Curs 2:
##
#################

1.Model church-turing
R: Spune ca un calcul poate fi facut de o masina doar daca este calculabil in sens Turing. Orice problema de calcul este calculabila daca se incadreaza in una din categoriile: general recursive functions, lambda-computable sau Turing computable(de fapt toate 3 sunt echivalente)
Turing(Automate), church(calcul functional lambda calcul pur formal pt automate)

2.Care sunt problemele masinii virtuale java sau c# ?
R: Performanta (datorita interpretoarelor) este 1-3 ori mai lenta decat implementarile similare in C(unsafe). Java e pe cale de disparitie, dar poate exista ptc sunt deja multe aplicatii in java(20 ani)

3.Ce este polyglot?
R: Polyglot este un limbaj de programare care are mai multe interpretoare(graal etc) si practic poti folosi pentru a rula mai multe limbaje. Se foloseste de AST pentru a scapa de diferentele de sintaxa
protocol de interoperabilitate
limbaj gazda (java de ex ->limbaj de baza Polyglot)
limbaje tinta(limbajele folosite in fct eval)

4.Implementare AST? De ce e mai buna decat o masina virtuala?
R: ast = abstract syntax tree; utilizare ast = ascunde diferentele de tratare a fiecarui limbaj; ajuta sa punem cap la cap instructiunile din toate limbajele -> nu mai conteaza din ce limbaj vin instructiunile

5.Ce este hotspot? Ce este graal?
R: Graal este interpretorul lui Polyglot. Hotspot este o masina virtuala java. Graal este scris in Java si se bazeaza pe Hotspot care e scris in C

6.Care sunt avantajele Graal fata de hotspot?
R: hotspot este doar proiectul initial, graal e bazat pe hotspot si aduce alte functionalitati: libraria truffle(care permite accesul la limbaje precum R,Ruby,js etc) + tool-uri + compiler nou ===> avantaj de timp la rularea programelor

7.Care sunt limbajele suportate de Graal la ora actuala?
R: js,ruby,r,python,java

8.Ce este calculul functional?
R:

9.Ce este o functie lambda?
R: Putem crea o functie fara sa le asignem un nume explicit(functii fara nume)

10.Explicati functiile Curring(?)

R: functii cu argument multiplu pot returna mai multe valori + functii ce primesc serial argumentele(primul apel are n parametri, urmatorul n-1, urmatorul n-2 pana se ajunge la functia cu un parametru care returneaza o valoare = recursivitate)

11.Diferenta dintre tipurile dinamice si tipurile statice de date?
R: tipuri statice = fiecare variabila este bine legat de un tip de data(de ex int c++). NU SE POATE SCHIMBA TIPUL UNEI VARIABILE IN TIMPUL PROGRAMULUI; tipuri dinamice = de ex python

12.Limbaje ce suporta tipuri "tari" de date si tipuri "slabe" de date
R: python suporta ambele(de ex poti face int(1) + float(2.3), dar nu poti 1 + "2.3"), javascript(tip slab de date) de ex: 1 + "2" = "12" -> face conversii implicite

13.Enumerati tehnicile curente de optimizare a unui cod
R: reorganizare noduri AST, evaluare partiala

14.Reorganizarea AST? Avantajele utilizarii acestei metode
R: Reorganizarea unui AST practic schimba structura arborelui astfel incat sa se evite pasi in plus, sporind practic eficienta codului

15.Ce inseamna evaluarea partiala?
R: Evaluezi diferite parti ale programului in vederea optimizarii(de exemplu partile pe care deja le stii inainte de compile time, codul)

16.Ce este un evaluator partial?
R: Un evaluator partial particularizeaza programele. De exemplu, daca un program este o functie prog definita pe date intrare(cunoscute la compile time) + date ce urmeaza a fi introduse in program de utilizator cu valori in Output, atunci un evaluator partial transforma programul intr-o versiune noua ce incapareaza datele deja cunoscute, specializand practic programul si facandu-l mai eficient(eventual se rescriu liniile de cod)

17.La ce se refera specializarea unui program?
R: Pentru un set de date(in1) programul se specializeaza, scriindu-se mai eficient instructiunile(de ex trecere de la program structurat la program nestructurat)

18.Optimizare in Graal?
R: Mai putine instructiuni datorita evaluarii partiale

19.Metode depanare Graal?
R: Logg(trimitere de mesaje) try-throw-catch

20.Ce sunt metricele unui program si de ce ne-ar trebui?
R: De ex: memorie RAM utilizata, viteza de executie, memorie utilizata(octeti),numar linii cod !!CounterKey(timp scurs). Poate gasesti leaks de memorie

21.La ce se refera dumping-ul?(poate unui executabil)
R: Se ingheata memoria pt functia care s-a oprit incorect(a crapat) din functionare si se salveaza datele cu scopul de a face debug pe ele

22.Constant Folding
R:

Curs 3:
##
#

1) Ce presupune descompunerea functionala?
R: Fragmentarea/impartirea proiectului pe module care, individual, realizeaza anumite functionalitati.

2) Care sunt dezavantajele specificatiilor?
R: Ele pot fi incomplete, gresite partial/total si inselatoare (din cazua diferentelor de limbaj si reprezentare dintre proiectant si beneficiar). Aceste dezavantaje apar din cauza faptului ca un client isi poate dori noi imbunatatiri pe parcursul dezvoltarii proiectului.

3) Ce sunt coeziunea si cuplarea?
R: Coeziunea = cat de apropiate sunt operatiile dintr-o metoda
Cuplarea = cat de stransa e legatura intre doua metode

4) Pasi pt proiectarea POO.
R: -izoleaza
-abstractizeaza (ia obiecte din realitate si le retine aspectele importante, creand o descriere)
-determina responsabilitatile/relatiile dintre obiecte

5) Metodologia POO
R: -brainstorming (intr-un grup, fiecare persoana expune spontan propriile idei, activitatea fiind "moderata"/"coordonata" de un lider. Initial toate info sunt considerate corecte)
-filtrarea claselor (determinarea listelor de clase candidate si reducerea claselor -> exista clase care difera f putin si pot fi inglobate in aceeasi clasa)
-crearea de scenarii (necesare pt a fi siguri ca exista o relatie corecta intre clase)
-algoritmi (proiectati pt a defini toate actiunile pe care clasele trb sa le poata efectua)

6) La ce se folosesc fisele CRC? (class responsibility collaboration)
Numele clasei, superclasa, subclasa + Responsabilitati si colaborari intre clase.

