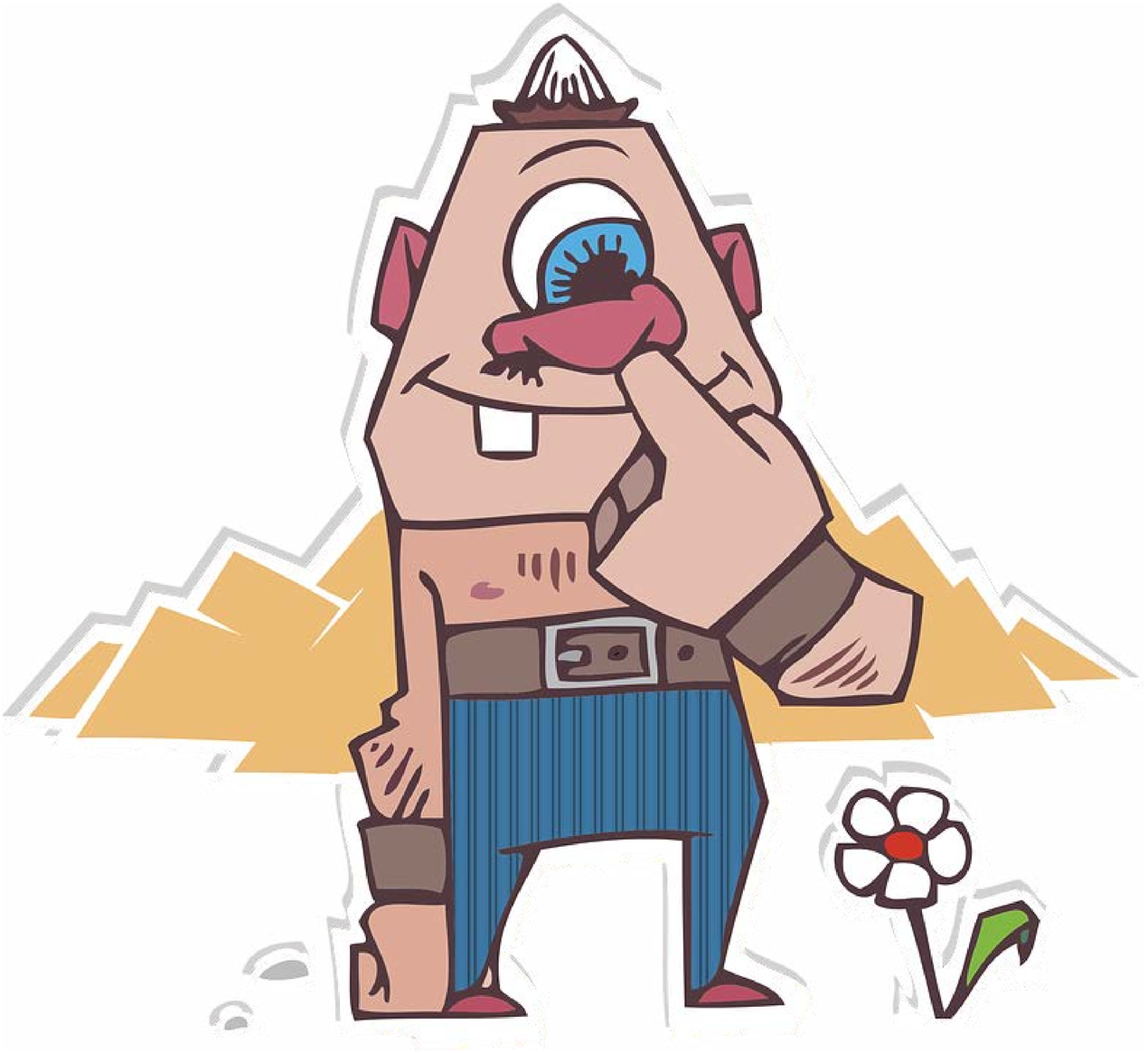
User Defined Functions


blue
leave
pole
moment
sing
hine
delirious
clhim
whisper
time
size
may
spring
stare
through
those
see
behind
your
picture

```
access_log.select(f.split("user_agent", " ").alias("words"))\
          .select(f.explode("words").alias("word"))\
          .groupBy("word")\
          .agg(f.count("*").alias("count"))\
          .orderBy(f.col("count").desc())\
          .limit(10).toPandas()
```

