Занятие 20

Numpy, Pandas

Что напечатают эти операторы?

```
a = {True : "1", 1 : "one", '1':'1', "1":"1"}; print(a)
a = {1: "1", True : "one"};  print(a)
b = {False: "0", 0: "zero"};  print(b)
b = {0: "0", False: "zero"};  print(b)

if 1:
    print('Yes')
else:
    print('No')
if 2:
    print('Yes')
else:
    print('No')
print (1 == True, 2 == True)
```

Задача 19-1

В кошельке лежат бумажные купюры 10, 50, 100, 200, 500, 1000, 2000, 5000 рублей, каждой купюры по одной штуке.

Какие суммы можно составить, если использовать этот набор купюр?

Подсказка. Используйте одну из наиболее подходящих функций модуля itertools.

Задача 19-2

Реализуйте класс Fibonacci, который реализует итератор, генерирующий бесконечную последовательность чисел Фибоначчи.

Например:

```
fibonacci = Fibonacci()
print(next(fibonacci))
print(next(fibonacci))
print(next(fibonacci))
print(next(fibonacci))
print(next(fibonacci))
```

Должен печатать следующие числа:

1

1

2

3

Задача 19-3

Определите класс Person. При создании объекта p = Person(‘Иванов’, ‘Михаил’, ‘Федорович’) необходимо ввести полное имя человека.

При печати объекта должно выводиться следующее:

print(p) # чивородеФлиахиМвонавИ

Магические методы сравнения

__eq__(self, other)

Определяет поведение оператора равенства ==

__ne__(self, other)

Определяет поведение оператора неравенства, !=

__lt__(self, other)

Определяет поведение оператора меньше, <

__gt__(self, other)

Определяет поведение оператора больше, >

__le__(self, other)

Определяет поведение оператора меньше или равно, <=

__ge__(self, other)

Определяет поведение оператора больше или равно, >=

Итератор

Итератор (iterator) – это объект, который используется для прохода по итерируемому элементу.

В основном используется для коллекций (списки, словари и т.д)

Протокол Iterator в Python включает две функции.

Один – iter(), другой – next().

И два дандера **__iter__ __next__**

import itertools

```
import itertools
dir(itertools)
# 'accumulate’, 'chain’, 'combinations’, 'combinations_with_replacement’, 'compress’, 'count’, 'cycle’, 'dropwhile', 'filterfalse’, 'groupby’, 'islice’, 'permutations’, 'product’, 'repeat’, 'starmap’, 'takewhile’, 'tee’, 'zip_longest'
```

itertools.permutations

```
import itertools
for x in itertools.permutations([1,2,3]):
        print(x, end = ‘ ‘)
# (1, 2, 3) (1, 3, 2) (2, 1, 3) (2, 3, 1) (3, 1, 2) (3, 2, 1)
```

itertools.combinations

```
import itertools
for x in itertools.combinations([1,2,3,4], 3):
        print(x, end = ‘ ‘)
# (1, 2, 3) (1, 2, 4) (1, 3, 4) (2, 3, 4)
# А если второй параметр 1, 2 ,4?
```

itertools.combinations_with_replacement

```
import itertools
for x in itertools.combinations_with_replacement([1,2,3,4], 3):
        print(x, end = ‘ ‘)
#(1, 1, 1) (1, 1, 2) (1, 1, 3) (1, 1, 4) (1, 2, 2) (1, 2, 3) (1, 2, 4) (1, 3, 3) (1, 3, 4) (1, 4, 4) (2, 2, 2) (2, 2, 3) (2, 2, 4) (2, 3, 3) (2, 3, 4) (2, 4, 4) (3, 3, 3) (3, 3, 4) (3, 4, 4) (4, 4, 4)
```

itertools.cycle

```
import itertools
x  = itertools.cycle([1,2,3,4])
for _ in range(10):
      print(next(x), end = ' ‘)
# 1 2 3 4 1 2 3 4 1 2
```