7) Flow-ul general pt implementarea si dezvoltarea unei probleme OOP?
Definirea spatiu de obiecte, Implementarea
Abstractizare -> retinerea informatiilor importante ale unui obiect/concept real
Incapsulare -> Izolam anumite date de exterior.
Mostenirea -> preluarea proprietatilor altor clase

8) Cum creeaza kotlin obiectele asociate unei clase?
Instantele claselor se creeaza prin intermediul constructorilor. (nu exista cuvantul cheie new, trecerea de la abstract la clase)

Clasa reprezinta o descriere a unui concept, iar obiectul (instanta) reprezinta o referinta catre un obiect ce contine membrii in memorie.

9) Care este rolul lui "this"?
This face referire la instanta curenta.

10) Care este rolul lui scope?
Specificatori de acces -> protejeaza anumite elem

11) Cum se face controlul fluxului de executie ca o expresie?
Prin structuri de decizie (if-else si when (= switch) si try-throw-catch).

12) Ce este NULL-ul?
Un pointer care nu contine nicio informatie si nu pointeaza nicaieri (pt a "distruge" pointerul dupa dealocare). Exista o problema cu variabilele nullable si non-nullable

13) Explicati versificarea si conversia de tip in Kotlin.
Conversie implicita (inteligenta) - nu e nevoie sa specifici tipul de date.

14) Cum se poate folosi "when" ca un "switch case" in paradigma structurata?
In loc de default -> else
Cazurile (ex case 1: se scriu 1 ->)

15) Cum se poate utiliza when ca o expresie?
O functie = when(...);

16) Ce este clasa anonima?
Este o clasa care se foloseste in cazul unor apeluri singulare specifice.

17) Ce sunt clasele pt gestiunea datelor?
Clasele de tip data sau enum. Clasele de tip data sunt clase care au DOAR rolul de a pastra date (informatii)

18) Ce sunt metodele statice in Kotlin?

19) Ce sunt obiectele companion si unde se folosesc?

20) Suporta kotlin mostenirea simpla direct?

21) Ce sunt functiile cu expresie unica?

22) Ce sunt functiile membru?

23) Cand sunt recomandate functiile locale?

24) Ce sunt functiile top-level?
R: Fct top-level sunt fct globale, definite in afara oricarei clase/obiect.

25) Ce este list comprehension si cum se poate face in kotlin?

26) Functii cu expresie unica.
R: Sunt facute pe o singura linie, iar egalul e pus pe post de return -> fun sum(a: Int, b: Int) = a+b;

27) Cuv cheie open
R: Cuv cheie open sugereaza faptul ca acea clasa poate fi mostenita/derivata. Implicit, in Kotlin, clasele sunt nederivabile.

28) Clase singleton
R: Clase care pot fi instantiate o singura data.

##
Curs 4:

1) Ce este proiectarea aplicatiilor software? (cum proiectam aplicatia propriu-zisa)
R: Cu ajutorul diagramelor UML. Proces de rezolvare a unor probleme, obiectivul fiind sa se gaseasca
si sa se descrie o cale de a implementa necesitatile functionale ale sistemului, tinand cont de
constrangerile impuse de (nivel de calitate asteptat, specif platformei tinta, fonduri disponibile etc)

2) Care e diferenta intre must-have si nice-to-have?
R: Must-have e lucrul de baza pe care il vrea neaparat clientul, iar nice to have e un lucru bun de avut, dar nu neaparat

3) Definiti o componenta.
R: O piesa de hardware/software care are un rol bine definit si poate fi inlocuita oricand cu alta

4) Din ce e alc un sistem daca il analizam din prisma analizei modulare?
R: Componente, subsisteme, module.

5) Mecanisme centrale?
R: persistenta, gestionarea memoriei, gestionarea proceselor, gestiunea mesajelor, gestiunea retelei
de comunicatii, tranzactiile, tratarea evenimentelor/exceptiilor, interfata cu utilizatorul

5') Modelul domeniului
R: System -> nume, responsabilitati
Subsistem
Component -> name (ele pot fi schimbate cu unele imbunatatite)
Module, Framework

6) Ce este UML-ul?
R: Este un limbaj grafic pt vizualizarea, specificarea si dezolvarea si documentarea unui sist software de dimensiuni medii sau mari.(Unified Modelling Language)
= un standard pt a crea schema unui sistem

7) Ce este modelul de proiectare si implementare in cascada?
R: Fiecare noua functionalitate se implementeaza dupa ce acela anterior este complet realizat si testat.
Etape: specificarea cerintelor, analiza sistemului, proiectarea sistemului, implem sistemului testarea sistemului, instalarea sistemului, mentenanta sistemului.

8) In ce conditii se pot suprapune atat waterfall-ul cat si AGILE-ul?
R: In waterfall, o etapa viitoare nu poate incepe inainte ca o etapa din trecut sa se fi incheiat (o etapa noua poate depinde de o etapa din trecut)
In AGILE, noile etape depind de cele vechi, dar se proiecteaza simultan (pot depinde unul de celalalt in paralel)

9) Enumerati modelele sistem (structura, functionalitate, comportament dinamic)
R: Modelul obiect (str sistemului, obiectele si relatiile)
Modelul functional (fluxuri de date din sistem)
Modelul dinamic (cum reactioneaza sist la stimuli externi)

10) Enumerati modelele taskurilor
R: Harta PERT (specifica legaturile intre taskuri)
Planificarea (cum poate fi acest task realizat in durata de timp rezervata)
Harta organizationala (care sunt rolurile in proiect sau organizatie)

10') O diagrama este o reprezentare vizuala intr-un model

11) Enumerati mecanismele centrale ale unei aplicatii OOP.
R:

12) Cum se defineste un model dpdv al UML?
R: Descriere completa a unui sistem dintr-o perspectiva particulara (lucrurile se vad pe portiuni)

12') Tipuri de proiectare specifice
R: proiectare arhitecturala
- proiectarea claselor
- proiectarea interfetei vizuale
- proiectarea bazelor de date
- proiectarea algoritmilor
- proiectarea protocoalelor

13) La ce e buna abordarea bazata pe componente in proiectare?
R: Componentele pot fi inlocuite ulterior cu unele mai bune, acest lucru fiind posibil datorita nivelului ridicat de incapsulare.

