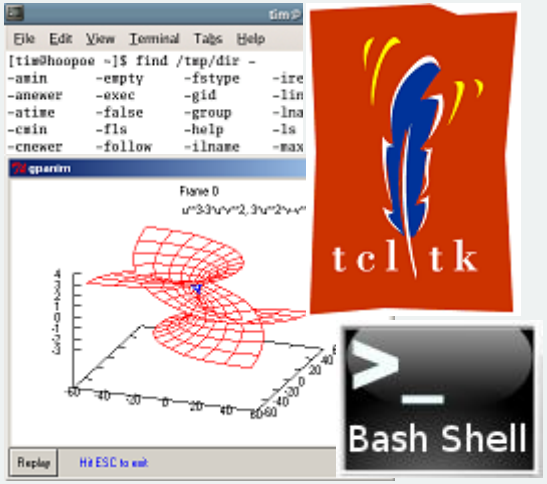
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Интерпретируемые языки программирования



Лабораторная работа №5

Основы языка Python.

Скрипт на языке Python

Создание файла

```
touch ./script.py
```

Написание кода

```
#!/usr/bin/python

print "Hello from python! :)"
```

Установка прав

```
chmod +x ./script.py
```

Запуск

```
./script.py
```

Вывод данных на консоль (Python 3)

Для вывода данных на консоль используется функция print

```
print ("Hello, world!")

print ('Hello, world!')

print ("Hello,", 'world!')
```

Для вывода данных на консоль без перевода строк можно использовать следующие варианты записи:

```
import sys

sys.stdout.write('Hello,')
sys.stdout.write("world")
```

```
print ('Hello', end='')
print ("world")
```

Форматированный вывод данных на консоль: Python2

```
s1 = 'A'
print ("Value: %s" % s1)
```

```
s1 = 'A'
s2 = 'B'
print ("Values are: %s and %s" % (s1, s2))
```

```
s1 = 'A'
s2 = 'B'
print ("Value1: %s" % s1, "Value2: %s" % s2)
```

```
s1 = 'A'
s2 = 'B'
print ("Values are: %(a)s and %(b)s" % {‘a’ : s1, ‘b’ : s2})
```

Форматированный вывод данных на консоль: Python3

Использование форматирования в функции print:

```
s1 = 'A'
s1 = 'B'

print("Values: {} {}".format(s1, s2))


print("Values: {0} {1}".format(s1, s2))


print("Values: {1} {0}".format(s1, s2))


print("Values: {a} {b}".format(a=s1, b=s2))
```

Аргументы функции print

```
print(*items, sep=' ', end='\n', file=sys.stdout, flush=False)
```

Получение информации о типе данных переменной

```
x=1
y=3.1415

print("Тип X='{}'\nТип Y='{}'".format( type(x), type(y) ))
```

Чтение данных с консоли (Python 3)

```
val = input("Enter value:")
print("Val = {}".format(val))

print("Val={}\nVal type={}".format(val, type(val)))
```

Функции приведения типов:

```
str()
int()
long()
float()
```

Приведение типов

```
x = input("Enter x value: ")
y = input("Enter y value: ")

print("x={}\ny={}".format(x, y))

print("x type={}, y type={}".format( type(x), type(y) ))

z = int(x) + int(y)

print("z={}".format(z))
print("type z={}".format( type(z) ))
```

Условный оператор if (1)

```
if var1 == var2:
        print "var1 = var2"
else:
        print "var1 != var2"
```

Операторы сравнения:

==
!=
<
>
<=
>=

```
if var1 > var2 and var2 < var3:
        print "var2 is in the middle"
```

Логические операторы:

and
or
not

Условный оператор if

```
if var1 == var2:
        print "var1 = var2"
else:
        if var1 < var2:
                print "var1 < var2"
        else:
                print "var1 > var2"
```

```
if var1 == var2:
        print "var1 = var2"
elseif var1 < var2:
        print "var1 < var2"
else:
        print "var1 > var2"
```

Данные в python

- числа
  - целые — x = 4
  - вещественные — y = 3.1415
  - комплексные — z = complex(1, 4)
- строки
- списки
- множества
- кортежи
- словари

Операции над числами в python (1)

