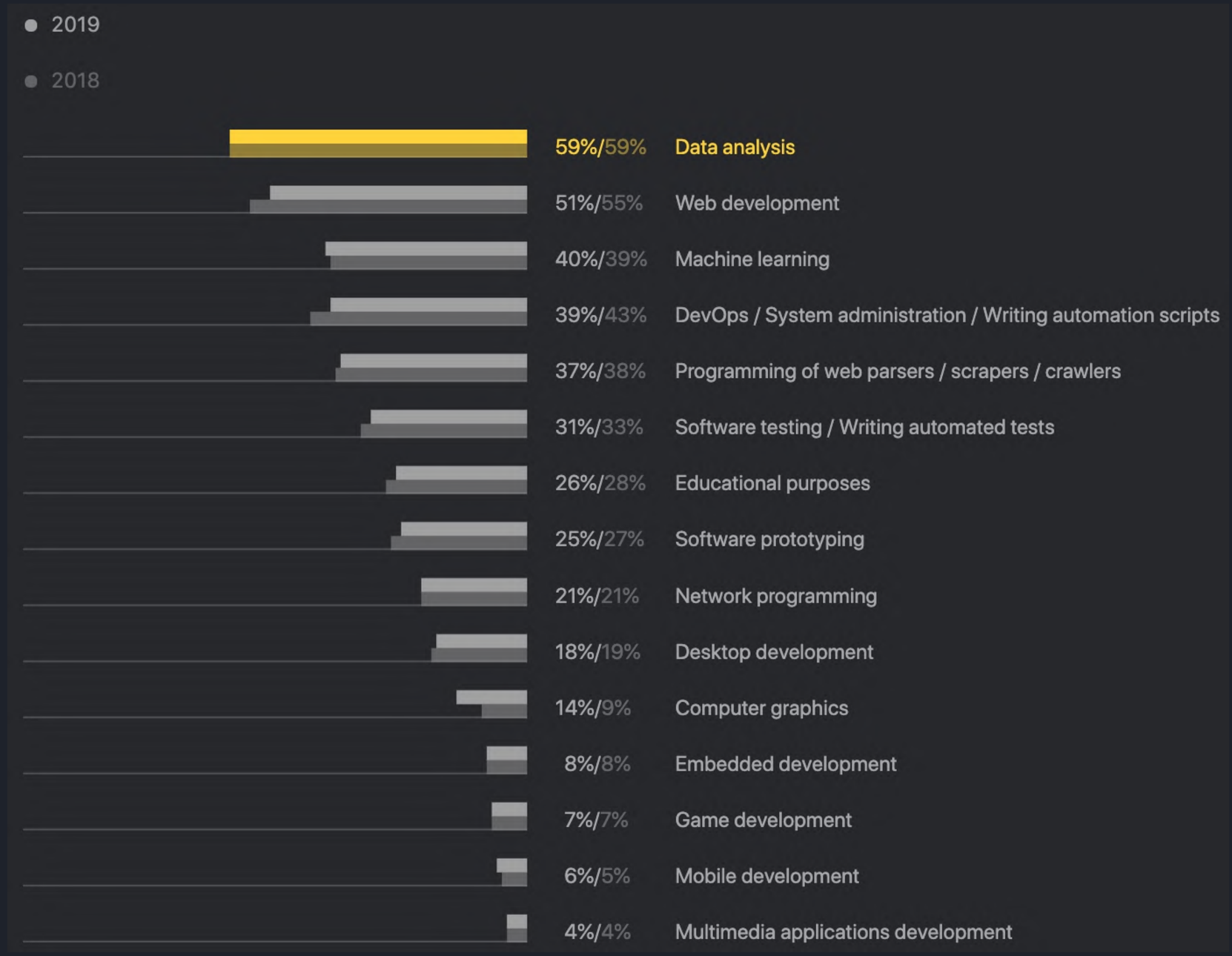
Python

Why Python?



Типы данных

- **str()** - строка
- **int()** - целое число
- **float()** - рациональное число (с точкой)
- **bool()** - булево значение True or False

Пример перевода одного типа в другой:

```
a = '21' # это строка
a = int(a) # перезапись переменной a, которая теперь содержит целое число а не строку
a = float(a) # перезапись переменной a с типом данных "число с точкой"
a = 21.0
a = str(a) # перезапись переменной на тип данных "строка"
```

Strings

```
a = 'Fgdh_8774'
b = 'JD_1989'
c = input()
print(a, b, c)
```

input() - Функция, которая позволяет осуществить ввод строки с клавиатуры при выполнении программы.

print() - Функция, которая выводит на печать то, что подается внутрь ее.

a, b, c - переменные. Они хранят в себе данные любых типов, в том числе строки, числа др.