14) Dati exemplu de aplicarea principiului cresterii coeziunii
R: grupeaza ce este apropiat cat mai mult (intareste cat mai mult ce e similar)

15) Exemple de aplicare a principiului reducerii cuplarii
R: Cuplarea apare atunci cand exista interdependente intre un modul sau altul
 - cuplarea e mai stransa cu cat exista mai multe legaturi intre module/blocuri

16) La ce se refera principiul de baza in gestiunea abstractiilor?
R: ascund padurea ca sa nu ma incurc de copaci
 entitate generica rezultate din analize (nu e necesar sa stim detalii de implementare (tip de date))

17) Care sunt principiile complementare reutilizarii? (cum facem o clasa care opereaza pe niste numere sa fie reutilizabila)
R: proiecteaza pt reutilizare + proiecteaza PRIn reutilizare

18) La ce se refera proiectarea pt flexibilitate?
R: Anticiparea schimbarilor ce pot aparea pe parcurs. (reducerea cuplarii + cresterea coeziunii
-> mai usoara inlocuire a unor chestii in sistem)
-polimorfism
-utiliz de cod reutilizabil
-mecanism de generare a exceptii

19) Cum se gestioneaza problemele de inlocuire sau disparitia obiectelor puse la dispozitie de o tehnologie sau framework?
R: De ex, in cazul inlocuirii, se asigura backwards compatibility prin respectarea principiilor SOLID (Liskov substitution)

20) La ce se refera proiectarea pt portabilitate?
R: Aplicatia sa poata fi rulata pe mai multe sisteme de operare cu usurinta. -> utilizand un interpretor asemenea JVM-ului??
- supravietuirea pe termen lung a softwareu-lui creat
- utilizarea de limbaje interpretate sau framework-uri specializate
21) La ce se refera proiectarea pt testabilitate?
R: Trb luate masuri pt a usura procesul de testare (manuala, automata)
 -> utilizam design patternuri
 -> utilizarea unor feature-uri care accepta linie de comanda

21') Scopurile proiectarii OOP
-> usurinta in modificare sau adaugarea unor noi functionalitati
-> gestionarea independentei intre clase si pachete
-> scaderea motivelor pt care modelul s-ar putea schimba daca ar aparea schimbari in clase/pachete

22) Coeziunea claselor
R: ->plasarea metodelor care opereaza pe anumite date cat mai aproape de datele respective

23) Ce este principiul de raspundere unica? (Single responsability principle)
R:

24) Principiul separarii interfetelor?
R:

25) Alte principii SOLID?
R:

26) Ce este BPP-ul?
R: Bune practici de proiectare (Divide and conquer)

27) TDD = Test-Driven Development
scrie(un test)
executa(toate testele)
scrie(codul tinta)
executa(testele)
refactorizeaza(codul)

1 - write a test that fails
2 - make the code work
3 - eliminate redundancy

Curs 5
###

1) Ce este un eveniment?
R: Un eveniment = un tip de semnal generat catre program
-> indica programului faptul ca s-a intamplat ceva

2) Prezentati o maniera de gestionare a unui eveniment, inclusiv OS-ul!
R: Evenimentul e generat de tastatura/mouse, e adaugat intr-o coada si este trimis aplicatiei, unde se executa, apoi OS-ul il trimite rezultatul pe monitor

3) Cum se gestioneaza dpdv arhitectural un eveniment?
R: Toate elem sunt bagate intr-o coada de evenimente si ele sunt scoase pe rand din aceasta pt a fi executate

4) Diferenta intre evenimente sincrone si asincrone
R: Evenimentele asincrone nu asteapta confirmari. (Hai baieti la bere -> cine reactioneaza se duce)
  Cele sincrone asteapta confirmare pt a fi executate

5) Care sunt sursele comune elementelor sincrone si asincrone?
R: Mouse-ul si tastatura.

6) Cum se trateaza un eveniment dpdv al programatorului?
R: Polling (citire secventiala intr-o bucla infinita a tuturor comenzilor dispozitivelor de intrare)
Interrupt-driven

7)
R:

8) La ce se refera EDP si care sunt componentele principale implicate? (Event driven programming)
R: Are ca si componente:
-generatoare de evenimente
-sursa evenimente
-bucla de evenimente
-gestionare de evenimente
-event mapper (asociere intre evenim si gestionare -> coada de evenimente)
-inregistrare evenimente

9) Ce cuprinde diagrama de secvente specifica EDP?
R: Un appuser, event, eventhandler.

10) Ce este dispatcherul in EDP?
R:

11) Ce este wys-wyg (what you see is what you get) si care e legatura cu EDP?
R:

12) Explicati look-n-feel-ul!
R:

13) Care sunt criteriile minimale care trb indeplinite de un GUI pt a funct corect?
R:

14) La ce se refera BIG Corellation
R:

15) Ce este un widget?
R: Este un obiect din cadrul unui GUI orientat obiect

16) Explicati MVC-ul (Model view controller)!
R: Controler, view, Model

17) Ce este SDL si unde se foloseste
R: SDL este o biblioteca grafica dezvoltata pt C++ -> se foloseste pt crearea

de aplicatii vizuale pe entitati cu resurse limitate

18) Care sunt modulele SDL si care sunt echivalentele lor cu DivX.
R: Componente SDL: video, gestiunea evenimentelor, joystick, audio, cd-rom, fire de executie, timere
Componente DirX:directDraw, directInput, direntInput, directSound, Fara ech (F/E), F/E, F/E

19) Ce este List Comprehension si care sunt cele mai comune utilizari
R: generarea unei liste sau a unui vector intr-o singura linie (list1 = [i for i in range(0,8) if (i%2==0)]

20) Ce este un eveniment virtual si cum se trateaza in Tkinter
R: Combobox (care opt e afisata implicit) -> org sub forma de grid

21)..

