print('Конец программы')

Конец программы

**if** height **<** 170:

print('В танкисты')

**else**:

print('В другие войска')

print('Конец программы')

*# условия выполняются по порядку, как только одно выполнено - проверки останавливаются*

height **=** 160

**if** height **<** 170:

print('В танкисты')

**elif** height **<** 200:

print('В десантники')

**else**:

print('В другие войска')

print('Конец программы')

В танкисты

Конец программы

*# блоков elif может быть хоть сколько*

*#height = 160*

height **=** int(input('Введите рост:'))

**if** height **<** 170:

print('В танкисты')

**elif** height **<** 180:

print('На флот')

**elif** height **<** 200:

print('В десантники')

**else**:

print('В другие войска')

print('Конец программы')

Введите рост:171

На флот

Конец программы

**Логические операторы and и or**

**True** **and** **True**

True

**True** **and** **False**

False

**True** **or** **False**

True

**not** **True**

False

((9 **>** 7) **and** (2 **<** 4))

True

(8 **==** 8) **or** (6 **!=** 6)

True

**not**(3 **<=** 1)

True

**if** age **>=** 18:

user\_input **=** int(input('Введите рост'))

**if** user\_input **<** 170:

print('В танкисты')

**elif** user\_input **<** 185:

print('На флот')

**elif** user\_input **<** 200:

print('В десантники')

**else**:

print('В другие войска')

**else**:

print('Непризывной возраст')

print('Конец программы')

Введите рост45

В танкисты

Конец программы

account\_amount **=** 1000

loan\_amount **=** 1500

days\_to\_pay **=** 6

is\_holiday **=** **False**

​

should\_notify **=** account\_amount **<** loan\_amount **and** ((days\_to\_pay **<** 10 **and** **not** is\_holiday) **or** (days\_to\_pay **==** 15))

should\_notify

account\_amount **<** loan\_amount **and** ((days\_to\_pay **<** 10 **and** **not** is\_holiday)

**or** (days\_to\_pay **==** 15))

*# функция input принимает пользовательский ввод*

name **=** input('Введите имя')

​

**if** name: *# пустая строка вернет False. Зачем все это? Не писать, что if name != ''*

print('Привет, ', name)

**else**:

print('Привет, Мир!')

number **=** int(input('Введите число'))

​

**if** number: *# 0 вернет False*

print('Число: ', number)

**else**:

print ('Число равно нулю')

Введите число0

Число равно нулю

# Простые типы данных

my\_integer **=** 10

type(my\_integer)

int

my\_float **=** 5.5

type(my\_float)

float

my\_string **=** 'Hello World!'

my\_string\_2 **=** "Hello World"

type(my\_string\_2)

str

my\_bool **=** **True**

*# my\_bool = False*

type(my\_bool)

bool

x **=** 5

y **=** 1

print(type(x **>** y))

<class 'bool'>

*# преобразование типов*

salary **=** 1000

print('Ваша годовая зарплата составляет ', salary, ' условных единиц')

Ваша годовая зарплата составляет 1000 условных единиц

print('Ваша годовая зарплата составляет ' **+** str(salary) **+** ' условных единиц')

Ваша годовая зарплата составляет 1000 условных единиц

*# неявное преобразование типов*

print(20 **/** 5.1)

print( **True** **and** **True**)

3.9215686274509807

True

## Строки

'Hi, ' **+** 'Oleg'

'Hi, Oleg'

my\_string**=**"Hello"

my\_string.upper()

'HELLO'

my\_string.lower()

'hello world!'

my\_string.capitalize()

'Hello world!'

my\_string.replace('Hello', 'Goodbye')

'Goodbye'

len(my\_string)

## f-строки

name **=** 'oleg'

lang **=** 'python'

t **=** f"Hello, {name.capitalize()}, i know {lang} a bit"

print(t)

print(type(t))

Hello, Oleg, i know python a bit

<class 'str'>

## Индексация и срезы

my\_string **=** 'Hello World'

my\_string[2]

'l'

my\_string[**-**4]

'o'

my\_string[0:4]

'Hell'

my\_string **=** 'Hello World'

my\_string[0:8:2]

'HloW'

my\_string[6:]

'World'

my\_string[:5]

'Hello'

## Упражнение

Выделите только из даты следующего формата

date **=** '2019-08-27T23:59:00.932'

print(date[0:10])

2019-08-27

## Проверка на вхождение элемента в объект

my\_string **=** 'Hello World'

target\_string **=** 'World'

**if** target\_string **in** my\_string:

print('find!')

find!

