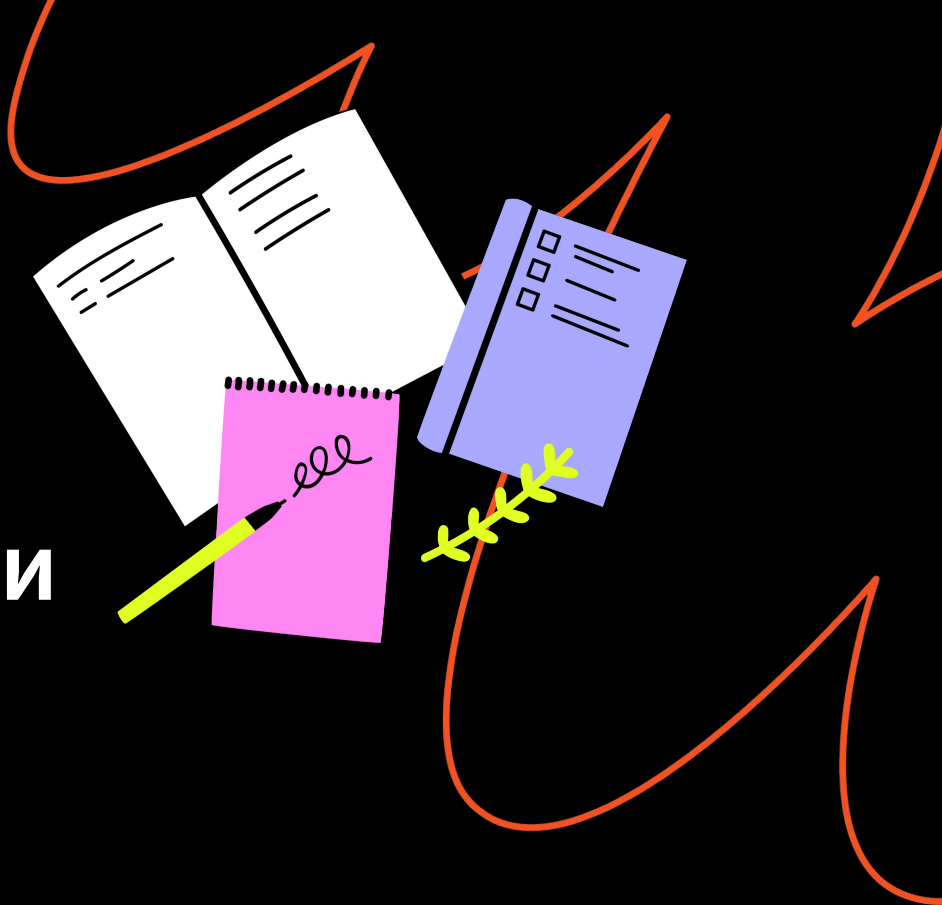




Данные, функции и модули в python

Основы основ и начало начал





Файлы



Файлы

Хранение данных

Передача данных в клиент-серверных проектах

Хранение конфигов

Логирование действий



Файлы

Как работать с файлами:

Связать файловую переменную с файлом,
определив модификатор работы

a – открытие для добавления данных

r – открытие для чтения данных

w – открытие для записи данных

w+, r+



Файлы

```
with open('file.txt', 'w') as data:
    data.write('line 1\n')
    data.write('line 2\n')

colors = ['red', 'green', 'blue']
data = open('file.txt', 'a')
data.writelines(colors) # разделителей не будет
data.close()

path = 'file.txt'
data = open(path, 'r')
for line in data:
    print(line)
data.close()
```



Демонстрация



Функции и модули



Функции

Это фрагмент программы, используемый
множественно

```
def function_name(x):  
    # body line 1  
    # . . .  
    # body line n  
    # optional return
```



Функции

```
def f(x):  
    return x**2
```

```
def f(x):  
    if x == 1:  
        return 'Целое'  
    elif x == 2.3:  
        return 23  
    else:  
        return
```

```
print(f(1))           # Целое  
print(f(2.3))         # 23  
print(f(28))          # None  
print(type(f(1)))     # str  
print(type(f(2.3)))   # int  
print(type(f(28)))    # NoneType
```



Функции

```
new file function_file.py file hello.py
```

```
def f(x):  
    if x == 1:  
        return 'Целое'  
    elif x == 2.3:  
        return 23  
    else:  
        return
```

```
import function_file  
  
print(function_file.f(1))      # Целое  
print(function_file.f(2.3))    # 23  
print(function_file.f(28))     # None
```



Функции

```
new file function_file.py file hello.py
```

```
def f(x):  
    if x == 1:  
        return 'Целое'  
    elif x == 2.3:  
        return 23  
    else:  
        return
```

```
import function_file as ff  
  
print(ff.f(1))           # Целое  
print(ff.f(2.3))        # 23  
print(ff.f(28))         # None
```



Функции

```
def new_string(symbol, count):  
    return symbol * count
```

```
print(new_string('!', 5))    # !!!!!  
print(new_string('!'))       # TypeError missing 1 required ...
```



Функции

```
def new_string(symbol, count = 3):  
    return symbol * count
```

```
print(new_string('!', 5))    # !!!!!  
print(new_string('!'))       # !!!  
print(new_string(4))         # 12
```



Функции

```
def concatenatio(*params):  
    res: str = ""  
    for item in params:  
        res += item  
    return res
```

```
print(concatenatio('a', 's', 'd', 'w')) # asdw  
print(concatenatio('a', '1', 'd', '2')) # a1d2  
# print(conatenatio(1, 2, 3, 4)) # TypeError: ...
```



Рекурсия?