22) Explicati maniera arborescenta de creare a meniurilor in Tkinter.
R:

23) Ce trebuie urmarit cand se descrie un frame la un moment?
R:

24) Explicati tratarea evenimentelor in Android.

---

1. Ce este un eveniment?
   Un semnal generat de catre o aplicatie, un sistem hardware.
   Poate fi definit ca un tip de semnal generat de cate program. Indica ca s-a intamplat ceva
   Ex: buton de mouse, miscare de mouse,
2. Prezentati o posibila maniera de gestionare a unui eveniment, inclusiv OS-ul
   Tastatura -> OS -> coada evenimente -> programe -> OS -> monitor

3. Cum se gestioneaza dpdv arhitectural un eveniment?
   Se baga intr-o coada de evenimente si cand trebuie sa fie tratete se scot pe rand din coada

??
4. Care este diferenta dintre evenimentele sincrone si cele asincrone?
   Sincron : trimit mesaj, astept confirmarea ca primul a ajuns
   apoi trimit al doilea mesaj. eu rezerv resurse care sa primeasca evenimentele
   asincron : trimit toate deodata nu astept confirmare
   nu ne mai batem capul daca va fi receptionat si tratat
   nu se mai blocheaza aplicatia sa astepte confirmari
   mouse tastatura : asincron
5. Care sunt sursele comune pt evenimentele sincrone si asincrone?
   Mouse-ul si tastatura

6. Cum se trateaza un eveniment dpvd al programatorului?
   Se trimite la distribuitor care le trimite la un anumit gestionant

7. Care sunt avantajele procesarii orientate pe evenimente?
   mai portabile
   permit tratarea rapida a erorilor
   se po folosi in time-slicing
   incurajeaza reutilizarea codului
   merge mana in mana cu oop

8. La ce se refera EDP si care sunt componentele principale implicate?
   Event driven programming
   Generatoare de evenimente
   Sursa evenimentelor
   Bucla de evenimente
   Gestionari de evenimente
   Event mapper
   Inregistrarea evenimentelor

9. Ce cuprinde diagrama de secventa specifica EDP?

10. Ce este dispecerul in EDP?
   Cel care controleaza evenimentele
   Distribuie evenimentele

11. Ce inseamna wizzy wig si care este legatura cu EDP?

12. Explicati look-n-feel?

13. Care sunt criteriile minimale care trebuie implementate de un GUI pentru a se mula pe utilizator?

14. La ce se refera PIC correlation?

15. Ce este un GUI chat?

16. Explicati MVC-ul
   Model view controller - modelezi datele, view inseamna cum arati datele, controller da operatii modelului
   Controler modifica starea modelului, cand se schimba starea view-ul afiseaza noua stare

17. Ce este SDL si unde se foloseste?
   Este o biblioteca folosita la creare de software

SimpleDirectMediaPlayer

18. Care sunt modulele SDL si echivalentele lui cu DIVX?
Video - DriectDraw
Gestiunea evenimentelor - DirectInput
Joystick - DirectInput
Audio - DirectSound
CD-ROM - NU are echivalent
Firul de executie - NU are echivalent
Timere - NU are echivalent

19. Ce este list comprehension? Cele mai comune utilizari
Scriem liste/structuri multidimensionale pe o linie
list1 = [i**2 for in range(0,8)] (pentru for)
list1 = [i for i in range(0,8) if(i%2==)] (pentru if else)
list1 = [i**2 if (i%2==) else i**3 for i in range(0,8)] (if else)
//matrice : list1 = [j for j in range(0,6) for i in range(0,3)]

20. Ce este un eveniment virtual? Cum se

21. Ce este si unde este necesara organizarea matriceala a entitatilor intr-un GUI python?
Fiecare element este pe o linie si o coloana.

22. Explicati maniera arborescenta de creare a meniurilor in tkINTER
Creez submeniurile si apoi le asamblez intr-un meniu mare

23. Ce ar trebui urmarit cand trebuie tratat un eveniment Frame

24. Explicati tratarea evenimentelor in Android

25. Cum se pot adauga servicii in Android?

----------CURS
6----------

1) Enumerati domeniile posibile de utilizare python (se poate face orice cu python)
R: - aplicatii web
- frontend(inclusiv pt cloud)
- backend
- hardware low level (suport nativ pt microcontrollere)
- arhitecturi complexe bazate pe microservicii
- inteligenta artificiala
- securitate (mai mult automatizare si criptografie, dar nu treburi serioase)
-treburile serioase de securitate se fac cu assembly

2) Care sunt paradigmele de programare suportate de python?
R: - paradigma functionala
- paradigma orientata obiect

- paradigma imperativa
- paradigma procedurala

3) Ce tipuri de conversii explicite pt tipurile de date suporta python?
R: integer(val), long(val), complex(real,imag), string(str)

4) Explicati instructiunea try-except!
R: In blocul try se scriu instructiuni care ar putea genera erori (exceptii) - imp cu 0, nedeschidere fisier etc.
- in except se trateaza exceptia respectiva
UTILA atunci cand ne dorim sa nu crape programul cand avem o exceptie (o transmitem mai sus
si blocul parinte o trateaza)
except Exception (tratam orice)

5) Ce este o exceptie?
R: Raspunsul la o situatie exceptionala ce poate aparea in timpul rularii unui program (impartire cu 0, nedeschidere fisier
index out of bounds etc.)

6) Unde este utilizata instructiunea pass?
R: -nimic

7) Ce diferente sunt in python fata de C in termeni de operatori simpli?
R: Operatorul putere si impartire intreaga (python). Nu avem ++ si -- in python

8) Ce tipuri de operatori de atribuie sunt suportati in python?
R: =, +=, -=, ... /=, *=, %=, **=, //=

9) Care e diferenta intre operatorii de apartenenta si cei de identitate?
R: - Cei de apartenenta verifica daca 2 operatori se afla intr-o secventa (a in b)
- Cei de identitate verifica daca 2 variabile au aceeasi referinta. (a is b)

[10) Caracterul = ceea ce se afiseaza pe ecran -> imaginea caracterului (matrice 8/8 cu puncte luminoase)]
-fiecarui caracter ii e asociat un cod (tabela ascii) astfel incat sa se determine punctele luminoase in matricea resp
UNICODE = imagine net superioara a caracterelor pe ecran (pe mai multi biti, matrici mai mari de puncte -> densitate mai mari, calitate mai buna)

11) De ce se poate executa un cod in interiorul unui sir in python?
R: Datorita faptului ca python e un limbaj interpretat (pot sa execut un string). Cu eval, exec.
- exec = se adauga variabile globale, locale mediului temporar (mini program)
- eval = evalueaza un string ca o expresie python si intoarce rezultatu (similar exec)
- substitute

12) Ce este eval-ul?

