Модуль os

@python2day

Модуль `os` позволяет работать с файловой системой, с окружением, управлять процессами.

Импорт модуля:

```
In [1]: import os
```

os.environ

os.environ возвращает словарь с переменными окружения и их значениями. Можно использовать синтаксис обращения по ключу через квадратные скобки, если переменная окружения точно существует (если переменной нет возникнет исключение).

```
In [2]: os.environ["HOME"]
Out[2]: '/home/nata'

In [3]: os.environ["TOKEN"]
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [3], in <cell line: 1>()
----> 1 os.environ["TOKEN"]

File ~/venv/pyneng-py3-8-0/lib/python3.8/os.py:673, in _Environ.__getitem__(self, key)
    670     value = self._data[self.encodekey(key)]
    671 except KeyError:
    672     # raise KeyError with the original key value
--> 673     raise KeyError(key) from None
    674 return self.decodevalue(value)

KeyError: 'TOKEN'
```

Или использовать get, тогда при отсутствии переменной окружения, возвращается None:

```
In [3]: os.environ.get("HOME")
Out[3]: '/home/nata'

In [4]: os.environ.get("TOKEN")
```

> Примечание
>
> Технически os.environ возвращает объект типа mapping, но на данном этапе проще считать его словарем.

Переменные окружения считываются в момент импорта модуля os, если какие-то переменные были добавлены во время работы скрипта, они не будут доступны через `os.environ`.

os.mkdir

os.mkdir позволяет создать каталог:

```
In [2]: os.mkdir('test')

In [3]: ls -ls
total 0
0 drwxr-xr-x  2 nata  nata  68 Jan 23 18:58 test/
```

os.listdir

Функция listdir возвращает список файлов и подкаталогов в указанном каталоге. Порядок файлов в списке произвольный, если нужно получить их в порядке сортировки имен, можно использовать sorted.

```
In [2]: os.listdir("09_functions")
Out[2]:
['test_task_9_4.py',
 'task_9_2a.py',
 'task_9_1a.py',
 'test_task_9_2.py',
 'task_9_3a.py',
 'test_task_9_3a.py',
 'task_9_3.py',
 'test_task_9_3.py',
 'config_sw2.txt',
 'test_task_9_2a.py',
 'config_sw1.txt',
 'test_task_9_1a.py',
 'test_task_9_1.py',
 'task_9_4.py',
 'task_9_1.py',
 'config_r1.txt',
 'task_9_2.py']

In [3]: sorted(os.listdir("09_functions"))
Out[3]:
['config_r1.txt',
 'config_sw1.txt',
 'config_sw2.txt',
 'task_9_1.py',
 'task_9_1a.py',
 'task_9_2.py',
 'task_9_2a.py',
 'task_9_3.py',
 'task_9_3a.py',
 'task_9_4.py',
 'test_task_9_1.py',
 'test_task_9_1a.py',
 'test_task_9_2.py',
 'test_task_9_2a.py',
 'test_task_9_3.py',
 'test_task_9_3a.py',
 'test_task_9_4.py']
```

Текущий каталог можно указать так `"."` или вызывать listdir без аргументов:

```
In [7]: os.listdir('.')
Out[7]: ['cover3.png', 'dir2', 'dir3', 'README.txt', 'test']

In [7]: os.listdir()
Out[7]: ['cover3.png', 'dir2', 'dir3', 'README.txt', 'test']
```

os.path

Разные операционные системы (ОС) имеют разные соглашения об именах путей, поэтому в стандартной библиотеке есть несколько версий модуля `os.path`. Модуль os автоматически подгружает нужную часть для работы с текущей ОС. Например, при запуске одних и тех же функций модуля os на Windows и Linux, разделителем пути будут считаться разные значения.

При необходимости работы на Linux с путями Windows и наоборот, можно использовать модули `posixpath`, `ntpath` вместо `os.path`.

os.path.exists

Функция `os.path.exists` проверяет существует ли указанный путь и возвращает True, если пусть существует и False иначе:

```
In [5]: os.path.exists('test')
Out[5]: True

In [6]: if not os.path.exists('test'):
   ...:     os.mkdir('test')
   ...:
```

os.path.isdir, os.path.isfile

Функция `os.path.isdir` возвращает True, если путь ведет к каталогу и False иначе:

```
In [4]: os.path.isdir("09_functions")
Out[4]: True

In [5]: os.path.isdir("/home/nata/repos/pyneng-tasks/exercises/09_functions/")
Out[5]: True

In [6]: os.path.isdir("/home/nata/repos/pyneng-tasks/exercises/09_functions/task_9_1.py")
Out[6]: False

In [7]: os.path.isdir("09_functions/task_9_1.py")
Out[7]: False
```

Функция `os.path.isfile` возвращает True, если путь ведет к файлу и False иначе:

```
In [9]: os.path.isfile("09_functions/task_9_1.py")
Out[9]: True

In [10]: os.path.isfile("09_functions/")
Out[10]: False
```

С помощью проверок `os.path.isdir` и `os.path.isfile` и `os.listdir` можно получить списки файлов и каталогов (в примере для текущего каталога).

Список каталогов в текущем каталоге:

```
In [8]: dirs = [d for d in os.listdir('.') if os.path.isdir(d)]

In [9]: dirs
Out[9]: ['dir2', 'dir3', 'test']
```

Список файлов в текущем каталоге:

```
In [10]: files = [f for f in os.listdir('.') if os.path.isfile(f)]

In [11]: files
Out[11]: ['cover3.png', 'README.txt']
```

os.path.split

Функция os.path.split делает разделение пути на «основную часть» и конец пути по последнему `/` и возвращает кортеж из двух элементов. При этом для Windows автоматически будет использоваться обратный слеш.

Если в конце пути не слеша, разделение будет таким

```
In [6]: os.path.split("book/25_additional_info/README.md")
Out[6]: ('book/25_additional_info', 'README.md')

In [8]: os.path.split("book/25_additional_info")
Out[8]: ('book', '25_additional_info')
```

Если путь заканчивается на слеш, второй элемент кортежа будет пустой строкой:

```
In [7]: os.path.split("book/25_additional_info/")
Out[7]: ('book/25_additional_info', '')

In [9]: os.path.split("book/")
Out[9]: ('book', '')
```

Если в пути нет слеш, первый элемент кортежа будет пустой строкой:

```
In [30]: os.path.split("README.md")
Out[30]: ('', 'README.md')
```

os.path.abspath

Функция os.path.abspath возвращает абсолютный путь для указанного файла или каталога:

```
In [40]: os.path.abspath("09_functions")
Out[40]: '/home/nata/repos/pyneng-tasks/exercises/09_functions'

In [41]: os.path.abspath(".")
Out[41]: '/home/nata/repos/pyneng-tasks/exercises'
```