Python!

Lekcja 2 – operacje wejścia/wyjścia

Plan

- Stringi - powtórzenie
- Wypisywanie rzeczy na ekran
- Przyjmowanie danych z klawiatury
- Formatowanie danych
- Operacje na plikach

Stringi - powtórzenie

- Sringi – ciągi znaków

- W Python3 wszystkie stringi są w standardzie Unicode

- W Python2 trzeba podać kodowanie ręcznie na początku pliku:

#-*- coding: <kodowanie> -*-, np:
#-*- coding: utf-8 -*-
#-*- coding: windows-1250 -*-

Można też:
from __future__ import unicode_literals

https://pl.wikibooks.org/wiki/Zanurkuj_w_Pythonie/Kodowanie_znak%C3%B3w
http://python-future.org/compatible_idioms.html#unicode-text-string-literals

Wypisywanie rzeczy na ekran

Funkcja *print* – podstawowe narzędzie do wypisywania tekstu na ekran

Zgodnie z dokumentacją Pythona3:

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

sys.stdout – standardowe wyjście (ekran)

sys.stderr – standardowe wyjście błędów (z reguły też ekran)

2>/dev/null - Znajome?

Żeby w Python2 korzystać z tego dobrodziejstwa:
from __future__ import print_function

https://docs.python.org/3/library/functions.html#print

Wypisywanie rzeczy na ekran

Proste przykłady:

```
>>> print("Hello World!")
Hello World!
>>> print("Hello World!\n")
Hello World!

>>>
```

https://docs.python.org/3/library/functions.html#print

Przyjmowanie danych z klawiatury

Funkcja *raw_input* – oczekuje aż użytkownik dostarczy dane i konwertuje je na string

```
>>>raw_input("Podaj cosik: ")
Podaj cosik: 12
'12'
>>> a = raw_input("Znowu podaj cosik: ")
Znowu podaj cosik: 23
>>> a
'23‘
```

Istnieje też metoda *input,* która wykrywa typ wpisywanych danych

Formatowanie danych

Funkcja *format* – formatuje dany string w zadany sposób.
Włącznie z podstawianiem danych w odpowiednie miejsca.

```
>>> "Stefan ma {} lata".format(a)
'Stefan ma 23 lata‘
>>> "Stefan ma {:.3f} litrów wódki".format(0.070000)
'Stefan ma 0.070 litrów wódki‘
>>> "Stefan ma {} lata {:.3f} litrów wódki".format(23, 0.070000)
'Stefan ma 23 lata 0.070 litrów wódki'
>>> "Stefan ma {1} lata {0:.3f} litrów wódki".format(23, 0.070000)
'Stefan ma 0.07 lata 23.000 litrów wódki'
```

Uwaga na wersje! W 2.6 trzeba podawać pozycje.

https://docs.python.org/3/library/string.html#formatspec
https://pyformat.info/

Formatowanie danych - bonus

Metoda *join* – łączy listę stringów w jeden

```
>>> " ".join(['a', 'b', 'c'])
'a b c'
>>> "\n".join(['a', 'b', 'c'])
'a\nb\nc'
>>> print("\n".join(['a', 'b', 'c']))
a
b
c
```

https://docs.python.org/3/library/string.html#formatspec
https://pyformat.info/

Operacje na plikach

Funkcja *open* – funkcja otwierająca pliki

open(*file*, *mode='r'*, *buffering=-1*, *encoding=None*, *errors=None*, *newline=None*, *closefd=True*, *opener=None*)

file – ścieżka do pliku

mode – tryb, w jakim zostanie otwarty plik

- r – odczyt
- w – zapis, przy uprzednim wyczeszeniu zawartości
- a – dopisanie wartości do końca pliku
- b – tryb binarny (domyślnie tekstywy – t)
- + – aktualizacja pliku (tryby r i w)

Operacje na plikach

```
>>> f = open(‘test.txt’, ‘w+’)
>>> f.write(‘jakieś znaki’)
>>> f.close()
```

Można też użyć słowa kluczowego *with* – otwiera blok kodu (!) z użyciem magangera kontekstu. Zasadniczo zwiększa pewność, że blok kodu zostanie wykonany prawidłowo. W tym wypadku gwarantuje też zamknięcie pliku.

```
>>> with open('test.txt', ‘r') as f:
...     f.read()
```

Zagadka: co robi metoda readlines() ?

Dziękuję za uwagę!