	word	count
0	Mozilla/5.0	75565
1	like	63791
2	Gecko)	58926
3	(KHTML,	58551
4	NT	50439
5	AppleWebKit/537.36	48648
6	Safari/537.36	48606
7	(Windows	37942
8	CLR	32140
9	.NET	31648



UDF

User Defined Function

In this video you will learn

- how to create UDF for dataframe api
- how to use UDF in sql queries
- how to build word count and parse user agent

len

```
len_udf = f.udf(len)
```

```
access_log.select(len_udf("user_agent").alias("len"))\
          .limit(3).toPandas()
```

	len
0	120
1	88
2	153

```
access_log.select(len_udf("user_agent").alias("len"))\
          .printSchema()
```

```
root
 |-- len: string (nullable = true)
```

String

Boolean

Date

Timestamp

Double

Float

Integer

Long

Short

Array

Map

Struct

String

Boolean

Date

Timestamp

Double

Float

Integer

Long

Short

Array

Map

Struct

String

Boolean

Date

Timestamp

Double

Float

Integer

Long

Short

Array

Map

Struct

```
len_udf = f.udf(len, t.IntegerType())
```

```
access_log.select(len_udf("user_agent").alias("len"))\
          .printSchema()
```

```
root
 |-- len: integer (nullable = true)
```

```
def parse_user_agent(user_agent):
    user_agent = user_agent.split()
    return user_agent
```

```
parse_user_agent_udf = f.udf(parse_user_agent,
                             ???????)
```

```
parse_user_agent_udf = f.udf(parse_user_agent,
                             t.ArrayType(t.StringType()))
```

```
access_log.select(parse_user_agent_udf("user_agent").alias("words"))\
          .select(f.explode("words").alias("word"))\
          .groupBy("word")\
          .agg(f.count("*").alias("count"))\
```

	word	count
0	Mozilla/5.0	75565
1	like	63791
2	Gecko)	58926
3	(KHTML,	58551
4	NT	50439
5	AppleWebKit/537.36	48648
6	Safari/537.36	48606
7	(Windows	37942

```
def parse_user_agent(user_agent):
    user_agent = re.sub("/?[\d_.]+", "", user_agent)
    user_agent = user_agent.split()
    return user_agent

parse_user_agent_udf = f.udf(parse_user_agent, t.ArrayType(t.StringType()))
```

```
access_log.select(parse_user_agent_udf("user_agent").alias("words"))\
          .select(f.explode("words").alias("word"))\
          .groupBy("word")\
          .agg(f.count("*").alias("count"))\
          .orderBy(f.col("count").desc())\
          .limit(10).toPandas()
```

	word	count
0	;	104286
1	Mozilla	85737
2	like	63791
3	Gecko)	58926
4	AppleWebKit	58783
5	(KHML,	58551
6	Safari	58273
7	NT	50439
8	Chrome	49870
9	(Windows	37942

```
def parse_user_agent(user_agent):
    user_agent = re.sub("/?[\d_.]+", "", user_agent)
   user_agent = re.sub("[;\(\):,]", "", user_agent)
    user_agent = user_agent.split()
    return user_agent

parse_user_agent_udf = f.udf(parse_user_agent, t.ArrayType(t.StringType()))
```

```
access_log.select(parse_user_agent_udf("user_agent").alias("words"))\
          .select(f.explode("words").alias("word"))\
          .groupBy("word")\
          .agg(f.count("*").alias("count"))\
          .orderBy(f.col("count").desc())\
          .limit(10).toPandas()
```