R: Python interpreter-ul face exact ceea ce face eval-ul: interpreteaza un sir de string-uri si returneaza un rezultat.

13) Care sunt diferentele intre un tuplu si un dictionar?
R: Tuplele sunt immutable, iar dictionarele sunt mutable.
Tuplele sunt un fel de liste constante, iar dictionarele niste hashtable-uri (contine chei si fiecarei chei ii este asociata o valoare ->hash value)
Tuplele = colectii indexate de date (declarate cu paranteze rotunde)

14) Care sunt atributele unui obiect de tip fisier?
R: .read(), .readline(), .readlines(), .rename(), .delete(), .mkdir(), .chdir()

15)
R:

16) Cum se trateaza apelul unei functii in python? (ce se intampla cand interpreterul de python intalneste apelul unei fct?)
R:

17) Cum poti simula transferul prin referinta la iesirea din functie in python?
R: Prin tablouri

18) La ce ne folosesc cuv cheie date ca parametru in python?
R:

19) Care sunt diferentele intre C++ si suportul primar de OOP oferit de python?
R: - in python nu mai exista ;
- in python marcajul de bloc se face prin indentare
- in python avem self, iar in C++ this
- pt copierea unui obiect in python folosim clone()

20) Cum se poate adauga rapid persistenta in python? (sa stochezi f rapid ceva ca sa fie valabil intre 2 sesiuni)
R: cu shelve

21) De ce as fi nevoit sa construiesc elem de baza ale unui GUI in python de la 0?
R: pt ca vreau sa modific functionalitatile

###################################### CURS 7
##
#####
- Paradigma Programarii Generice

1) Ce sunt functiile parametrizate?
R: Functii bazate pe generice (argumente parametrizabile)
-> Any (Kotlin) = Object (Java)
-> Unit (Kotlin) = void (Java)

2) Ce sunt tipuri parametrizate? (tipuri de date parametrizate) -> Ex: Array<Int> sau Array<Double> etc..
R: Sunt definite ca tipurile container (parametri pt clase)
-> cele mai cunoscute tipuri parametrizate sunt containerele

3) La ce se refera polimorfismul limitat superior?
R: Se refera la limitarea tipului de date la subtipurile celui mentionat.
R: In Kotlin restrictioneaza tipurile de parametri la subclasele unei anumite clase.

4) Ce sunt limitarile superioare multiple?
R: Limitare multiple: Ex: fun<T> minSerializable(first: T, second: T):T
where T: Comparable<T>, T: Serializable
-> se foloseste clauza "where"

5) Ce sunt tipurile generice de date?
R: clase/interfete care au tipuri parametrizabile de date. (Au un parametru T care poate fi Int, Double etc).

6) Ce sunt modificatorii de tip?
R: out -> poate fi folosit doar ca tip de return (parametru covariant de tip -> producator de T)
in -> poate fi folosit doar ca parametri la functii (parametru contravariant de tip -> consumator de T)

7) Ce este declaration site variance? (varianta la momentul declararii)
R: iesire -> produs (out) -> covarianta
intrare -> consumat (in) -> contravarianta

8) Ce este variance annotation?
R: Variance annotation se refera la parametrul covariant de tip: out.

9) Explicati clasele duale: producator/consumator.
R: Clase care folosesc un parametru covariant out si un parametru contravariant in.

10) Explicati type projection.
R:

12) Ce sunt functiile generice?
R: Functii care au la randul lor tipuri parametrice (specificate inaintea numelor functiilor)

13) Ce sunt constrangerile generice?
R: Ex. limitarea superioara

14) Ce este type erasure? (pierderea tipului)
R: Nu avem mecanismele necesare de a verifica tipurile de date la runtime.

15) Ce sunt tipurile de date algebrice?
R: Similar grupurilor din algebra

- se refera la un set inchis de tipuri compozite, alaturi de operatiile aferente

16) Ce sunt clasele sealed?
R: Clasa a carei functionalitate e restransa la fisierul in care e descrisa.
- contine operatiile asociate
-> sunt utilizate de obicei in calcul functional

17) Ce sunt colectiile?
R: Colectiile sunt grupuri de elemente ne ajuta sa stocam mai multe obiecte in acelasi loc.
Exemple: list, map, set

18) Ce operatii ne ofera colectiile List?
R: operatii CRUD -> isEmpty, contains, containsAll, get, indexOf, append

19) Ce sunt listele in Kotlin?
R: Listele sunt colectii generice ordonate care pot contine valori duplicate (spre deosebire de Set)

20) Care e diferenta intre List si MutableList?
R: In mutableList putem modifica elementele (sunt permise mult mai multe operatii fara de List)

21) Ce sunt colectiile Sealed?
R: O colectie de elemente care nu pot fi modificate???

22) Ce operatii ofera aceste colectii de tip Set?
R: Multimi practic

23) Ce este un Linked Hash Set?
R: Implementare a lui Set folosind un Hash_Table

24) Ce este un tree set?

25) Ce este o colectie Map?
R: Un hashmap -> o structura de date care contine elemente numite chei, carora le sunt mapate
anumite valori

26) Ce operatii ne ofera aceste colectii de tip Map?
R: -isEmpty, containsKey,containsValue, get, put etc.

27) Cum se poate parcurge un Map cu o bucla for?
R: Se parcurge cu forEach sau forKey
R: for(key in nume_map)
nume_map[key] -> ne da value (valoarea hash)
-

28) HashMap vs LinkedHashMap vs TreeMap.

R: LinkedHashMap = HashMap + LinkedList (lista dublu inlantuita)
TreeMap -> implementare pe arbori red-black

29) Ce este colectia Array?
R: Container care poate retine un nr fix de elemente de un tip dat.

