{iT}Factory

Intalnirea 1

Setup, Variabile, Tipuri de date

Sfaturi generale

- Să tratați cu seriozitate și profesionalism acest nou obiectiv.
- Cei care își ating obiectivele nu sunt întotdeauna cei mai smart, dar întotdeauna vor fi cei mai muncitori!.
- Alocă-ți timp pentru studiu. Rutina dă consistență. Consistența dă excelență.
- Să faceți tot posibilul să participați la toate sesiunile live.
- Să vă lăsați comentarii explicative în cod. Notițe pentru voi din viitor.
- Recomand să vizualizați înregistrarea. Să vă notați aspectele importante + întrebări pentru trainer pentru ora următoare.
- Să vă faceți temele și unde nu reușiți singuri, să întrebați pe grup. Trainerul va răspunde și vor beneficia și ceilalți cursanți de răspuns.
- Puteți chiar să faceți un grup doar de studenți și să vă întâlniți o dată pe săptămână să discutați temele împreună. Fiecare va veni cu o perspectivă nouă și în final toți vor avea de câștigat.
- În timpul orelor, să aveți curaj să puneți întrebări când ceva nu e clar.

Reguli curs

- Va exista un sheet de prezenta.
- In cadrul acestuia ne vom asuma si notiunile invatate. Nu trecem mai departe pana nu isi asuma toti noile concepte.
- Temele se vor adauga in Folderul grupei, veti face fiecare folder cu numele voastru. Veti primi feedback la aceste teme.
- Temele vor fi impartite in 2 categorii.
  - Obligatorii (se pot face doar cu notiunile invatate la clasa)
  - Optionale (acestea vor fi mai advanced si necesita poate extra research). Acest lucru ii va motiva si pe cei care au mai mult timp si le place sa se aventureze prin task-uri mai dificile.
- Va rog sa ma intrerupeti oricand aveti intrebari. Doar asa imi pot da seama unde trebuie sa mai insist cu explicatii/exemple.
- Va rog sa intrati cu 3 minute mai devreme in caz ca apar probleme tehnice. Astfel putem profita la maxim de cele 2 ore alocate
- Daca nu puteti intra, sau daca intarziati, anuntati trainerul pe grup

Obiective principale

Pana la final TOTI veti avea:

- Cunostinte solide despre bazele programarii in Python
- Cunostinte mai avansate si extrem de utile despre programarea bazata pe obiecte.
- Capacitatea sa identifice elemente si sa scrie test scripts cu ajutorul Selenium
- Un Proiect final de testare automata a aplicatiilor web.
  - Acesta va folosi tendintele actuale: metodologia Behavior Driven Development si Page Object Model Design pattern.
  - Va avea capacitatea sa genereze rapoarte HTML (‘living documentation’)
  - Veti sti de la A la Z acest framework, astfel ca veti avea capacitatea sa continuati sa il dezvoltati post curs acest proiect (pentru orice website doriti).
- Notiuni de baza despre API testing. (testarea backend - ce e in spate la un website).

Obiective secundare

Nu fac parte din curricula cursului LIVE dar va punem la dispozitie materiale extra ca sa aveti un avantaj la interviuri. Sfatul meu e sa va focusati pe ele doar dupa cursul live. Sa nu fiti overwhelmed de new info.

- **Cunostinte ale bazelor de date relationale - mySQL (Curs baze de date)**
- **Cunostinte teoretice despre testarea manuala - acces la o platforma mobila**
- **Capacitatea de a construi un mic brand personal (Curs Portofoliu Wordpress). Trebuie sa ai:**
    - **Website propriu prin care angajatorul sa te cunoasca pe tine si munca ta**
    - **CV european in eng / sau canva.com**
    - **Profil LinkedIn**
    - **Github public (un loc in cloud unde se pune codul scris de tine)**
    - **Veti primi feedback daca ne trimiteti un email cu ele la hello@itfactory.ro**