	word	count
0	Mozilla	86410
1	Gecko	75283
2	like	63874
3	KHTML	59008
4	AppleWebKit	58783
5	Safari	58365
6	Windows	52999

```
access_log.select(parse_user_agent_udf("user_agent").alias("words"))\
          .select(f.explode("words").alias("word"))\
          .groupBy("word")\
          .agg(f.count("*").alias("count"))\
          .orderBy(f.col("count").desc())\
          .limit(10).toPandas()
```

	word	count
0	Mozilla	86410
1	Gecko	75283
2	like	63874
3	KHTML	59008
4	AppleWebKit	58783
5	Safari	58365
6	Windows	52999

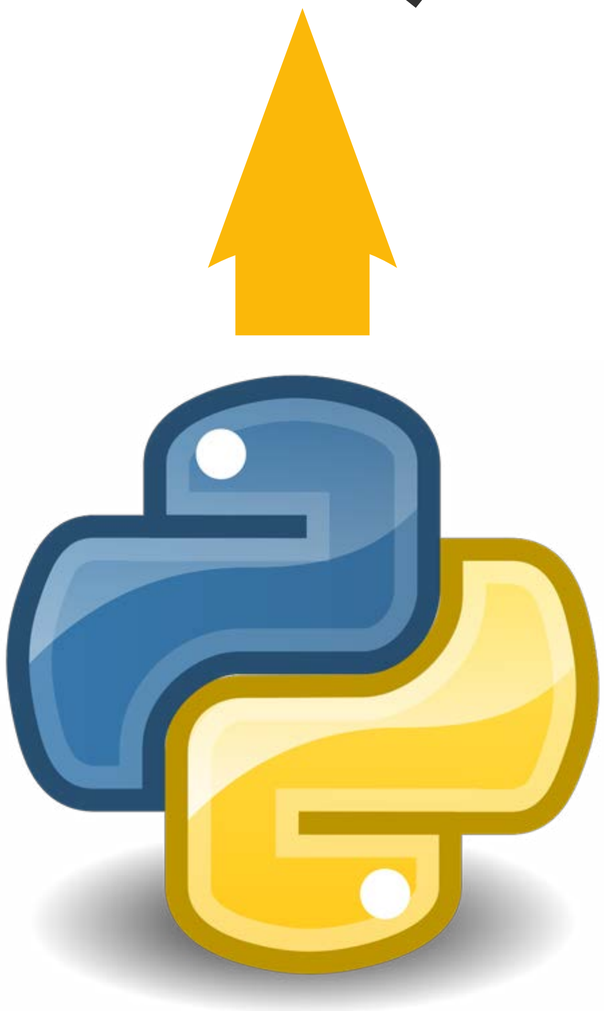
```
def parse_user_agent(user_agent):
    user_agent = re.sub("/?[\d_.]+", "", user_agent)
    user_agent = re.sub("[;\(\):,]", "", user_agent)
    user_agent = user_agent.lower()
    user_agent = user_agent.split()
    return user_agent

parse_user_agent_udf = f.udf(parse_user_agent, t.ArrayType(t.StringType()))
```

```
access_log.select(parse_user_agent_udf("user_agent").alias("words"))\
          .select(f.explode("words").alias("word"))\
          .groupBy("word")\
          .agg(f.count("*").alias("count"))\
          .orderBy(f.col("count").desc())\
          .limit(10).toPandas()
```

	word	count
0	mozilla	86441
1	gecko	75314
2	like	63905
3	khtml	59008
4	applewebkit	58814
5	safari	58396