30) Care e diferenta intre colectiile Java si colectiile Kotlin?
R: Colectiile Kotlin sunt deasupra celor java

31) Ce sunt tipurile platforma?
R: Marcate cu semnul exclamarii (!)

CURS 8
###

1) Ce este un model de proiectare (un design pattern)?
R: Pattern = O forma sau un model propus pt imitare
- ceva proiectat sau folosit ca model pt a face lucruri - folosit in diferite contexte (calapodul croitorului)
- o forma sau un proiect
- o configuratie de evenimente

2) Cum a aparut termenul de model de proiectare (termenul de design pattern)?
R: -> Cristopher Alexander (...) noi maniere de proiectare a unor cladiri si forme urbane
Fiecare poate fi vazuta ca o regula cu 3 parti

3) Cum a evoluat conceptul de design pattern?
R: - '87 -> Cunningham si Beck au facut un limbaj
- '90 -> gasca celor patru -> scrierea unui catalog care continea modele de rezolvare a unor probleme
- '95 -> GoF (gang of four) publica o carte pt design pattern-uri

4) Explicati modelele Riehle si Zuligoven ??? (not sure)
R: Cei doi mentioneaza 3 tipuri de modele software
- model conceptual:
- model de proiectare: explica modele software
- model de programare: constructii de limbaj pt a descrie forme

5) Unde sunt utile modelele de proiectare?
R: Sunt utile cam peste tot (chiar si in asamblare)
- pot fi aplicate la mai multe niveluri de abstractizare
- solutia unei probleme in caadrul unui context particular
- proiectarea unei clase poate fi reutilizata (implementari ca si colectii si generice)

6) Care sunt elementele unui model de proiectare?
R: - numele : trebuie ales a.i sa descrie pe scurt problema de proiectare, solutia si consecintele

- problema : se descrie situatia/cazurile in care se aplica acest model de rezolvare (+ conditii pt aplicare)
- solutia : descrie abstracta a modelului, a elementelor care compun proiectul si relatiile dintre ele
- consecintele : rezultatele obtinute si compromisurile facute

7) Ce este o arhitectura inchisa?
R: Arhitectura in care nu mai pot interveni (o aplicatie pe care numai cel care o produce poate aduce noi functionalitati)

8) Ce sunt modelele creationale?
R: Se ocupa de abstractizarea si instantierea unui proces
(se incapsuleaza cunostintele despre clasele concrete)

9) Explicati fabrica de obiecte (factory method).
R: - obiectele nu mai sunt create prin operatorul new, ci utilizand metoda fabrica (aceasta ne creeaza obiecte)
- practic este utila deoarece ascunde apelurile operatorului new si toate logica din spatele construirii anumitor obiecte
- mai mult, prezenta interfetei creator ne ajuta sa folosim polimorfismul la capacitate maxima

10) Cand se utilizeaza fabrica de obiecte?
R: - cand clasa nu poate anticipa ce tipuri de clase de obiecte se fol
- cand clasa isi foloseste subclasele pt a specifica obiectele pe care le creeaza
- cand parametrii sunt trimisi catre fabrica si specifica care clasa trebuie intoarsa

11) Explicati fabrica de fabrici de obiecte (fabrica abstracta)
R: - declara o interfata pentru fabrici care intorc mai multe obiecte (grupate pe familii)

12) Ce este singleton si unde se utilizeaza?
R:

13) Explicati modelul Builder.
R: - atunci cand un obiect nu poate fi creat intr-un singur pas (serializarea unui obiect concret)
- evitarea crearii mai multor constructori pt acelasi tip de obiect
- se pot adauga noi reprezentari pt obiect prin simpla adaugare a unui nou creator.

14) Cand se utilizeaza modelul Builder?
R:

15) Explicati modelul Prototype.
R: Prototype e un blueprint pe baza caruia se creeaza alte obiecte.

16) Cand se utilizeaza modelul Prototip?
R: Cand nu stim dinainte (din timp) detaliile obiectului pe care vrem sa-l cream.
- cand procesul de creare al unui obiect e costisitor (de ex din pct de vedere al timpului)

- clonarea obiectelor si adaugarea de noi feature-uri

Avantaje:
- reduce complexitatea obiectelor create
- adaugarea de noi obiecte in timpul executiei

17) Ce sunt modelele structurale?
R: Cum sunt formate clasele si obiectele pentru a forma structuri mai mari

18) Explicati modelul Adapter.
R: Nu se poate schimba o interfata sau nu e eficienta modificarea ei (librarie de ex)
-> o adaptam ca sa o putem folosi

- cand un sistem trb sa ofere posibilitatea sa ofere o implementare straina peste o interfata cunoscuta

19) Modelul Bridge (pod)
-

20) Fatada -> permite arhitectura inchisa (interfata de nivel inalt care ascunde complexitatea implementarii sistemului -> ofera o interfata mai simpla pt client)

21) Arhitectura deschisa -> orice sistem din layerul dealer management system poate apela orice
clasa din layerul de persistenta (database)

22) Lant de responsabilitati -> se plimba un document pe la mai multe departamente
- pasarea responsabilitatii de a efectua o functie catre alt obiect (care stie sa faca acel lucru)

23) Modelul automatului finit de stari (automat de cafea)

Modelul de stare -> cel mai utilizat in automatele cu stari finite
- abordarea clasica nu respecta principiul solid open-closed
- abordarea smart permite implementarea metodelor de catre fiecare stare nou adaugata (nu e nevoie sa modificam nimic in clasa Snail)

################################## CURS 9
##

1) Explicati modelul Bridge.
R: Bridge este un model de proiectare structural care decupleaza abstractizarea de implementare
asa incat cele doua sa poata variaindependent

2) Cand utilizam modelul Bridge?
R: Atunci cand implementarea unei ierarhii de clase ar putea deveni prea complicata

(ex f multe clase si relatii de mostenire), e util sa folosim modelul bridge pt a separa o anumita dimensiune intr-o ierarhie separata si sa folosim compozitia.
(Ex: avem o clasa abstracta Shape din care mostenim cerc si patrat. Daca vrem sa adaugam
inca o dimensiune pt culoare (cerc rosu, cerc albastru, patrat rosu, patrat albastru), adaugarea
de noi forme sau culori ar putea complica f mult ierarhia de clase.)
Solutia specifica Bridge este sa separam culorile intr-o ierarhie de clase separate, iar ierarhia Shape sa foloseasca obiecte de tip Color prin compozitie

3) Explicati modelul Composite.
R: Ne permite sa formam arbori intr-un mod similar cu sistemul de fisiere
- directoarele ar fi obiectele compuse (compozite)
- fisierele ar fi obiecte primitive (simple)

4) Cand utilizam modelul Composite?
R: E util sa folosim acest sablon de proiectare atunci cand modelul de baza al unei aplicatii poate fi reprezentat sub forma unui arbore
- mai concret, il folosim cand vrem sa implementam o o structura arborescenta de obiecte, in care fiecare obiect e tratat in mod similar (chiar daca e compus sau simplu)

5) Explicati modelul Facade?
R: Facade e un model structural care ofera o interfata simplificata a unei librarii, a unui framework sau a unei colectii complexe de clase.