Математические операции над числами

Операция	Значение
`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)

Операции над числами в python (2)

Побитовые операции над **целыми** числами:

Операция	Значение
`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

Операции над числами в python (3)

```
x = 4131

print("Dec x = {}, Bin x = {}".format( x, bin(x) ))
x = x >> 2
print("Dec x = {}, Bin x = {}".format( x, bin(x) ))

y = 2.71828
res1 = x * y
print(res1)

c = complex(2, -3)
res2 = c*(x - y)
print(res2)
```

```
E:\>python test.py
Dec x = 4131, Bin x = 0b1000000100011
Dec x = 1032, Bin x = 0b10000001000
2805.26496
(2058.56344-3087.84516j)
```

Работа со строками

```
strval = "PKIMS rulez!"

print(len(strval))

print(strval[2])

print(strval[1:])

print(strval[1:4])

print(strval[-3])

print(strval[-3:])

print(strval[3:-3])
```

Некоторые строковые методы

Метод	Описание
`<str>.center(<width>)`	возвращает копию <str>, выровненную по центру до <width>
`<str>.count(<sub> [, <start> [, <end>]])`	Считает число вхождений подстрок <sub> с возможным указанием диапазона
`<str>.startswith(<sub>)` `<str>.endswith(<sub>)`	Возвращает True, если строка начинается на <sub> / оканчивается на <sub>
`<str>.find(<sub> [, <start> [, <end>]])`	Ищет <sub> в строке <str> с возможным указанием диапазона, возвращает индекс или -1
`<str>.isalpha(), <str>.isdigit()`	Проверяет строку на то, что она является только символьной / только цифровой
`<str>.upper(), <str>.lower()`	Возвращает копию <str>, переведённую в верхний / нижний регистр
`<str>.split([sep])`	Разбивает строку по [sep], если не задан – по пробельным символам
`<str>.replace(<old>, <new> [, <count>])`	Замена подстрок в строке

Сырые строки в Python

```
f = open("/home/topgun/test.txt", 'r')
text = f.read()
print(text)
```

```
f = open("D:\\Test\\test.txt", 'r')
text = f.read()
print(text)
```

```
f = open("D:/Test/test.txt", 'r')
text = f.read()
print(text)
```

```
f = open(r"D:\Test\test.txt", 'r')

text = f.read()

print(text)
```

Форматированные строки в Python

```
print("Val = {}".format(val))

print(f"Val = {val}")
```

Списки в Python

```
l = [1, 2, 3, 4, 5, 6]
print(l)

for i in l:
        print (i)
```

```
l = []
print(l)

l.append(4)
print(l)
```

```
s = "ПКИМС рулит"
l = list(s)
print(l)
```

```
['П', 'К', 'И', 'М', 'С', ' ', 'р', 'у', 'л', 'и', 'т']
```

```
l = [1, 2, 3, 4, 5]

print(len(l))
print(l[0])
```

СПИСКИ, кортежи, множества, словари (1)

Список (List) – самая часто используемая структура данных в Python, аналог массивов с динамической типизацией в компилируемых языках программирования. Элементы индексируются.

```
l = list()
l = [ 1, 2, 3, 4, 5 ]

l.append(6)
l.append(5)

l[2]

l.count(5)

l.pop(0)

l.remove(4)

...
```

Списки, КОРТЕЖИ, множества, словари (2)

Кортеж (Tuple) – аналог списка, элементы которого нельзя изменить
Элементы индексируются.

```
t = tuple()
t = (1, 2, 3, 4, 5, 3, 4, 5)

t.count(3)

t.index(2)
t.index(2, 0)
t.index(2, 0, 4)
```

Списки, кортежи, МНОЖЕСТВА, словари (3)

Множество (Set) – набор уникальных элементов, порядок их следования не регламентирован.

```
s = set()
s = {1, 2, 3, 4, 5, 3, 4, 5}

s.add(6)

s.remove(5)

s.difference({3, 4})

s.intersection({3, 4, 5, 6})

s.symmetric_defference({3, 4, 5, 6})

...
```

Списки, кортежи, множества, СЛОВАРИ (4)

