Ministerul Educaţiei al Republicii Moldova
Universitatea de Stat „Alecu Russo” din Bălţi
Facultatea de Științe Reale, Economice și ale Mediului
Catedra de matematică și informatică

Curriculum

la unitatea de curs

Programare funcțională (Python, Scala, etc.)

Ciclul I, studii superioare de licență
Codul şi denumirea domeniului general de studiu: 44 Științe exacte
Codul şi denumirea specialităţii: 444.1 Informatica
Forma de învăţământ: cu frecvență

Autor: Mircea PETIC,
dr., conf. univ.

Bălţi, 2017

Discutat şi aprobat la şedinţa Catedrei de matematică şi informatică

Procesul-verbal nr. 15 din 09.06.2017

Şeful Catedrei de matematică şi informatică

conf. univ., dr. Eugeniu PLOHOTNIUC

Discutat şi aprobat la şedinţa Consiliului Facultăţii de Ştiinţe Reale, Economice şi ale Mediului.

Procesul-verbal nr. 15 din 27.06.2017

Decanul Facultăţii de Ştiinţe Reale, Economice şi ale Mediului

conf. univ., dr. Ina CIOBANU

© *Mircea PETIC, USARB, 2017*

Informaţii de identificare a unitatea de curs

Facultatea: Ştiinţe Reale, Economice şi ale Mediului
Catedra de matematică şi informatică
Domeniului general de studii: 44 Științe exacte
Domeniul de formare profesională: 444 Informatica
Denumirea specialității: 444.1 Informatica
Denumire unității de curs: Programarea funcțională (Python, Scala, etc.)

Administrarea unității de curs

Codul unității de curs	Credite ECTS	Total ore	Repartizarea orelor				Forma de evaluare	Limba de predare
			Prelegeri	Seminar	Laborator	Lucrul indiv.		
S.03.A.119	5	150	30	-	45	75	Examen	Română

Anul de studii şi semestrul în care se studiază: anul II, sem. III
Regimul unității de curs: obligatorie (la libera alegere)
Categoria formativă: unitate de curs de specializare.

Informaţii referitoare la cadrele didactice

Mircea Petic, dr. în informatică, conferenţiar universitar. Absolvent al Universităţii de Stat „Alecu Russo” din Bălți, specialitatea „Informatica și limba engleză aplicată”. A efectuat stagii în diverse universităţi, inclusiv peste hotare, unde s-a specializat în domeniul informaticii, utilizarea tehnologiei informaţiei şi a comunicaţiilor în învăţământ, procesarea limbajului natural, programarea obiect orientată, programarea logică şi funcțională, ingineria programării, managementul proiectelor.

E-mail: petic.mircea@yahoo.com

Orele de consultaţii - miercuri: 15.00 -17.00. Consultaţiile se oferă atît în regim „faţă-în-faţă”, cît şi prin utilizarea poştei electronice, YahooMessenger (petic.mircea) şi Skype sau rețeaua socială facebook. Numele în Skype – mir_cescu.

Descrierea unității de curs

Unitatea de curs „*Programarea funcțională (Python, Scala, etc.)*” este un curs facultativ care permite aprofundarea cunoștințelor în programare. Realizarea unui sistem informatic reprezintă o activitate complexă și de durată, ce antrenează mari resurse materiale, umane și de timp. Programarea funcțională este o paradigmă de programare care tratează calculul ca evaluare de funcții matematice și evită starea și datele muabile. Se pune accent pe aplicarea de funcții, spre deosebire de programarea imperativă, care folosește în principal de schimbările de stare. Modelul matematic al programării funcționale îl reprezintă calculul lambda. Limbajele funcționale moderne pot fi considerate extensii ale calculului lambda. Noțiunea de bază în această paradigmă este cea de funcțională sau funcție de nivel înalt, o funcție care poate accepta ca argument sau returna ca valoare o altă funcție.

Limbajele de programare funcționale, mai ales cele pur funcționale, sunt promovate mai ales în mediile academice, fiind rar folosite în dezvoltarea de software comercial. În cursul dat vor fi studiate aspecte programării funcționale în baza limbajelor Scala, Python, Lisp și Haskell.

Integrarea în programul de studii

Studierea unității de curs „*Programarea funcțională (Python, Scala, etc.)*” se sprijină pe cunoştinţele, capacităţile şi competenţele dezvoltate în cadrul unității de curs „*Bazele programării*” și „*Programarea structurată*” studiate în ciclul I de licență. Finalităţile şi conţinutul unității de curs sînt corelate cu finalităţile şi conţinuturile unităţilor de curs menționate mai sus.

Competenţe prealabile

1. *Bazele programării*: Tipuri de date simple şi structurate, expresii.
2. *Programarea structurată*: Instrucțiunile simple și compuse (condiționale și repetitive). Funcții. Proceduri. Recursivitate.

Competențele dezvoltate în cadrul unității de curs

Competențe profesionale:

CP1. Operarea cu fundamentele ştiinţifice ale informaticii şi matematicii şi utilizarea acestor noțiuni în comunicarea profesională.

CP2. Elaborarea modelelor pentru descrierea fenomenelor şi proceselor reale.