itertools.chain

```
import itertools
for x in itertools.chain([1,2,3,4], ‘abc’, {10:100, 20:200}):
        print(x, end = ‘ ‘)
# 1 2 3 4 a b c 10 20
```

itertools.zip_longest

```
list(zip([1, 2, 3], [11, 22, 33, 44])) # [(1,11), (2,22), (3,33)]

list(itertools.zip_longest([1, 2, 3], [11, 22, 33, 44], fillvalue = None))
# [(1, 11), (2, 22), (3, 33), (None, 44)]
```

itertools.groupby(iterable, key=None)

```
import itertools
s = 'aaabbbaaaaabbccccccd'
for k, v in itertools.groupby(s):
    print(k, ':', *v)
```

```
import itertools
s = 'aaa123bbbaaaaa345bbcccccd'
for k, v in itertools.groupby(s, key = lambda x: x.isdigit()):
    print(k, ':', *v)
```

```
import itertools
def odd(x):
    return x % 2
s = [1,2,4,6,3,3,10,8,4]
for k, v in itertools.groupby(s, odd):
    print(k,':', *v)
```

Задание

Есть список положительных и отрицательных чисел.

Постройте группы по знаку чисел, т.е. отдельно отрицательные, потом положительные, потом отрицательные и т.д. (или наоборот).

Т.е. если список [-1,-2,-3, 1,2,3,4, -10,-7, -5], то должно получиться три группы:

-1,-2,-3,

1,2,3,4,

-10,-7, -5

Numpy, Pandas

Numpy

NumPy — это библиотека Python, которую применяют для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй.

Полное название библиотеки — Numerical Python extensions, или «Числовые расширения Python».

(Предварительно: PyCharm: File / Settings / Project / Interpreter / + / numpy / Install package)

(pandas, matplotlib)

```
import numpy as np

arr = np.array([5, 3, 2, 0, 7, 12, 9, 1, 3, 4]) # В чем отличие массива np от списка Python? В чем сходство?
```

Отличия следующие:

- **Список может одновременно содержать элементы разных типов, массив - нет.** Например, вот типичный список [4, 7, 'альбом', ['1', '2', '3']] - он содержит одновременно числа, строку и еще один список. Если бы мы попытались создать такой массив, то получили бы ошибку.
- **Изменяемая размерность списка, постоянная у массива.** Мы часто пользуемся методом .append() у списка, чтобы добавить новый элемент. С массивом так поступить не получится, необходимо сразу создавать столько элементов, сколько планируете использовать в будущем.

```
print(arr.dtype) # int64 – тип данных одинаков для всех элементов
```

Как создать массив Numpy?

У массива есть две важные характеристики - **тип элементов** и **размерность**

```
import numpy as np
lst = [4, 5, 6]
arr1 = np.array(lst)                    # одномерный массив
lst = [[4, 5, 6], [7, 8, 9]]
arr2 = np.array(lst)                    # двумерный массив
 print(arr1.dtype, arr2.shape)   # тип данных и размерность массива
# В большинстве случае используются np.int64 для целых чисел и np.float64 для вещественных.
 arr1.shape # выполните для обоих массивов
```

Можно делать все арифметические операции с массивами - поэлементно

```
import numpy as np
a1 = np.array([1, 2, 3])
a2 = np.array([10, 20, 30])
print(a1 + a2) #   [11 22 33]
```

ones, zeros, shape, reshape

```
import numpy as np
a = np.zeros((2,3), dtype= int)
print(a)
print(a.shape)
b = a.reshape(6)
print(b)
b.shape
# Что напечатает?
b = a.reshape((3,2))
c = np.ones((3,4), dtype= int)
d = c.reshape(12)
# Сделайте те же манипуляции с c и d
```