# Списки

month\_list **=** ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep']

income\_list **=** [13000, 14000, 14300, 15000, 13800, 13000, 14900, 15200, 15300]

income\_by\_months **=** [['Jan', 13000], ['Feb', 14000], ['Mar', 14300], ['Apr', 15000], ['May', 13800], ['Jun', 13000], ['Jul', 14900], ['Aug', 15200], ['Sep', 15300]]

print(type(month\_list))

print(type(income\_list))

print(type(income\_by\_months))

<class 'list'>

<class 'list'>

<class 'list'>

*# индексация элементов в списке*

print(month\_list[0])

print(month\_list[**-**1])

print(income\_by\_months[**-**4])

Jan

Sep

['Jun', 13000]

*# срезы*

print(income\_by\_months[0:2])

print('--------------')

print(income\_by\_months[**-**8:**-**6])

print('--------------')

print(income\_by\_months[2:])

print('--------------')

print(income\_by\_months[:3])

print('--------------')

print(income\_by\_months[:3])

[['Jan', 13000], ['Feb', 14000]]

--------------

[['Feb', 14000], ['Mar', 14300]]

--------------

[['Mar', 14300], ['Apr', 15000], ['May', 13800], ['Jun', 13000], ['Jul', 14900], ['Aug', 15200], ['Sep', 15300]]

--------------

[['Jan', 13000], ['Feb', 14000], ['Mar', 14300]]

*# можно обращаться к любому уровню вложенности*

income\_by\_months **=** [['Jan', 13000], ['Feb', 14000], ['Mar', 14300], ['Apr', 15000], ['May', 13800], ['Jun', 13000], ['Jul', 14900], ['Aug', 15200], ['Sep', 15300]]

income\_by\_months[3][1]

15000

*# изменение списков*

income\_by\_months[0][1] **=** 13100

print(income\_by\_months)

[['Jan', 13100], ['Feb', 14000], ['Mar', 14300], ['Apr', 15000], ['May', 13800], ['Jun', 13000], ['Jul', 14900], ['Aug', 15200], ['Sep', 15300]]

income\_by\_months[0:2] **=** [['Jan', 13200], ['Feb', 13900]]

print(income\_by\_months)

[['Jan', 13200], ['Feb', 13900], ['Mar', 14300], ['Apr', 15000], ['May', 13800], ['Jun', 13000], ['Jul', 14900], ['Aug', 15200], ['Sep', 15300]]

income\_by\_months\_2 **=** [['Nov', 15400], ['Dec', 17000]]

income\_by\_month **=** income\_by\_months **+** income\_by\_months\_2

print(income\_by\_month)

[['Jan', 13200], ['Feb', 13900], ['Mar', 14300], ['Apr', 15000], ['May', 13800], ['Jun', 13000], ['Jul', 14900], ['Aug', 15200], ['Sep', 15300], ['Nov', 15400], ['Dec', 17000]]

## Распаковка списков

first, second, third **=** ['первый', 'второй', 'третий']

first

'первый'

*# когда число элементов неизвестно*

first, **\***\_ **=** ['первый', 'второй', 'третий']

first, other

**---------------------------------------------------------------------------**

**NameError** Traceback (most recent call last)

**<ipython-input-48-4aecdbfca162>** in <module>

1 **# когда число элементов неизвестно**

2 first**,** **\***\_ **=** **['первый',** **'второй',** **'третий']**

**----> 3** first**,** other

**NameError**: name 'other' is not defined

first, **\***other, last **=** ['первый', 'второй', 'третий', 'четвертый']

first, last

('первый', 'четвертый')