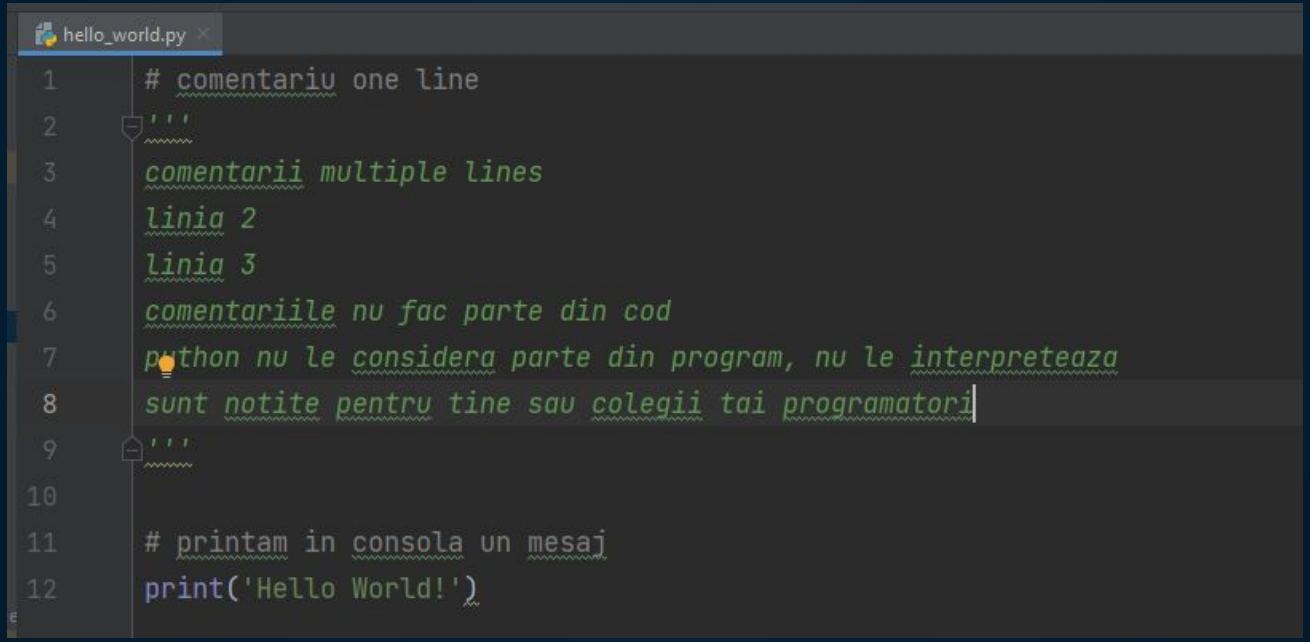
Obiective Intalnire 1

- Sa avem toti setup functional
- Sa intelegem cum functioneaza programarea si de ce e importanta pt. Automation
- Primul program Hello World
- Ce este un comentariu?
- Sa stim si sa putem explica altora ce e o variabila si de ce avem nevoie de ea
- Sa intelegem cele mai uzuale tipuri de date
- Sa intelegem ce este type casting si de ce e util
- Sa intelegem cum functioneaza print statement
- Sa stim cum luam date de la tastatura (user input)
- Ce e un assert si la ce e bun?
- Sa descoperim si sa aprofundam complexitatea unui string
  - Index
  - Length
  - Slicing
  - Metode ajutatoare

Principii de baza in programare

- A compila = a traduce din ‘human reading syntax’ in ‘machine language’
- Codul se interpreteaza secvential, linie cu linie, de sus in jos
- Machine language = binary code (cod binar) - combinatii diferite de 0 si 1
- Principiul seamana cu cel din codul morse. Pt 1 se transmite un impuls electric, pt 0 o pauza.
- 1 bit = memorie in care incape doar o singura valoare. 1 (true), 0 (false)
- 1 Byte = 8 biti. Numere intre 0 (00000000) si 255 (11111111)
- 1 Kilobyte = 1.024 bytes
- 1 Megabyte = 1.024 kilobytes (1.048.575 bytes)
- Terminal - zona in care trimitem instructiuni catre program (altele decat cod python)
  - Ex: ‘python --version’
  - Tot de aici putem instala librarii externe (ex: pip install selenium)
- Consola - zona in care primim output (raspuns vizual) de la programul rulat
- IDE - Integrated Development Environment - Pycharm. Este un editor de cod
- Venv - Virtual environment - zona care foloseste in mod izolat si securizat toate librariile externe

Hello World + Comentarii



Variabile

```
# am declarat si initializat variabile
marca_masina = 'Volvo'
model_masina = 'XC 60'
```