SQL



```
spark_session.udf.register("parse_user_agent",
                           parse_user_agent,
                           t.ArrayType(t.StringType()))
```

```
spark_session.udf.register("parse_user_agent",
                           parse_user_agent,
                           t.ArrayType(t.StringType()))
```

```
spark_session.sql("""
    select parse_user_agent(user_agent) as user_agent
    from web.access_log
""").limit(10).toPandas()
```

	user_agent
0	[macintosh, intel, applewebkit, gecko, chrome,...
1	[windows, gecko, firefox, opera]
2	[compatible, msie, windows, infopath, netc, nete]
3	[linux, android, nb-no, samsung, gt-i, build/k...
4	[linux, android, nb-no, samsung, gt-i, build/k...
5	[macintosh, intel, applewebkit, gecko, version...
6	[windows, applewebkit, gecko, chrome, safari]
7	[macintosh, intel, applewebkit, gecko, chrome,...
8	[windows, windows, en-us, gecko, firefox, system]
9	[macintosh, de-de, applewebkit, gecko, version...

```
spark_session.sql("""
    select explode(parse_user_agent(user_agent)) as word
    from web.access_log
""").limit(10).toPandas()
```

	user_agent
0	macintosh
1	intel
2	applewebkit
3	gecko
4	chrome
5	safari
6	windows
7	gecko
8	firefox
9	opera

```
spark_session.sql("""
  select word, count(*) as cnt
  from (
    select explode(parse_user_agent(user_agent)) as word
    from web.access_log
  ) s
  group by word
""").limit(10).toPandas()
```

	word	cnt
0	foxy	2
1	wuid=cbfabcabdfccabdb	4
2	xoom	68
3	sleipnir	25
4	fi-fi	16
5	hardy	27
6	mddsjs	14
7	wuid=cbbacdcdda	9
8	buildb	67
9	seznam	23

```
spark_session.sql("""
  select word, count(*) as cnt
  from (
    select explode(parse_user_agent(user_agent)) as word
    from web.access_log
  ) s
  group by word
  order by cnt desc
""").limit(10).toPandas()
```

	word	cnt
0	gecko	75314
1	applewebkit	58814
2	safari	58396
3	windows	52999
4	chrome	49901
5	linux	18600
6	firefox	13721
7	android	13444
8	macintosh	13383
9	intel	13262

Browser?

Device?

user agents

https://pypi.python.org/pypi/user-agents

```
import user_agents as ua
```

```
import user_agents as ua
```

```
user_agent = ua.parse(u’Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)...’)
```

```
import user_agents as ua
```

```
user_agent = ua.parse(u’Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)...’)
```

```
print user_agent.browser.family
print user_agent.os.family
print user_agent.device.family
```

```
Other
Mac OS X
Other
```

```
get_browser_udf = f.udf(lambda x: ua.parse(x).browser.family)
get_os_udf      = f.udf(lambda x: ua.parse(x).os.family)
get_device_udf  = f.udf(lambda x: ua.parse(x).device.family)
```

```
get_browser_udf = f.udf(lambda x: ua.parse(x).browser.family)
get_os_udf      = f.udf(lambda x: ua.parse(x).os.family)
get_device_udf  = f.udf(lambda x: ua.parse(x).device.family)
```

```
access_log.select("user_agent",
                  get_browser_udf("user_agent").alias("browser"),
                  get_os_udf("user_agent").alias("os"),
                  get_device_udf("user_agent").alias("device"),
                 )\
          .limit(5).toPandas()
```

	user_agent	browser	os	device
0	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)...	Chrome	Mac OS X	Other
1	Mozilla/5.0 (Windows NT 5.1; U; de; rv:1.9.1.6...	Opera	Windows XP	Other
2	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT ...	IE	Windows XP	Other
3	Mozilla/5.0 (Linux; Android 4.4.2; nb-no; SAMS...	Chrome Mobile	Android	Samsung GT-I9505
4	Mozilla/5.0 (Linux; Android 4.4.2; nb-no; SAMS...	Chrome Mobile	Android	Samsung GT-I9505

Within this lesson you have

- trained to write UDF for DataFrame API
- learned how to use UDF in SQL queries
- solved practical problems with log analysis