

Values, Expressions, Variables, Types & Syntax.



PluralSight + CodeSchool
= 72 hours free access

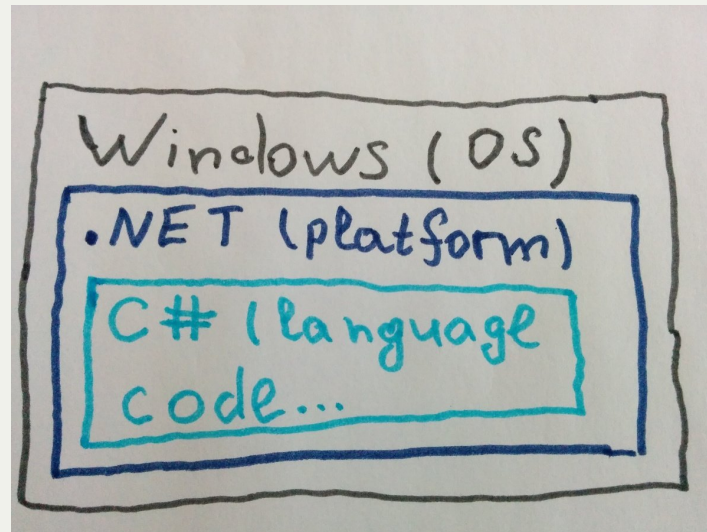
Учете на повече от 1
МЯСТО.



Какво е Windows?

Какво е .NET?

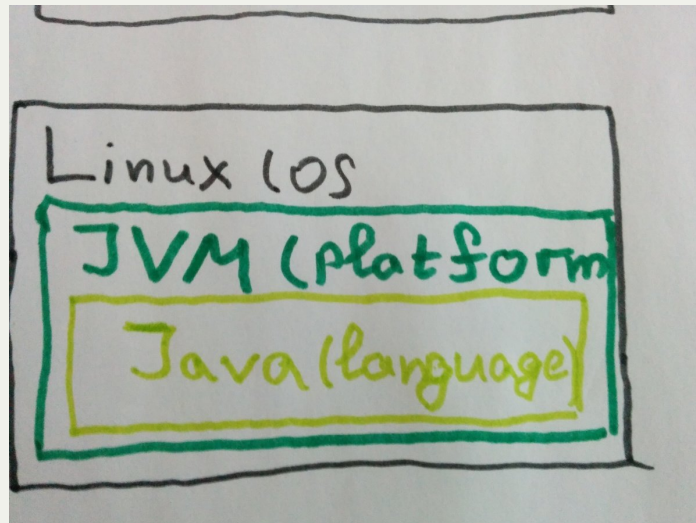
Какво е C#?



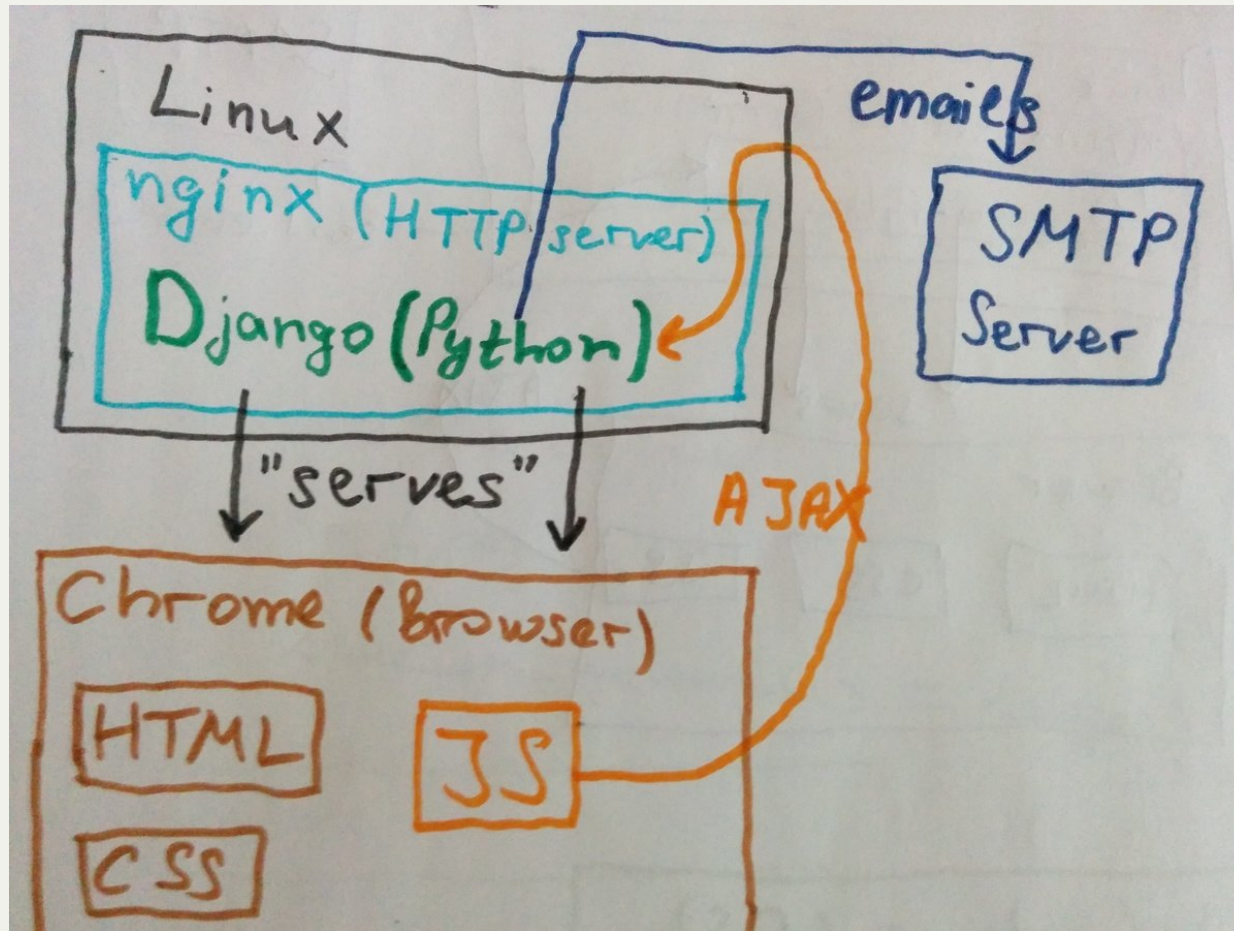
Какво е Linux?