Словарь (Dictionary) – неупорядоченный список элементов с доступом по ключу.
Также – ассоциативные массивы, также хэш-таблицы.

```
d = dict()
d = {name='Dmitry', surname='Bulakh'}
d = {'name' : 'Dmitry', 'surname' : 'Bulakh'}

d['name']
d.get('name')

d['name'] = 'Dmitiy'

...
```

Цикл for

```
for i in range(10):
        print(i)
```

```
for i in range(1, 10):
        print(i)
```

```
for i in range(1, 10, 2):
        print(i)
```

Работа с файлами: чтение текстовых файлов

```
f = open("myfile.txt", 'r')

text = f.read()

print (text)

f.close()
```

```
f = open("myfile.txt", 'r')

for line in f:
  print (line)

f.close()
```

```
f = open("myfile.txt", 'r')

while True:
  line = f.readline()
  if line == "":
    break
  print (line)

f.close()
```

Работа с файлами: запись текстовых файлов

```
f = open("data.txt", 'w')
for count in range(1, 100, 5):
    f.write("Number is : ")
    f.write(str(count) + "\n")
f.close()
```

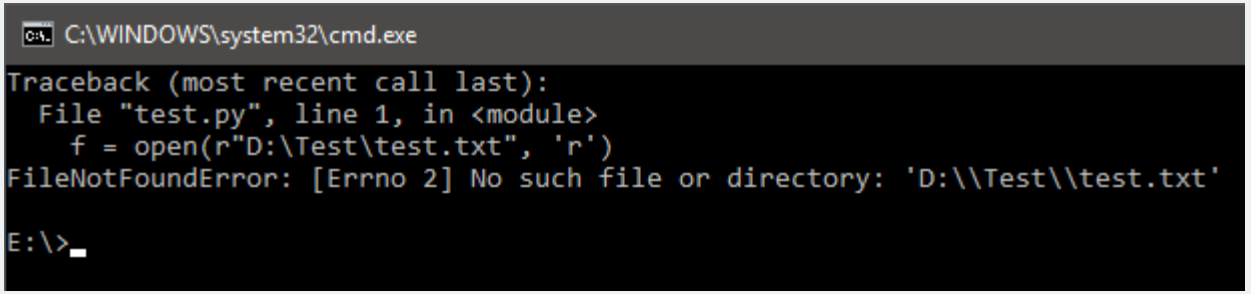
```
f = open("data.txt", 'w')
for count in range(1, 100, 5):
        print("Number is : {}".format(count), file=f)
f.close()
```

Поэлементное чтение файлов

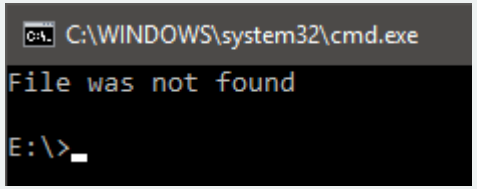
```
f = open("data.txt", 'r')

for line in f:
  for token in line.rstrip().strip():
    print(token)

f.close()
```

Обработка исключений (1)

```
f = open(r"D:\Test\test.txt", 'r')
text = f.read()
print(text)
```




```
try:
        f = open(r"D:\Test\test.txt", 'r')
        text = f.read()
        print(text)
        f.close()
except FileNotFoundError:
        print("File was not found")
```



Обработка исключений (2)

```
try:
        f = open(r"D:\Test\test.txt", 'r')
        text = f.read()
        print(text)
        f.close()
except FileNotFoundError:
        print("File was not found")
```

```
try:
        f = open(r"D:\Test\test.txt", 'r')
except FileNotFoundError:
        print("File was not found")
else:
        text = f.read()
        print(text)
        f.close()
```

Обработка исключений (2)

```
x = 4

try:
        c = x/0
except ZeroDivisionError:
        print("Zero division detected")
else:
        print("Everything is OK")
finally:
        print("Will execute anyway")
```

Обработка исключений (3)

```
x = 'string'

try:
        c = x/0
except TypeError:
        print("Type error detected")
except ZeroDivisionError:
        print("Zero division detected")
except Exception:
        print("Base exception")
else:
        print("Everythong is OK")
finally:
        print("Will execute anyway")
```