Функции

```
import numpy as np
a1 = np.array([[1, 2, 3], [10, 20, 30]])
print(np.sum(a1))
print(np.sum(a1, axis = None)) # А если вместо None поставить 0? А если 1?
print(np.prod(a1, axis = None)) # А если вместо None поставить 0? А если 1?


np.around, np.sqrt, np.cbrt, np.abs, np.exp, np.gcd, np.lcm
print(np.gcd([6,24,72], [24,12,36])) # Что напечатает?
print(np.lcm([6,24,72], [24,12,36])) # А теперь?
```

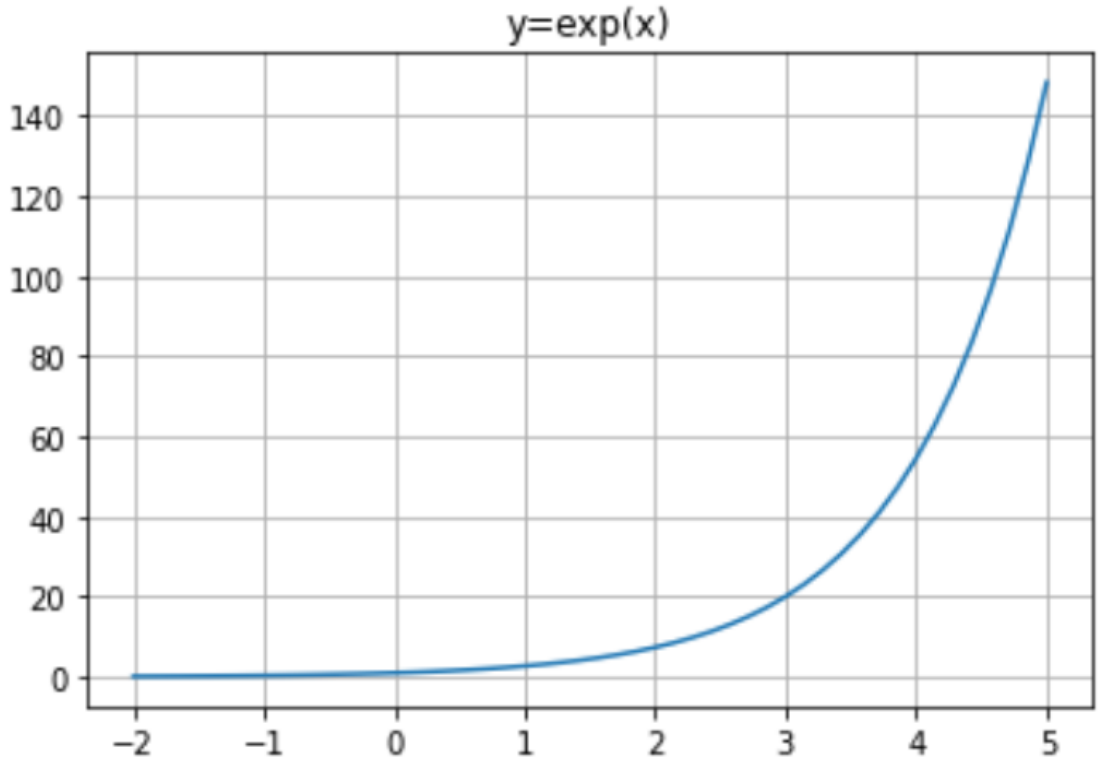
График np.exp

```
import numpy as np
import matplotlib.pyplot as plt


x = np.linspace(-2, 5, num=50)
y = np.exp(x)        # * 100


plt.plot(x, y)


plt.grid()
_ = plt.title("y=exp(x)")
plt.show()
```



Статистика

- **np.mean**(x, axis=None) - считает среднее, возвращает nan, если в массиве содержится константа np.nan
- **np.nanmean**(x, axis=None) - считает среднее, игнорирует nan
- **np.average**(x, axis=None, weights=None) - считает среднее, может вычислять взвешенное среднее, не обрабатывает np.nan
- **np.max**(x, axis=None) / **np.min**(x, axis=None) - возвращает максимум/минимум массива. Выбирает nan, если он содержится в массиве
- **np.nanmax**(x, axis=None) / **np.nanmin**(x, axis=None) - возвращает максимум/минимум массива. Игнорирует nan
- **np.median**(x, axis=None) - считает медиану, возвращает nan, если в массиве содержится константа np.nan
- **np.nanmedian**(x, axis=None) - считает медиану, игнорирует nan