6) Cand utilizam modelul Facade?
R: - Folosim Facade atunci cand vrem sa oferim o interfata simpla, directa pentru un sistem complex.
Practic: cand construiesti o biblioteca (de ex), vrei sa ii oferi clientului "functiile" pe care le folosesti.
- Nu vrei sa-i arati functiile pe care le folosesti doar tu, intrinsec. Il arati doar ce poate reutiliza el.

7) Explicati modelul Decorator.
R: Decorator e un model de proiectare structural care permite atasarea de noi comportamente
(functii) obiectelor prin plasarea acestora in niste wrappere (ambalaje) care
adauga functionalitati)

8) Cand utilizam modelul Decorator?
R: Utilizam modelul Decorator atunci cand vrem sa adaugam functionalitati noi unor obiecte la runtime.
- De asemenea, utilizam acest model atunci cand extinderea unui obiect prin mostenire nu e posibila (sau complica prea mult sistemul)

9) Explicati modelul Proxy. (proxy = un intermediar)
R: Proxy e un model de proiectare structural care ne ofera un substituent pentru un anumit obiect.

Acest "intermediar" are acces la obiectul original si ne permite sa executam anumite actiuni inainte sau dupa ce o anumita cerere ajunge la obiectul original.

10) Cand folosim Proxy?
R: - Ex aplicatii de control parental: copilul vrea sa intre pe un URL (e trimis URL-ul ca si cerere unui proxy), iar daca
acesta este permis, atunci copilul va putea intra pe site. Daca nu, va primi un mesaj "Nu ai voie!".
- Alt exemplu: atunci cand vrem sa facem caching (cand vrem ca anumite cereri ale clientilor
sa fie salvate in cache pt a putea fi accesate mult mai usor ulterior).

11) Explicati modelul Flyweight.
R: Flyweight e un model de proiectare structural care ne permite sa salvam mai multe obiecte in
memorie prin pastrarea unor parti comune mai multor obiecte in loc sa punem obiectele intregi.

12) Ce este un model comportamental? (Behavioral)
R: Modelele comportamentale sunt responsabile de interactiunile dintre obiecte.

13) Explicati modelul Chain of Responsibility.
R: Chain of Responsibility e un model de proiectare comportamental care ne permite sa trimitem
o cerere printr-un lant de handlere. Cand primeste cererea, un anumit handler decide daca poate sa proceseze acea cerere. Daca nu poate, ea (cererea) va fi trimisa urmatorului handler din lant/lista.

14) Cand utilizam un lant de responsabilitati?
R: - Utilizam acest model atunci cand programul trebuie sa proceseze mai multe cereri, dar tipul si
ordinea acestora este necunoscuta in prealabil.
- De asemenea, putem folosi acest model atunci cand este necesar ca unele handlere sa se
executa intr-o ordine particulara.

15) Explicati modelul Observer.
R: Observer e un model de proiectare comportamental care ne permite sa definim un mecanism
de "abonare" prin intermediul caruia sa notificam mai multe obiecte (abonati, observatori) in legatura cu modificarile care au loc asupra subiectului (obiectului pe care il observa)

16) Cand folosim modelul Observer?
R: Folosim aceset model atunci cand schimbarile asupra unui obiect necesita modificarea altor
obiecte, iar setul de obiecte care trebuie modificate este necunoscut in prealabil.

17) Explicati modelul automatului finit.

R: Are la baza modelul State si se foloseste atunci cand un obiect se comporta diferit
in functie de starea in care se afla (ex automatul de cafea)

18) Explicati modelul Visitor.
R: Visitor e un model de proiectare comportamental care ne permite sa separam algoritmii de
obiectele asupra carora opereaza.

19) Explicati modelul Command.
R: Folosim Command atunci cand vrem sa programam (schedule) anumite executia anumitor
actiuni sau dorim sa le executam separat.

20) Explicati modelul Memento.
R: Undo/redo

21) Explicati modelul Iterator.
R: Gestioneaza parcurgerea unei colectii de elemente.

22) Explicati modelul Strategy.
R: Se schimba obiectul in fct de strategie abordata (Ex. o parcare cu plata care are pret diferit
de stationare pt zi si pt noapte)

23) Explicati modelul Mediator.
R: Un model care are ca scop reducerea complexitatii.
- intermediar virtual (intr-un software de editare video, se incarca un obiect video de dimensiuni reduse - comprimat)

CURS 10
#

1) Ce este calculul paralel?
R: Permite executarea mai multor programe in acelasi timp (desfasurarea simultana a mai multor procese)
- nr programe/procese = nr procesoare (avand in vedere ca ale noastre computere au maxim 8 procesoare, nu prea se aplica paralelismul, ci concurenta)

2) Ce este concurenta? (pseudoparalelism)
R: Concurenta = paralelism simulat (practic, mai multe procese au nevoie de o resursa comuna si concureaza pt a obtine acces la ea)
- nr de entitati care efectueaza ceva > nr procesoare

3) Ce este concurenta la nivel de date?
R: Se folosesc date (variabile) comune.