Использование модулей (1)

```
import os

print(os.getlogin())
print(os.getcwd())
```

```
import os
from sys import platform

print(os.name)
print(platform)
```

```
import os, sys

print(os.name)
print(sys.platform)
```

```
import os
import sys
from sys import platform

print(sys.argv)
print(os.name)
print(platform)
```

Использование модулей (2)

```
import os
import sys
from sys import platform


print(sys.argv)
print(os.name)
print(platform)
```

```
import sys
from sys import *


print(argv)
print(os.name)
print(platform)
```

```
from sys import argv as args

print(args)
```

Регулярные выражения: модуль 're' (1)

```
import re

ret = re.match('a+', 'aaababbccc')

if ret:
  print('Yes')
else:
  print('No')
```



```
if ret:
  print('Yes')
  print(ret.group())
else:
  print('No')
```

```
import re

ret = re.fullmatch('a+', 'aaababbccc')

if ret:
  print('Yes')
  print(ret.group())
else:
  print('No')
```

Регулярные выражения: модуль 're' (2)

```
import re

ret = re.search('b+', 'aaababbccc')

if ret:
  print('Yes')
  print(ret.group())
else:
  print('No')
```



```
if ret:
  print('Yes')
  print(ret.group())
  print(ret.start())
  print(ret.end())
else:
  print('No')
```

```
import re

ret = re.findall('b+', 'aaababbccc')

if ret:
  print('Yes')
  print(ret)
else:
  print('No')
```



```
if ret:
  print('Yes')
  print(ret)
  print(len(ret))
  print(ret[0])
else:
  print('No')
```

Решение задачи поиска модулей

```
import re

str = 'module inv(x, y);'

ret = re.search('module\s+(\w+)\s*\(', str)

if ret:
  print('Yes')
  print(ret.group(0))
  print(ret.group(1))
else:
  print('No')
```

Функции (1)

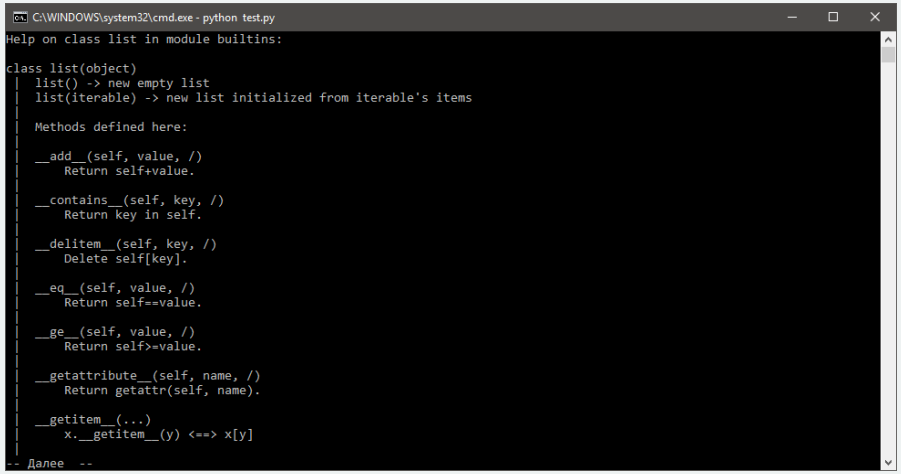
```
def func1(a, b, strval):
  c = a+b
  print (strval)
  return c



sum = func1(4, 5, "String value!")
print sum
```

```
def func1(a, b = 45):
  c = a+b
  return c


sum = func1(4)
print (sum)
```