Integer and float - просто число

```
a = 12
b = 3.4
c = a+b
print(c)
```

В отличие от строк, над числами можно совершать Арифметические операции:

- **Сумма** **+**
- **Разность** **-**
- **Умножение** *****
- **Деление** **/**
- **Возведение в степень** ******
- **Целочисленное деление** **//**
- **Остаток от деления** **%**

</>

Str()

Как создать и вывести на экран строку

```
a = 'jhkjf'
print(a)
b = input()
print(b)
```

a, b - переменные. Они хранят в себе данные, в данном случае строку.

input() - функция, которая попросит ввести строку с клавиатуры при выполнении программы

print() - функция, которая отображает те или иные данные. На вход принимает строку, число, либо переменную

Основные методы для строк (функции)

replace() - замена данного символа на другой

upper() - все буквы сделать заглавными

lower() - все буквы сделать строчными

capitalize() - первую букву сделать заглавной

Данные функции не изменяют строку.
Чтобы изменения вступили в силу строку необходимо пересохранить в текущую или другую переменную

```
a = 'abcd_12'
a.upper()
>>> ABCD_12 # но исходная строка не изменена
a = a.upper() # строка пересохранена в текущую переменную
print(a)
>>> ABCD_12


b = 'Саша'
b = b.replace('С', 'М') #перезаписали строку заменив символ 'С' на 'М'.
print(b)
>>> Маша
```

len() & index()

```
len() # возвращает длину строку
a = 'abcd12'
len(a)
>>> 6


index() # Возвращает порядковый номер конкретного элемента в строке (нумерация начинается с 0)
a.index('1')
>>> 4  # символ '1' находится на позиции с номером 4
```

Slicing

str[start: stop: step]

Для некоторой строки str можно вывести ее часть задав диапазон:

start & stop - левая и правая границы. Левая - входящая, левая - не входящая во множество

a = 'abcd1234'

a[1:3]

>>> 'bc' # вывести элемент с первого(второго по человечески) по третий не включительно (1й и 2й)

a[:3]

>>> 'abc' # вывести элементы сначала по 3й не включительно

a[3:]

>>> 'd1234' # вывести элементы с 3го по последний.

```
a = 'abcd1234'


a[3]
>>> 'd' # вывести 3й элемент
a[-1]
>>> '4' # вывести последний элемент
a[-2]
>>> '3' # вывести второй с конца элемент
a[-3: -1]
>>> '23' #  вывести элементы с третьего с конца по последний не включительно.
a[: -2]
>>> 'abcd12' # вывести элементы сначала по предпоследний не включительно
a[-3:]
>>> '234' # вывести элементы с третьего с конца до конца
```

str[start: stop: step]

step - шаг

step = 1 - выводить элементы подряд

step = 2 - выводить элементы через один

step = 3 - выводить элементы через 2

step = -1 - выводить элементы подряд но в обратном порядке

a[::-1] Вывести все элементы но в обратном порядке

a[::-2] вывести элементы в обратном порядке через 1

```
a = 'abcd1234'


a[1::2]
>>> 'bd24'
a[1::-1]
>>> '4321dcb'
a[::-1]
>>> '4321dcba'
a[-3: -1: -1]
>>> '32'
```

Логические операции

</>

```
a, b = 'a', 'b'
a == b    # Проверка на равенство элементов
>>> False
a != b  #  Проверка на НЕравенство элементов
>>> True
# значение логической операции можно сохранить в переменную, тип которой будет bool.
res = a != b
res
>>> True

# Проверка вхождения элемента в строку

x = 'abcd1234'
'a' in x
>>> True
'm' in x
>>> False
```

```
x = 'шел_мужик_попукивал_палочой_постукивал'

if len(x) == 10:
  print('Респект')
else:
  print('Иди нахрен!')
```

условие

тело оператора

Условный оператор для чисел

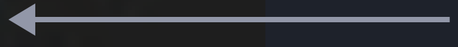
```
x = 25
y = 34


if x > y:
  z = x - y
elif x < y:
  z = y - x
else:
  z = x*y

print(z)
```

Усл_1

Усл_2



Усл_3 - если не выполняются предыдущие 2 условия

Логические операции **and, or, not**

- **Логические операции -** применяют к операндам типа **boolean,** они возвращают значение типа **boolean**

  – бинарные операции **and, or** и **xor**, унарная **not**,
- **Логическое выражение** – имеют тип **boolean**.

Not - логическое отрицание
And – логическое умножение
Or - логическое сложение
Xor – или-не, строгое или.

a	b	a and b	a or b	a xor b	not a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Цикл for

```
for счетчик from … to:
    тело цикла
```

Цикл принимает на вход итерируемый объект:

- числовой диапазон
- строку
- список
- любой другой массив данных

Каждый шаг цикла называется ИТЕРАЦИЯ.

При каждой итерации в теле цикла осуществляется какое-либо действие над элементом итерируемого объекта

Синтаксис цикла for:

for i in range(start, stopn step):

i - объект итерируемого объекта

range() - функция, которая возвращает набор чисел в диапозоне [start : stop) с шагом step

ПРИМЕРЫ:

```
for i in range(3):
        print(i)
>>> 0 1 2

for i in range(4, 1, -1):
        print(i)
>>> 4 3 2

a = 'abcd1234'
for i in a:
        print(i)
>>> a b c d 1 2 3 4
```