Какво е JVM?

Какво е Java?



Как работи Odin?



Какво е текстов файл?

hello.py



Името на файла



Разширението на файла



Какво е **текстов файл**?

1. Файл, който е съставен от последователност от редове.
2. Файл, в който няма форматиране.
3. Често се нарича **"plain text"**.
4. **Може да го прочетете през текстов редактор.**



Така изглежда един .pdf:

```
1 %PDF-1.3
2 %Ааоаёёó ДАÆ
3 4 0 obj
4 << /Length 5 0 R /Filter /FlateDecode >>
5 stream
6 x^AYTÉNÄ0^P½ç+^^<94>%aq½ÄNre9Ä<89>H<96>8P^N(j^E^"AUÄÿ3vk7ç-<88>J\^°0^^ÜäYb<I^D5&8â<8c>»!K#,Tq<9f><8b>B*©QH^NMët<88>
>[¼ ^?>^Shfà~Ê<85>(<8c>ö<92>æUE^S:<93>B<9b>0`ÖD;Á®³NFkc/^EBÁ×øë«dUä^Q#p?ÉE<84>¼<9b>RD<94>p<88>ÖÄ6¼iø0<9a>^Vg^V^EÄL
ö<94>VQ<95><8c>Km <94><82>mN·VBÄ<8e>p<87>t'sáH»Aè^Ea/£d»7ÜääàðK°ó*p^MÖ 5Ê2ÜÄ^ä0ç&SÜøQP:ëÄÿ<93>év<85>Ä¼pó<92>JØ! [Ä
<82>W-ä:w5Læ5Tó^Z!®Ê^Yi³<87>y^C¶^]Ä·<85>ùç,^VÜÜ^P<97>^C%ùmó^E(^]6+f<98>4Ê^U_à^äð<92>z<91><8d>G<9e><8c>^Bi<9^Nô<8c>
<?>éeIäéß^A_6<87><95>ÿø}^N@Uð<8f>\<9b>Äð~ù<96>g4g¹I5»b¥!Ô ^LóJ^Dº^`Uÿè$'»<9b>á^'Y½½^?<<8c>1}ç~^S5]^K ÛXµH<8c>i_µ^B^
W^I<8d>ú^SÜ0^QÎ
7 endstream
8 endobj
9 5 0 obj
10 396
11 endobj
12 2 0 obj
13 << /Type /Page /Parent 3 0 R /Resources 6 0 R /Contents 4 0 R /MediaBox [0 0 720 525]
14 >>
15 endobj
16 6 0 obj
17 << /ProcSet [ /PDF /Text /ImageB /ImageC /ImageI /ColorSpace<< /Cs1 7 0 R
18 >> /Font << /TT2 9 0 R /TT3 10 0 R >> /XObject << /Im1 11 0 R >> >>
19 endobj
20 11 0 obj
21 << /Length 12 0 R /Type /XObject /Subtype /Image /Width 532 /Height 840 /Interpolate
22 true /ColorSpace 7 0 R /Intent /Perceptual /SMask 13 0 R /BitsPerComponent
23 8 /Filter /FlateDecode >>
24 stream
25 x^Aí<9d>?^mù}Ý$^Q^"R^CéR 0^XìÆ<90>0àÁ!<9d>^K¥^H^R*^T!lPcG<95>E
```

Какво е текстов файл?