Функции

```
def fib(n):  
    if n in [1, 2]:  
        return 1  
    else:  
        return fib(n-1) + fib(n-2)
```

```
list = []  
for e in range(1, 10):  
    list.append(fib(e))  
print(list) # 1 1 2 3 5 8 13 21 34
```



Кортежи



Кортежи

Кортеж – это неизменяемый “список”

```
t = ()
print(type(t))           # tuple
t = (1,)
print(type(t))           # tuple
t = (1)
print(type(t))           # int
t = (28, 9, 1990)
print(type(t))           # tuple
colors = ['red', 'green', 'blue']
print(colors)             # ['red', 'green', 'blue']
t = tuple(colors)
print(t)                  # ('red', 'green', 'blue')
```



Кортежи

```
t = tuple(['red', 'green', 'blue'])
print(t[0])      # red
print(t[2])      # blue
# print(t[10])   # IndexError: tuple index out of range
print(t[-2])     # green
# print(t[-200]) # IndexError: tuple index out of range

for e in t:
    print(e)      # red green blue

t[0] = 'black'    # TypeError: 'tuple' object does not support
item assignment
```



Кортежи

```
t = tuple(['red', 'green', 'blue'])
red, green, blue = t
print('r:{} g:{} b:{}'.format(red, green, blue))
# r:red g:green b:blue
```



Словари



Словари

Неупорядоченные коллекции произвольных объектов с доступом по ключу

```
dictionary = {}  
dictionary = \  
    {  
        'up': '↑',  
        'left': '←',  
        'down': '↓',  
        'right': '→'  
    }  
  
print(dictionary)  # {'up': '↑', 'left': '←', 'down': '↓', 'right': '→'}  
print(dictionary['left'])  # ←  
# типы ключей могут отличаться
```



Словари

```
print(dictionary['up'])    # ↑  
# типы ключей могут отличаться
```

```
dictionary['left'] = '←'  
print(dictionary['left']) # ←  
#print(dictionary['type']) # KeyError: 'type'  
del dictionary['left'] # удаление элемента
```

```
for item in dictionary:    # for (k,v) in dictionary.items():  
    print('{}: {}'.format(item, dictionary[item]))  
# up: ↑  
# down: ↓  
# right: →
```



Множества



Множества

Неупорядоченная совокупность элементов

```
a = {1, 2, 3, 5, 8}
b = {'2', '5', 8, 13, 21}
print(type(a))    # set
print(type(b))    # set
```



Множества

```
a = {1, 2, 3, 5, 8}
b = set([2, 5, 8, 13, 21])
c = set((2, 5, 8, 13, 21))

print(type(a))  # set
print(type(b))  # set
print(type(c))  # set

a = {1, 1, 1, 1, 1}
print(a)  # {1}
```



Множества

```
colors = {'red', 'green', 'blue'}
print(colors)          # {'red', 'green', 'blue'}
colors.add('red')
print(colors)          # {'red', 'green', 'blue'}
colors.add('gray')
print(colors)          # {'red', 'green', 'blue', 'gray'}
colors.remove('red')
print(colors)          # {'green', 'blue', 'gray'}
# colors.remove('red') # KeyError: 'red'
colors.discard('red')   # ok
print(colors)          # {'green', 'blue', 'gray'}
colors.clear()          # { }
print(colors)          # set()
```



Множества

```
a = {1, 2, 3, 5, 8}
b = {2, 5, 8, 13, 21}
c = a.copy()           # c = {1, 2, 3, 5, 8}
u = a.union(b)         # u = {1, 2, 3, 5, 8, 13, 21}
i = a.intersection(b)  # i = {8, 2, 5}
dl = a.difference(b)   # dl = {1, 3}
dr = b.difference(a)   # dr = {13, 21}

q = a \
    .union(b) \
    .difference(a.intersection(b))
# {1, 21, 3, 13}
```



Множества

Неизменяемое множество

```
a = {1, 2, 3, 5, 8}
b = frozenset(a)
print(b) # frozenset({1, 2, 3, 5, 8})
```



Списки



ИТОГИ



Итоги

Как работать с файлами

Как создаются методы

Зачем нужны модули

Напомнили себе рекурсию с python-особенностями

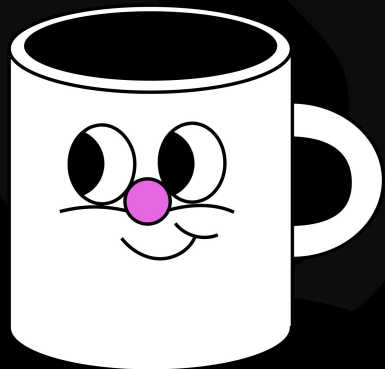
Научились хранить данные в:

- Списки

- Кортежи

- Словари

- Множества





Спасибо
за внимание

A yellow hand-drawn smiley face is positioned to the right of the text. It has two vertical lines for eyes and a curved line for a smile, partially overlapping the word 'Спасибо' and the word 'за'.