CP3. Proiectarea, elaborarea şi analiza algoritmilor pentru rezolvarea problemelor.

CP4. Programarea, dezvoltarea şi mentenanţa aplicațiilor informatice în limbaje de nivel înalt.

Competențe transversale:

CT1. Aplicarea regulilor de muncă riguroasă şi eficientă, manifestarea unei atitudini responsabile față de domeniul profesional, pentru valorificarea optimă şi creativă a propriului potențial în situații specifice, cu respectarea principiilor şi a normelor de etică profesională.

CT3. Identificarea oportunităților de formare continuă şi valorificarea eficientă a resurselor şi tehnicilor de învățare pentru propria dezvoltare.

Finalităţile unității de curs

La finalizarea studierii disciplinei studentul va fi capabil să:

1. definească conceptul programare funcțională;
2. cunoască destinația și modul de lucru cu funcționale;
3. posede deprinderi de lucru cu expresiile lambda;
4. utilizeze corect evaluarea strictă și non-strictă;
5. posede deprinderi de lucru cu evaluarea lazy;
6. explice corect modul de programare funcțional cu limbaje nefuncționale;
7. explice corect diferența între programarea funcțională și imperativă;

Conținuturi

a) Prelegeri

Nr. d/o	Subiectele de studiu	Nr. de ore
1	Programare și limbaje de programare. Importanța programării funcționale ca nouă metodologie de programare.	2
Limbajul de programare funcțională – Scala		
2	Medii de dezvoltare Scala. Funcții de nivel înalt.	2
3	Structuri de date funcționale. Gestionarea erorilor fără excepții. Funcții stricte și non-stricte.	2
4	Paralelism pur funcțional. Testarea bazată pe proprietăți. Parsări	2
5	Monoizi. Monazi. Functori aplicativi și transversabili.	2
6	Lucrul cu datele externe.	2
7	**Lucrarea de control nr. 1**	**2**
Alte limbaje de programare funcțională		
8	Introducere în limbajul de programare funcțională Haskell. Deosebiri și asemănări	2

	cu Scala.	
9	Studiu de caz în limbajul de programare funcțională Haskell.	2
10	Deosebiri și asemănări între limbajele de programare Scala și CLISP.	2
11	Studiu de caz în limbajul de programare funcțională CLISP.	2
12	Programarea funcțională în limbaje nefuncționale (Python)	2
13	Programarea funcțională în limbaje nefuncționale (Java și C#)	2
14	**Lucrarea de control nr. 2**	**2**
15	**Recapitulare**	**2**
Total		**30**

b) Laborator

Nr. d/o	Tematica lecţiilor de laborator	Nr. de ore
1	Medii de programare funcțională pe Scala.	2
2	Module. Obiecte și spații de nume.	2
3	Funcții de nivel înalt.	2
4	Funcții polimorfice.	2
5	Structuri de date funcționale.	2
6	Gestionarea erorilor fără excepții.	2
7	Funcții stricte și non-stricte	2
8	Paralelism funcțional.	2
9	**Sarcini individuale nr. 1**	**2**
10	Parsarea documentelor JSON	2
11	Monoizi.	2
12	Monazi.	2
13	Lucrul cu datele externe.	2
14	Programarea funcțională în limbajul Haskell.	2
15	Programarea funcțională în limbajul CLISP – 1.	2
16	Programarea funcțională în limbajul CLISP – 1.	2
17	Programarea funcțională în limbajul Python - 1.	2

18	Programarea funcțională în limbajul Python - 2.	2
19	Programarea funcțională în limbajul Java.	2
20	Programarea funcțională în limbajul C#	2
21	**Sarcini individuale nr. 2**	**2**
22	***Susținerea proiectului Scala***	***2***
23	Recapitulare	1
Total		**45**

Activităţi de lucru independent

Proiectul va presupune elaborarea unei aplicații Scala care rezolvă o problemă din viața reală. Aplicația va conține și o descriere a problemei soluționate și a modului în care a fost rezolvată. Codul aplicației va avea comentarii explicative. Descrierea aplicației va fi prezentate într-un raport editat într-un document Word pe 3-4 pagini format A4, font #12, 1.5 intervale. Activitatea va fi evaluată atît de către colegi cît şi de către titularul disciplinei într-o şedinţă aparte.

Criterii de evaluare:

- Corectitudinea rezolvării problemei prin elaborarea aplicației Scala;
- Relevanța și valoarea comentariilor;
- Exactitate (logică, ortografică) a raportului prezentat;

Termenul limită (deadline) de prezentare a sarcinii – perechea a 22-a (laborator)