## Операции со списками

income\_by\_months

[['Jan', 13200],

['Feb', 13900],

['Mar', 14300],

['Apr', 15000],

['May', 13800],

['Jun', 13000],

['Jul', 14900],

['Aug', 15200],

['Sep', 15300]]

*# Удаляем элемент по индексу*

**del**(income\_by\_months[**-**1])

income\_by\_months

[['Jan', 13200],

['Feb', 13900],

['Mar', 14300],

['Apr', 15000],

['May', 13800],

['Jun', 13000],

['Jul', 14900],

['Aug', 15200]]

*# удаляем элемент по значению*

month\_list.remove('Sep')

print(month\_list)

['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug']

*# добавляем элемент в конец списка*

income\_by\_months.append(['Dec', 17000])

income\_by\_months

[['Jan', 13200],

['Feb', 13900],

['Mar', 14300],

['Apr', 15000],

['May', 13800],

['Jun', 13000],

['Jul', 14900],

['Aug', 15200],

['Dec', 17000]]

*# добавляем элемент по нужному индексу*

income\_list.insert(2, 1111111)

print(income\_list)

[13000, 14000, 1111111, 14300, 15000, 13800, 13000, 14900, 15200, 15300]

*# считаем количество вхождений элемента в список*

income\_list.count(13000)

2

*# узнаем индекс элемента в списка (только первое вхождение!)*

income\_list.index(13000)

*# income\_list.index(13000, 1)*

0

*# разворачиваем список*

month\_list.reverse()

month\_list

['Aug', 'Jul', 'Jun', 'May', 'Apr', 'Mar', 'Feb', 'Jan']

*# узнаем длину списка*

len(income\_list)

10

*# сумма элементов*

sum(income\_list)

1239611

*# максимальный элемент элементов*

max(income\_list)

1111111

*# минимальный элемент элементов*

min(income\_list)

13000

*# сортировка по возрастанию*

sorted(income\_list)

*# изменить порядок сортировки*

sorted(income\_list, reverse**=** **True**)

*# а это сортировка строк по алфавиту*

sorted(month\_list)

['Apr', 'Aug', 'Feb', 'Jan', 'Jul', 'Jun', 'Mar', 'May']

## Изменение списков

В примере ниже переменная a и b на самом деле указывают на один и тот же объект. В результате, при добавлении в b очередного элемента этот элемент добавляется и в исходном листе a

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

b **=** a

b.append(4)

'a = {}'.format(a)

'a = [1, 2, 3, 4]'

id(a), id(b)

*# создаем копию объекта*

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

b **=** list(a)

b.append(4)

print(a)

print(b)

id(a), id(b)

[1, 2, 3]

[1, 2, 3, 4]

(1839176042048, 1839176000256)

Используем модуль copy

**import** copy

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

b **=** copy.copy(a)

id(a), id(b)

(1839176070208, 1839175468224)

b.append(4)

print('a = {}'.format(a))

​

print('b = {}'.format(b))

a = [1, 2, 3]

b = [1, 2, 3, 4, 4]

## Списки и строки

queries\_string **=** "смотреть сериалы онлайн,новости спорта,афиша кино,курс доллара,сериалы этим летом,курс по питону,сериалы про спорт"

*# преобразование строки в список (например, из CSV-файла)*

queries\_string.split(' ')

['смотреть',

'сериалы',

'онлайн,новости',

'спорта,афиша',

'кино,курс',

'доллара,сериалы',

'этим',

'летом,курс',

'по',

'питону,сериалы',

'про',

'спорт']

*# Преобразование списка в строку*

​

','.join(['Столбец 1', 'Столбец 2', 'Столбец 3'])

'Столбец 1,Столбец 2,Столбец 3'

*# проверка вхождения элемента в список:*

​

*# 'Москва' in ['Ленинград', 'Одесса', 'Севастополь', 'Москва']*

'Москва' **not** **in** ['Ленинград', 'Одесса', 'Севастополь', 'Москва']

False