1. Файл, който е съставен от последователност от редове
2. Файл, в който няма форматиране

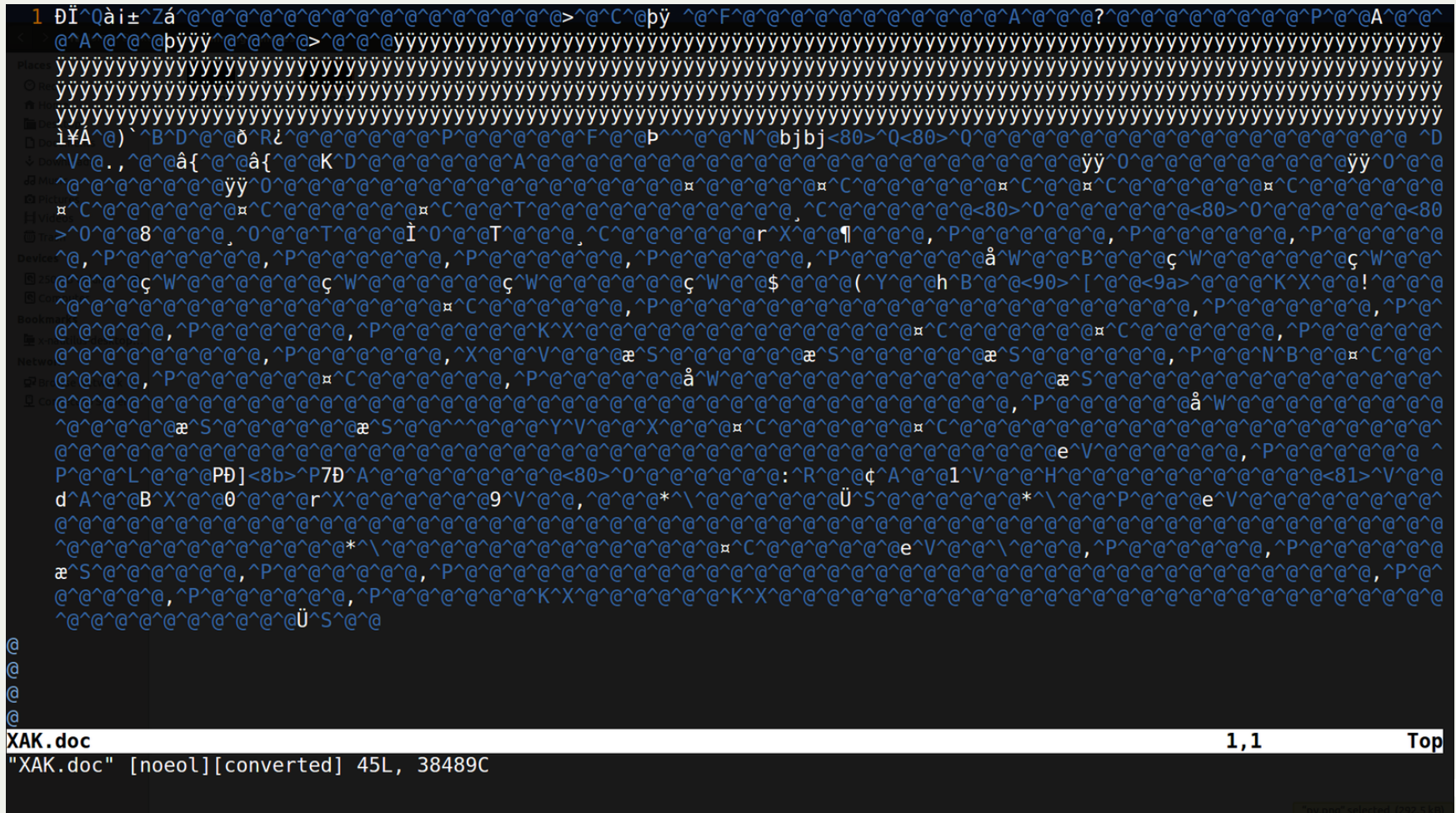
Introduction-Presentation.pdf

Така изглежда един .py:

```
135 functions = functions[2:]
136
137 for f in functions:
138     result = compose(result, f)
139
140 return result
141
142 def main():
143     dsl_filename = sys.argv[1]
144     rawContents = open(dsl_filename).read()
145
146     contents = compose_all([clean_empty_lines, lines])(rawContents)
147     imports = filter_imports(contents)
148     testCases = filter_test_cases(contents)
149     testComment = contents[len(imports):][0].replace("'", "")
150
151     indexedTestCases = zip(range(0, len(testCases)), testCases)
152
153     testCases = list(map(test_line_to_test_case, indexedTestCases))
154
155     print(get_test_template().format(**{
156         "imports": unlines(imports),
157         "test_name": get_test_classname(dsl_filename),
158         "test_description": testComment,
159         "test_cases": unlines(testCases)
160     })))
161
162 if __name__ == '__main__':
163     main()
164
```

parse.py

Така изглежда един .doc:



Няколко важни неща:

1. Пишем код в **текстови файлове**! Не в Word.
2. **Разширението** на текстовите файлове подсказват за езика, на който са написани.
(* .py, * .cpp, * .java, * .hs и тн.)
3. Работим с **текстови редактори** (Sublime, Vim, Emacs etc.) или със **Среди за Разработка** (IDE – Visual Studio, Eclipse, PyCharm etc.)



Започваме! Няколко важни равенства:

1 програма = 1 или повече текстови файла.

1 файл = 0 реда или повече код.

1 ред код = 0 или повече символи (от
клавиатурата)



Програма от 1 файл

```
# single.py

print("My file is called: {}".format(__file__))
print("I am a single program from a single file")
```



Програма от 2 файла

```
# first.py  
  
print("I am the first file!")
```

```
# second.py  
  
import first  
  
print("I am the second file!")
```



Задача! Решете 3та
задача от week 0.



Всеки език за
програмиране си има
правила за това как се
пише. **Тези правила се
наричат синтаксис.**



Важни неща за синтаксиса:

- Всеки език има своите правила и тези правила са много строги.
- Всеки език има програма, която разбира синтаксиса и изпълнява написания код.
- Ако синтаксисът е грешен, не може да пуснете програмата.
- Не може да пуснете Python код през C++ програмата за разбиране.



Python е програмата,
която чете *.py файлове.

Изпълнява ги ред по ред. Ако има
синтактична грешка, ви я показва
в голям червен текст.



SyntaxError: EOL while scanning string literal

```
# syntax_error.py  
print("What am I missing?)
```



Може да изпълняваме
Python код и извън
файлове.

REPL:

>>>



```
Python 3.4.0 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.0 (default, Apr 11 2014, 13:05:11)
[GCC 4.8.2] on linux
Type "copyright", "credits" or "license()" for more information.
>>> |
```



radorado@radorado ~/code/Programming0-1/week0-Starting <master*>

\$ py

Python 3.4.0 (default, Apr 11 2014, 13:05:11)

[GCC 4.8.2] on linux

Type "help", "copyright", "credits" or "license()" for more information.

>>>

1 ↩



Read Evaluate Print Loop

>>>



В REPL може да пишем:

1. Values (Стойност): `>>> 5`
2. Изрази (Expressions): `>>> 5 + 5`
3. Променливи (Variables): `>>> a = 5`



В Python, най-простата единица код е една стойност (value):

1. `>>> 5`
2. `>>> 5.0`
3. `>>> "Python"`
4. `>>> True`
5. `>>> None`



На колко са равни следните изрази?

$$(1 + 2 * (3 - 5)) / 3 = ?$$

$$x^2 - 1 = 0$$

$$20\% * 100$$



Какво всъщност е + ?

$$4 + 5$$

Наричаме го **оператор**.

Той има нужда от лява и дясна
операнда.



Израз (Expression) =
Прости стойности (Values) +
оператори (Operators) в правилен
за езика синтаксис.

```
>>> 5 + 5  
10
```

```
>>> "Rado" + "Rado"  
"RadoRado"
```



Изразите се пресмятат
(Evaluate от REPL) докато
не получим проста
стойност (Value)!



Почивка! Глътнете малко ЧИСТ ВЪЗДУХ.

И питайте въпроси, ако има.



Решете задача 4 от
седмица 0!



Променливите дават име на стойност или израз.

Променливите съдържат стойността!

```
tip_percentage = 10
```

```
discount = 0.25  
total_sum = (100 + 200 + 300) * discount
```



Променливите служат за:

Даване на **име** на определена
стойност:

```
name = "Radoslav Georgiev"  
age = 24  
favourite_language = "Haskell"
```



Променливите служат за:

Променяне на вече зададена стойност:

```
name = "Radoslav Georgiev"  
name = "Ivaylo Bachvaroff"  
name = 20
```



Променливите служат за:

Променяне на вече зададена стойност, чрез стара стойност:

```
name = "Radoslav"  
name = name + " "  
name = name + "Georgiev"
```



Променливите могат да са равни и на израз:

```
>>> name = "Radoslav" + " " + "Georgiev"  
>>> name  
"Radoslav Georgiev"
```

Но както всеки израз, той се
пресмята до дадена стойност.



Решете 5та и 6та задача
от седмица 0!



Сгига толкова за това
упражнение!