- O variabila este un container din memorie care stocheaza valori
- Va puteti imagina o cutiuta, pe care punem un label
- Variabilele au nume unic, ca sa poata fi identificate si folosite ulterior
- Variabila e creata in momentul in care ii atribuim o valoare
- Nu putem pune spatiu in numele unei variabile (my_var sau myVar)
- Variabilele incep cu litera mica dar pot contine cifre (user1) si simbolul _
- Variabilele sunt case sensitive (myvar=3 e diferita de myVar=5)
- Variabilele pot sa isi schimbe valoarea pe parcursul executiei programului (suprascriere)
    - Si chiar si tipul de date
- Putem atribui mai multe valori in one line, sau aceeasi valoare mai multor variabile

```
x, y, z = "Orange", "Banana", "Cherry"
a = b = c = "Apple"
```

Tipuri de date

- Datele salvate in variabile pot avea diferite tipuri
- Exista mai multe tipuri de date dar cele mai importante/folosite sunt
  - int - numar intreg
  - float - numar zecimal
  - bool - adevarat/fals
  - string - sir de caractere de la tastatura delimitate de ‘ ‘ sau “ “

- In intalnirea 3 vom discuta si despre colectii, care sunt tot tipuri de date (list, dict, set, tuple)

```
marca = 'Dacia' # string - sir de caractere
an_fabricatie = 1987 # int - nr intreg
pret = 2300.500 # float -  nr zecimal
inmatriculata = False # bool - A/F
```

Functia type() si type casting

- O functie este o logica de cod predefinita care face ceva
- Are sintaxa nume_functie()
- In paranteze punem datele de intrare / input
- Vom discuta pe larg despre functii in capitolele urmatoare
- Functia type ne expune tipul de date al variabilei date ca input

```
nume = 'Andy'
print(type(nume)) # => <class 'str'>
```

- Functiile int(), str(), bool(), float() schimba tipul de date. (ex: int('3') => 3)

```
cifra = '3'
cifra = int(cifra) # schimbam tipul de date / type casting
print(type(cifra)) # => <class 'int'>
```

Functia print()

- Printeaza in consola ce punem intre paranteze
- Daca dorim sa facem o concatenare (adunare) de stringuri, putem face asta cu +

```
nume = 'Andy'
prenume = 'Sinpetrean'
print('Numele meu complet este ' + prenume + ' ' + nume)
```

- Daca dorim sa aduname int + string (mere cu pere), vom primi eroare
- Exista 2 solutii pentru a rezolva aceasta problema

```
nume = 'Andy'
varsta = 33
print('Ma numesc ' + nume + ' si am varsta de ' + str(varsta))
print(f'Ma numesc {nume} si am varsta de {varsta}') # aceasta varianta e recomandata
```

Assert

- Assert e o modalitate in programare de a face verificari
- Verifica daca statement (propozitia) este evaluata in final ca True
- Daca raspunsul e True, codul curge mai departe
- Daca raspunsul e False, codul se opreste si da o eroare. Nu se executa liniile de cod ce urmeaza
- Toate testele automate se termina in mod normal cu un assert, deci cu o verificare.

```
a = 1
# il intreb pe python: hey, a este egal cu 1?
assert a == 1
print('trec pe aici')
assert a == 2
print('nu mai ajung aici')
```

Functia input()

- Functia input() ne ajuta sa luam date de la tastatura si sa le salvam intr-o variabila
- Daca nu facem type casting, defaultul datelor date de user = string
- Ulterior putem accesa valorile salvate in variabile dupa necesitate

```
nume = input('Alege un nume') # default - string
varsta = int(input('Alege o varsta')) # fortam varsta sa fie un int
```

String (index, len(), slicing, metode)

- Fiecare caracter dintr-un string, are un numar asociat (index), incepand de la 0
- Functia len() ne spune cate caractere are stringul
- Slicing - putem accesa ‘felii’ din string folosind urmatoarea sintaxa
  - My_str[start_pos, end_pos, pas]
- Dupa my_str daca punem . ajungem la functii ajutatoare
  - Upper, lower, replace, count etc
  - Accesati descrierea lor apasand CTRL+Click pe numele lor

```
prop = 'Andy este prescurtarea de la Andrei'
print(len(prop)) # => 35
print(prop[0]) # => A
print(prop[0:3]) # => 'And', indexul de la last_pos se exclude, pasul e optional
print(prop[::-1]) # => parcurgere inversa
print(prop.upper()) # => tot cu litere mari
```