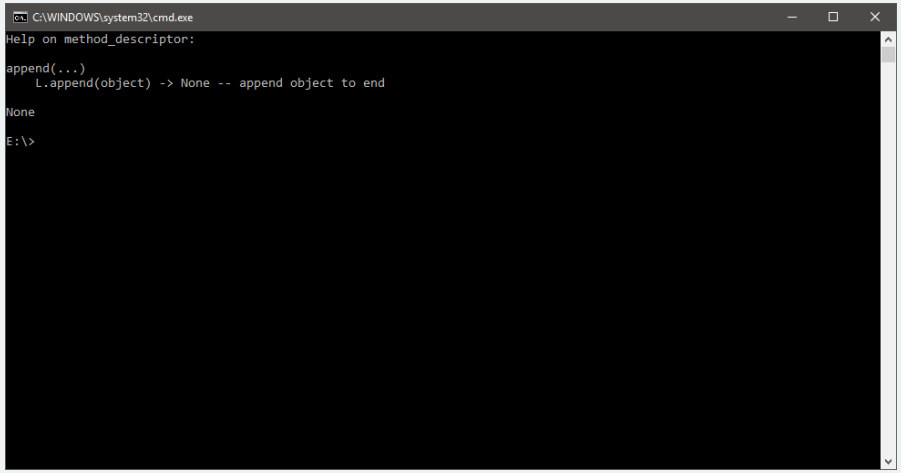
```
def func1(a, b, strval):
  c = a+b
  print (strval)
  return c


sum = func1(strval ="String", a=4, b=6)
print sum
```

Документирование программного кода (1)

```
print(help(list))
```

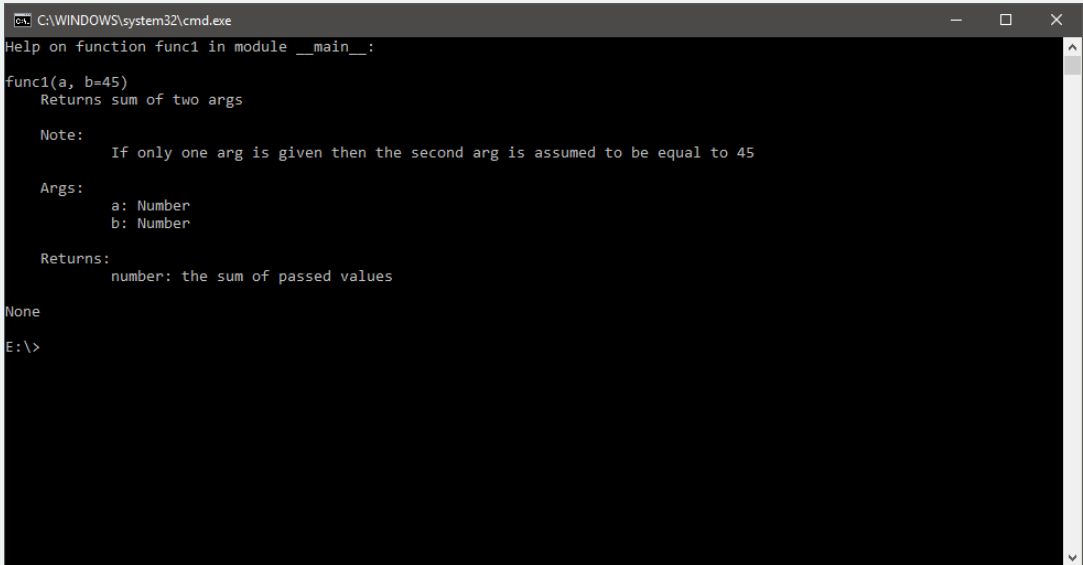



```
print(help(list.append))
```



Документирование программного кода (2)

```
def func1(a, b = 45):
        """
        Returns sum of two args

        Note:
                If only one arg is given then the second arg is
                assumed to be equal to 45

        Args:
                a: Number
                b: Number

        Returns:
                number: the sum of passed values
        """
        c = a+b
        return c


print(help(func1))
```



Возврат нескольких значений из функции (1)