Percentile

- **np.percentile(a, q, axis=None)** - в массиве a найти процентиль q, возвращает np.nan, если она содержится в массиве

- **np.nanpercentile(a, q, axis=None)** - в массиве a найти процентиль q, игнорирует np.nan

Логические операции

1. Сравнение массивов с помощью ==

```
import numpy as np
a1 = np.array([1, 2, 3])
a2 = np.array([2, 2, 2])
print(a1 == a2) # [False True False]
```

2. Сравнение массивов с помощью >

```
import numpy as np
a1 = np.array([1, 2, 3])
a2 = np.array([2, 2, 2])
print(a1 > a2) # [False False True]
```

3. Сравнение массивов с числами

```
import numpy as np
arr1 = np.array([3, 7, 5, 9])
print(arr1 > 5) # Результат # [False True False True]
```

4. Индексация булевскими массивами

```
arr = np.array([3, 4, 5, 1])
mean = np.mean(arr) # 3.25
condition = arr > mean # [False, True, True, False]
print(arr[condition]) # [4, 5]
```

Задание

Дан одномерный массив.

Выберите и напечатайте те элементы массива, которые строго меньше 25 процентиля.

```
import numpy as np
arr = np.array([2, 7, 7, 8, 8, 6, 8, 7, 6, 7])
condition = arr < np.percentile(arr, 25)
print(arr[arr < np.percentile(arr, 25)]) # или print(arr[condition])
```

Поиск, сортировка, выборка

1. np.where(condition, x1, x2) – по условию или из x1 или из x2

```
x = np.array([ 3, 5, 1, 8, 14, 17, 3, 10, 15, 18])
y = x * 10
print(np.where(x > 5, x, y)) # [30 50 10 8 14 17 30 10 15 18]
```

2. np.sort(x, axis= None)

```
arr = np.array([[19, 3, 9, 19, 10],[1, 4, 12, 16, 7]])
sorted_arr = np.sort(arr, axis = None) # 0, 1
print(sorted_arr)
```

Задания

1. Дана матрица, напечатайте количество строк в ней и количество столбцов

2. Вводится число n. Создайте двумерный массив Numpy со значениями от 1 до n * n

Например, для n = 3, результатом будет [[1,2,3], [4,5,6], [7,8,9]]

Сосчитать медианы для матрицы 3 * 3, для строк и для столбцов.

Pandas

Pandas — это библиотека Python для обработки и анализа структурированных данных, её название происходит от «panel data» («панельные данные»).

Панельными данными называют информацию, полученную в результате исследований и структурированную в виде таблиц.

Для работы с такими массивами данных и создан Pandas.

```
import pandas as pd
import numpy as np
import matplotlib as plt
```

Основные объекты pd.Series, pd.DataFrame

DataFrame

```
df1 = pd.DataFrame([[1,'Bob', 'Builder'],                    # из двумерного списка
                    [2,'Sally', 'Baker'],
                    [3,'Scott', 'Candle Stick Maker']],
        columns=['id','name', 'occupation'])
print(df1) # Выполните

df2 = pd.DataFrame({                                         # из словаря
    'country': ['Kazakhstan', 'Russia', 'Belarus', 'Ukraine'],
    'population': [17.04, 143.5, 9.5, 45.5],
    'square': [2724902, 17125191, 207600, 603628]
 })
print(df2) # Выполните
```

DataFrame

```
df = pd.DataFrame() # пустой
print(df)

lst = ['First', 'Second', 'Third']
print(pd.DataFrame(lst))

dct = {'ID': [101, 102, 203], 'Dept': ['Dev', 'BA', 'Test']}
print(pd.DataFrame(dct))

df['Subj'] = ['Java', 'Python', 'Pandas'] # Новая колонка

# Напечатайте и проверьте, что новая колонка появилась.
# Добавьте еще одну колонку
```