Exemple de teme pentru proiect

1. Numărarea cuvintelor dintr-un fişier txt scris în limba română cu diacritice (cel puțin 1000 cuvinte). Separatori de cuvinte sînt: spațiile libere, semnele de punctuație, simbolul ”-”.
2. Este nevoie de a automatiza evidenţa iepurilor din crescătorie. Se cunosc cîţi iepuri sînt la începutul fiecărei luni, cîţi au murit şi cîţi s-au născut în cursul fiecărei luni.
3. Determinarea greutății ideale a unei persoane cunoscînd înălţimea, vîrsta şi sexul persoanei. Formulele de calcul sînt: Gmasculin = 50 + 0.75 * (înălțime - 150) + (vîrsta - 20)/4, Gfeminin = Gmasculin – 10, unde înălţimea este exprimată în cm şi vîrsta în ani. Sexul indică separat.
4. O bază de date conține un tabel cu cîteva cîmpuri. Se cere să se transforme acest tabel într-un fișier JSON sau fișier txt. Numele bazei de date și cel al fișierelor de ieșire se va indica prin interfață.
5. Este nevoie de a transforma o suma de bani dintr-o valută în alta. În acest scop se va utiliza cursul de cumparare și cel de vînzare pentru valutele corespunzătoare. Se lucrează cu valutele: leul moldovenesc, dolarul american, euro, leul românesc, rubla rusă și hrivna ucraineană.

Resursele informaţionale la unitatea de curs

A. Literatura de bază

1. CHIUSANO, P. BJARNASON, R. *Functional Programing in Scala*, Manning Shelter Island, USA, 322 p.
2. ДУШКИН, Р. *Функциональное программирование на языке Haskell*, ДМК, Москва, 2016, 608 стр.

B. Literatura suplimentară

3. ХОРСТМАН, К. *SCALA для нетерпеливых*, ДМК, Москва, 2015, 408 стр.
4. ДУШКИН, Р. В. *Практика работы на языке Haskell*, ДМК, Москва, 2010, 288 стр.
5. РОГАНОВА, Н. *Функциональное программирование*, МГИУ, Москва, 2010, 215 стр.
6. MERTZ, D. *Functional programming with Python*, O'REILLY Media Inc., 2015, 49 p.

C. Resurse Internet

7. Situl oficial IDE IntelliJIDEA [online], [vizitat 09.10.2016] Disponibil: < www.jetbrains.com >
8. Situl oficial Scala IDE [online], [vizitat 09.10.2016] Disponibil: <http://scala-ide.org/>
9. Online compiler and IDE [online], [vizitat 09.10.2016] Disponibil: <http://ideone.com/ >
10. Documentația oficială Scala [online], [vizitat 09.10.2016] Disponibil: < http://www.scala-lang.org/ >

Evaluarea

Cunoştinţele, capacităţile şi competenţele studenţilor vor fi evaluate:

1. La prelegeri (**PR**):
   1.1. Lucrare de control scrisă nr. 1: perechea a 7-a (**LC1**).
   1.2. Lucrare de control scrisă nr. 2: perechea a 14-a (**LC2**)
2. În cadrul lecţiilor de laborator (**LLab**):
   2.1. Sarcini individuale nr. 1: perechea a 9-a (**SI1**)
   2.2. Sarcini individuale nr. 2: perechea a 21-a (**SI2**)
   2.3. Proiect: perechea a 22-a (**Pro**)
3. La examenul final, conform orarului întocmit de decanat (**Ex**).

Nota finală la disciplina „Programarea funcțională” se calculează conform formulelor:

N_evaluarea_curentă=1/2x**PR**+1/2x**LLab**

N_finală=0,6x*N_evaluarea_curentă*+ 0,4x*N_examen*,

Unde **PR**=(**LC1**+**LC2**)/2 şi **LLab**=(**SI1**+**SI2+Pro**)/3

Examenul final se susţine scris, care va include un test complex cu diferite tipuri de itemi. Pentru a fi admis la examen, este obligator ca ambele note (**PR** şi **LLab**) să fie cel puțin 5.0.

Recuperarea notelor și susținerea repetată a examenului are loc în datele stabilite de orarul întocmit de decanat de susținere a restanțelor.

Baremul de apreciere

(în baza REGULAMENTULUI
cu privire la evaluarea rezultatelor academice ale studenţilor în
Universitatea de Stat „Alecu Russo” din Bălţi)

Procentajul	Nota
100 – 91	10

90 – 81	9
80 – 71	8
70 – 66	7
65 – 61	6
60 – 51	5
50 – 41	4
40 – 31	3
30 – 16	2
15 – 0	1

Principiile de lucru în cadrul disciplinei

1. Calendarul cursului (termenii-limită de prezentare a sarcinilor propuse spre rezolvare, momentele de evaluare etc.) este corelat cu calendarele la alte discipline din semestru. De aceea prezentarea sarcinilor după termenul-limită indicat în calendar nu este salutată, iar studenţii care amînă frecvent prezentarea sarcinilor îşi formează o imagine nefavorabilă.
2. Nu este salutată întîrzierea la ore.
3. Este salutată poziţia activă a studentului, care studiază din propria iniţiativă noi conţinuturi, propune soluţii (aplicaţii, instrumente Web), formulează întrebări în cadrul prelegerilor şi a orelor practice.
4. În cadrul disciplinei o atenţie sporită va fi oferită respectării principiilor *etice*. Prezentarea unor soluţii a sarcinilor, preluate de la colegi sau din alte surse, preluarea informaţiilor din diverse surse, fără a face trimitere la sursă, va fi considerată *plagiat* şi va fi sancţionată prin note de „1” .