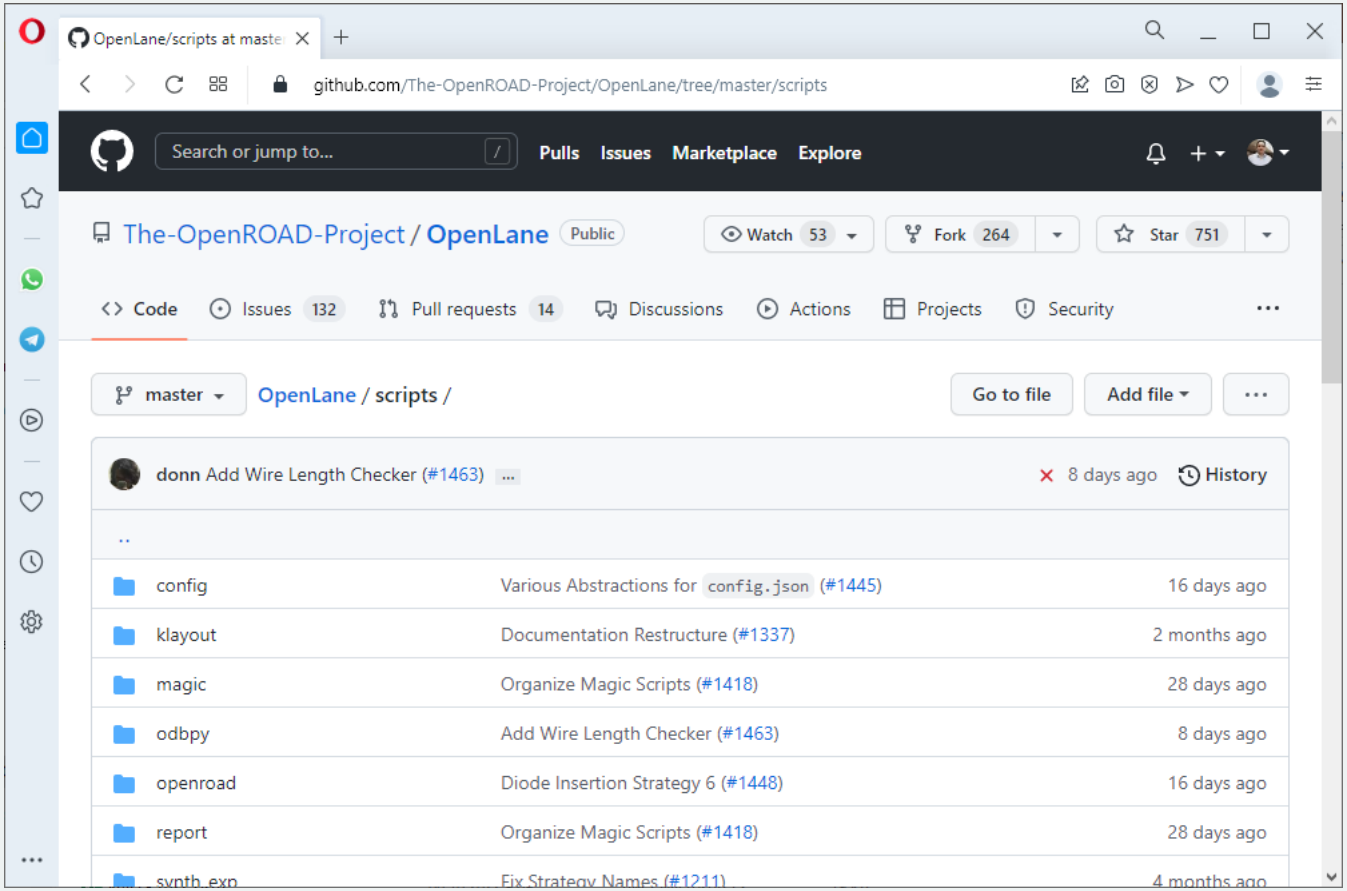
```
def add_and_sub(a, b):
        plus_val       = a + b
        minus_val      = a - b
        return (plus_val, minus_val)

x = add_and_sub(4, 5)

print(x)
```

```
print(x[0])
print(x[1])
```

```
def add_and_sub(a, b):
        plus_val       = a + b
        minus_val      = a - b
        return (plus_val, minus_val)

x, y = add_and_sub(4, 5)

print(x)
print(y)
```

Задание (1)

Есть проект OpenLane - открытый цифровой маршрут проектирования ИС (URL: https://github.com/The-OpenROAD-Project/OpenLane)
В его составе помимо ПО, выполняющего задачи проектирования, есть ряд скриптов, которые должны обрабатывать результаты моделирования (папка scripts)



Задание (2)