Задание

Создайте DataFrame из словаря:

```
dct = {'Год':[2001, 2002, 2003, 2004, 2005],
        'Товар':['A','B','C','D','E'],
        'Шт':[10, 20, 30,40,50],
        'Цена':[100,50, 30, 20, 5]
        }
```

Задание

Добавьте столбец ‘Итого’

df['Итого'] = df['Цена'] * df['Шт']

Проверьте, что колонка добавилась

excel

Импорт:

df_from_excel = pd.read_excel(‘test.xlsx’)

Экспорт:

df.to_excel('test.xlsx')

Запишите наш df в эксель, откройте его в df1 и проверьте

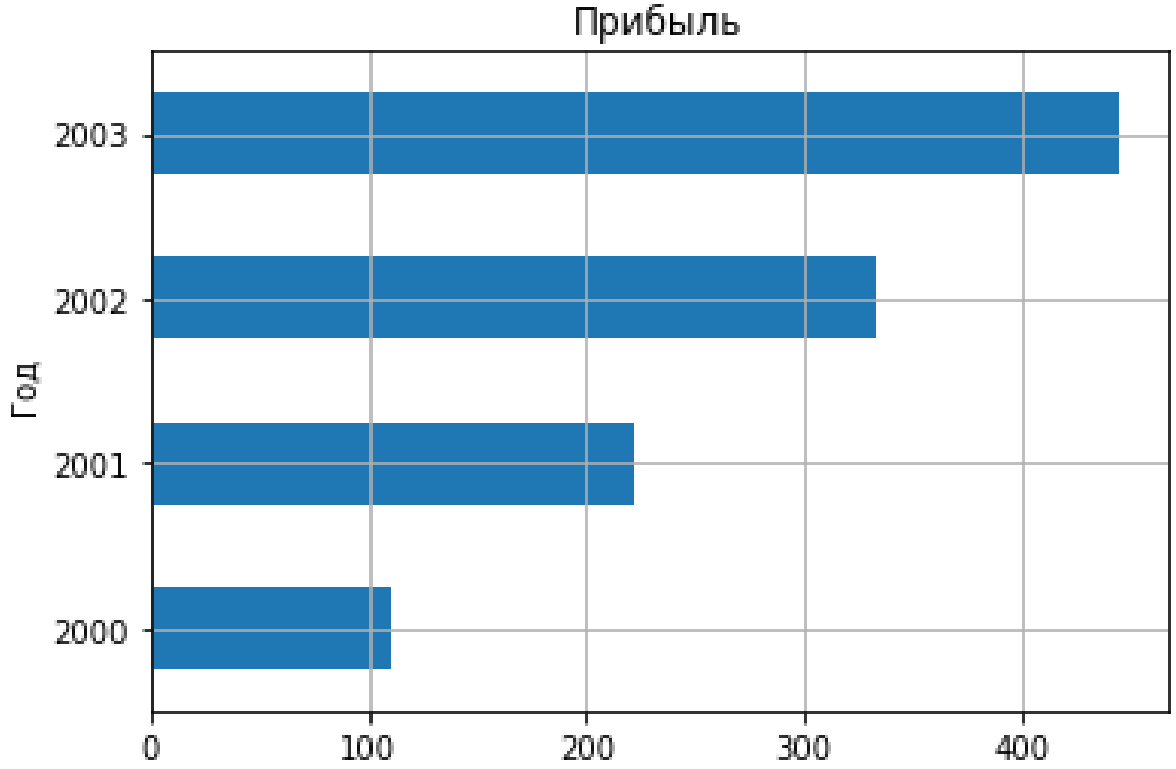
Гистограммы

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame({'Год':[2000, 2001, 2002, 2003],
                   'Прибыль': [111,222,333,444]})
df.index=df["Год"]

df['Прибыль'].plot(kind='barh')

plt.grid()
_ = plt.title("Прибыль")
plt.show()
```



Работа с данными – выполните каждую команду