Один из таких скриптов – скрипт extract_antenna_violators.py, код которого приведён на слайде слева.

```
import re
import click

@click.command()
@click.option("-o", "--output", required=True, help="Output file to store results.")
@click.argument("report", nargs=1)
def extract_antenna_violators(output, report):

    pattern = re.compile(r"\s*([\S+]+)\s*\([\S+]+\)\s*[\S+]+")

    vios_list = []
    current_net = ""
    printed = False

    with open(report, "r") as f:
        for line in f:
            m = pattern.match(line)
            if m is not None:
                current_net = m.group(1)
                printed = False

            if "*" in line and not printed:
                print(current_net)
                vios_list.append(current_net + " ")
                printed = True

    with open(output, "w") as f:
        f.write("\n".join(vios_list))

if __name__ == "__main__":
    extract_antenna_violators()
```

Задание (3)

Задача этого скрипта – анализировать файл отчёта программы, проверяющей схему на наличие нарушений антенных правил (URL: http://www.vlsi-expert.com/2008/07/antenna-effects.html). Пример участка файла показан на слайде.

```
OpenROAD e036ecfaca4bc0efe88a54085efcf0f562c48a6b
This program is licensed under the BSD-3 license. See the LICENSE file for details.
Components of this program may be licensed under more restrictive licenses which must be honored.
[INFO ODB-0222] Reading LEF file: /openlane/designs/s44/runs/RUN_2022.08.19_10.16.09/tmp/merged.nom.lef
...

[INFO ODB-0223]     Created 13 technology layers
[INFO ODB-0224]     Created 25 technology vias
[INFO ODB-0225]     Created 441 library cells
[INFO ODB-0226] Finished LEF file:  /openlane/designs/s44/runs/RUN_2022.08.19_10.16.09/tmp/merged.nom.lef
[INFO ODB-0127] Reading DEF file: /openlane/designs/s44/runs/RUN_2022.08.19_10.16.09/results/routing/lut_s44.def
[INFO ODB-0128] Design: lut_s44
[INFO ODB-0130]     Created 28 pins.
[INFO ODB-0131]     Created 3902 components and 15064 component-terminals.
[INFO ODB-0132]     Created 2 special nets and 14600 connections.
[INFO ODB-0133]     Created 168 nets and 464 connections.
[INFO ODB-0134] Finished DEF file: /openlane/designs/s44/runs/RUN_2022.08.19_10.16.09/results/routing/lut_s44.def
Net net5
  _102_/B_N (sky130_fd_sc_hd__or2b_1)
    met2
    PAR:    8.64  Ratio:    0.00 (Area)
    PAR:   43.93  Ratio: 2778.20 (S.Area)
    CAR:  173.36  Ratio:    0.00 (C.Area)
    CAR:  870.17  Ratio:    0.00 (C.S.Area)

    met1
    PAR:  164.60  Ratio:    0.00 (Area)
    PAR:  826.11* Ratio:  400.00 (S.Area)
    CAR:  164.71  Ratio:    0.00 (C.Area)
    CAR:  826.25  Ratio:    0.00 (C.S.Area)
```

Нарушения антенных правил показаны записями, в которых около чисел стоит знак «*».
Эта запись означает, что ожидалось значение длины металлизации 400, а получено 826.11 (более, чем в 2 раза превышено допустимое значение)

Задание (4)

К сожалению, скрипт не работает как нужно (факт на сентябрь 2022 года), он не выдаёт информацию о том, какие нарушения встретились.

Ваша задача – исправить скрипт так, чтобы он выдал сообщения в файл отчёта в виде:

шаг 1 – просто выводил перечень имён инстансов элементов и пинов (для примера отчёта с предыдущего слайда пример результата показан ниже):

```
_102_/B_N
```

Эта запись значит, что для инстанса элемента с именем _102_ есть нарушение на пине этого элемента с именем B_N.

шаг 2 – помимо этого нужно выводить ещё и на каком металле это произошло:

```
_102_/B_N:met1
```

шаг 3 – помимо этого нужно выводить ещё и во сколько раз превышено допустимое значение:

```
_102_/B_N:met1 2
```

Задание (5)

Ссылки на файлы:

- скачать скрипт, который нужно поправить
- скачать первый тестовый пример файла отчёта для анализа
- скачать второй тестовый пример файла отчёта для анализа