```
df['a1'] # выбор одной колонки
df[['a1', 'b2']]  # выбор нескольких колонок
df.loc[0] # выбор одной строки
df.loc[0:2] # выбор нескольких строк
df[ df['Цена'] > 10] # выбор по условию
df[ (df['a1'] == 'x') | (df['a1'] == 'y')] # выбор по условию
df.iloc[0]
df[0:3]
New_df = df.T # Транспонирование
df.loc[0, 'Шт'] = 123
df.index
df.columns
df1 = df.set_index('Год')
df1.index
df1.loc[2002:2003]
df1.iloc[1:3]
df1.loc[2007] = ['F', 30, 100, 3000]
```

Задание

Напечатайте все элементы DataFrame

Воспользуйтесь df.index и df.columns для циклов

Воспользуйтесь df.loc для печати элемента

Выборки, информация – выполните эти операторы

```
df.head(3)    # первые три строки
df.tail(2)              # последние две
len(df)                 # количество строк в df
len(ratings['user_id'].unique()) # количество уникальных значений в столбце
 df.describe()                  # статистика о df
 df['Товар'].value_counts()   # количество значений в конкретном столбце
 df.columns             # список наименований колонок
 df.index               # список индексов
 df.loc[2000]           # конкретная строка
 df.sum()
 print(df1[['Год', 'Цена']].sum())
```

Изменение данных – выполните эти операции

```
anime['train set'] = True  # добавить новый столбец с конкретным значением
 df_new = df[['a1', 'b2']]   # новый дейта-фрейм из подмножества столбцов


df1 = anime[0:2]
df2 = anime[2:4]
pd.concat([df1, df2], ignore_index=True)


 df['new'] = df['a1'] + df['b2'] # Новая колонка
 df + df # Что получится?
```

Фильтрация

```
anime_modified.iloc[0:3] # первые три строчки, используется номер строки
anime[anime['type'] == 'TV’] # выбрать строчки по одном значению
anime[anime['type'].isin(['TV', 'Movie’])] # выбрать строчки по нескольким значениям
anime[anime['rating'] > 8]          # фильтрация по значению


anime.sort_values('rating', ascending=False) # сортировка


anime.groupby('type').count() # подсчитать сколько строчек с каждым значением в столбце
df.loc[(df.age > 50) & (df.gender == 'Female’)]
df.loc[(df.country == 'Poland') | (df.country == 'Czech Republic')]
```

Задания

- Найдите максимальную сделку
- Напечатать каждую вторую строку DataFrame
- Напечатать все строки, у которых цена меньше средней
- Дан df. Найдите минимальное значение для каждого столбца. Потом минимальное значение среди всех строковых значений и минимальное значение для всех числовых значений

https://www.postgresql.org/download/

Downloads

PostgreSQL Downloads

PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.

Packages and Installers

Select your operating system family:

Linux	macOS	Windows	BSD	Solaris

Source code

The source code can be found in the main file browser or you can access the source control repository directly at git.postgresql.org. Instructions for building from source can be found in the documentation.

Beta/RC Releases and development snapshots (unstable)

There are source code and binary packages of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used **for testing purposes only**, and not for production systems.

Задача 20-1

Человек приходит в кафе выпить чашку кофе, выбирая разные варианты напитка, иногда с пирожным, выбирая, что есть в меню.

Оплачивает иногда наличными деньгами, иногда картой.

Разработайте классы, их атрибуты, свойства и методы.

Может быть какую-то отчетность.

Задача 20-2

Напишите функцию, которая на вход получает DataFrame, который содержит числа и строки, а в качестве результата возвращает сумму всех чисел.

Задача 20-3

Создайте класс, экземпляр которого генерирует бесконечную циклическую последовательность из чисел и больших латинский букв.

1, A, 2, B, 3, C, .., X, 25, Y, 26, Z,1,A,2,B,3,C,..,X,25,Y,26,Z, 1, A…