- Pot aparea probleme de coerenta: Daca un thread scrie o noua valoare intr-o variabila, iar altul citeste variabila respectiva, nu se stie sigur care valoare va fi citita (cea veche sau cea noua)

4) Ce este paralelismul la nivelul datelor?
R: Nu exista probleme in asigurarea coerentei. Se folosesc seturi de date separate (sau se impart datele initiale in bucati si se lucreaza separat -> fiecare proces/thread cu bucata lui)

5) Ce este o corutina?
R: Ca si concept, o corutina este similara unui thread, adica poate contine un bloc de instructiuni
care se executa concurent cu restul codului. Totusi, corutinele sunt mai light-weight si nu sunt
legate de un anumit thread. Ele pot porni in cadrul unui thread si se pot termina in altul.

6) Explicati ciclul de viata al unei corutine.
R: stored -> on-stack -> running -> finished
- stored -> start minimal sub forma de obiecte copiate pe stiva
- on-stack -> datele corutinei au fost introduse in stiva de lucru, dar corutina nu a fost lansata in executie
- running -> o corutina intra in executie
- finished -> unde se ajunge cu terminarea corutinei respective

7) La ce se refera distrugerea la ordin?
R:

8) Ce face cuvantul cheie "suspend"?
R: Opreste un thread ca sa poata incepe alta corutina (de ex ca in cazul Alg. de planificare Round-Robin)
-> precizam faptul ca o functie poate fi intrerupta in interiorul ei pt a permite alteia sa se execute

9) Ce face instructiunea "run blocking"?
R:

10) Ce scope-uri exista pt corutine?
R:

11) Ce este un thread local?
R:

12) Cum se poate asigura coerenta (integritatea) datelor?
R: Prin sincronizarea corutinelor?? (Wild Guess)

13) Ce face instructiunea "async"?
R:

14) Cum se poate crea un thread in Kotlin?

R:

15) Ce tipuri de dispatcheri exista?
R:

16) La ce se refera reflexia si adnotarea in Kotlin?
R:

17) Ce este excluziunea mutuala?
R:

18) Ce este interblocarea (deadlock)?
R: Un proces asteapta ceva ce a blocat (ocupat, folosit) celalalt si viceversa.
(LIVELOCK -> problema filosofilor -> spre deosebire de deadlock, nu se blocheaza, ci se incearca mereu)

19) Diferenta intre thread-uri simple si thread-uri locale.
R:

20) Metode de wait si notify.
R:

21) Memorizarea (mecanism de caching) in contextul paralelismului.
R:

22) Ceva cuvinte cheie... (fold, ...)

23) Modalitati de sincronizare
R:

24) Care e diferenta intre memoria comuna si cea cu transfer de mesaje?
R: In memoria cu transfer de mesaje, datele sunt transmise catre thread-uri prin intermediul unor cozi pt a evita aparitia problemelor
R:

25) Relatia intre corutine si threaduri.
R: Corutinele folosesc mult mai putine resurse si sunt foarte lightweight in comparatie cu thread-urile (putem lansa si 100k corutine fara sa avem probleme)
Corutinele pot sa-si inceapa executia pe un thread si sa continue pe alt thread.

26) Cum pot mapa corutine pe threaduri?
R:

27) Relatia intre corutine si JVM?
R: Corutinele pot fi lansate in cadrul unui thread si sunt active atat timp cat thread-ul este activ.

28) Ce este un actor? (in contextul corutinelor)

R: Un actor e format din 3 lucruri (o corutina, un canal de comunicatie si o stare interna)

CURS 11
##

1) Ce stiti despre jurnalizare/logging?
R: Scrierea unor date/informatii/rezultate/ce se intampla intr-o aplic. intr-un singur loc (centralizare a informatiei)
-> uneori, informatiile sunt centralizate pe alta masina.

2) Ce ne ofera modulul "threading" din python?
R: O librarie care ne permite sa cream thread-uri

3) Care biblioteca din Python ofera cea mai buna accelerare?
R: multiprocessing

4) Cum se determina firul de executie curent si cum il folosim intr-o subclasa?
R: threading.currentThread()

5) La ce se refera asigurarea coerentei datelor?
R: Nu se rezuma la chestii de tip excluziune mutuala
- se refera si la cum garantam propagarea unei copii a unei variabile comune, folosita de diverse thread-uri
(nu reuseste sa ajunga in suficient timp)

6) Ce este un lock()?
R: este un obiect util in sincronizarea thread-urilor sau proceselor. Functioneaza ca un mutex
(permite accesul unui singur thread/proces in zona critica la un moment dat)

7) Ce este un rlock()?
R: Reentrant lock. Similar cu lock, doar ca poate fi folosit de mai multe ori de acelasi thread

8) Ce este semaforul?
R: Tip de date abstract(int) -> O modalitate de sincronizare a accesului concurent la o resursa partajata (comuna) de catre
mai multe procese/thread-uri (putem permite accesul a n procese/thread-uri la zona critica la un moment dat)

9) Ce este un thread cu conditie?
R: Conditia are in spate un mutex (condition.acquire() e un fel de lock() ; condition.release() e un fel de unlock())
- conditie.notify() -> notifica orice alt thread care asteapta sa intre in zona critica (ii spune ca poate intra)
Conditia e utila pt a comunica intre thread-uri.

10) Ce este un thread cu evenimente?

R:

11) Modulul multiprocessing.
R: Librarie care ne permite sa cream procese. Permite folosirea mai multor procesoare de pe masina (nu se limiteaza la unul singur, cum fac thread-urile).
Din acest motiv, multiprocessing ofera cea mai buna accelerare
Explicatie: Practic, thread-urile ruleaza in cadrul aceluiasi proces, iar un proces ruleaza pe un singur core. Pt a folosi
mai multe core-uri, trebuie sa lansam in executie mai multe procese.

12) Cum gestionam starea curenta a unui proces?
R:

13) Cum realizam sincronizarea proceselor?
R: - cu semafoare (excluziune mutuala de ex)
- synchronized
- AtomicInteger()

14) Ce este si cum functioneaza un eveniment?
R:

15) Care e rolul unui obiect de tip condition?
R: Obiectele de tip Condition sunt utile pentru comunicarea intre procese/thread-uri.
Poate semnala faptul ca un thread trebuie sa astepte sau ca poate sa-si continue executia.

16) Ce este o bariera?
R:

17) Ce este interblocarea?
R: Se refera la momentul in care 2 thread-uri asteapta un "input" de la celalalt pt a putea sa-si continue executia, dar niciunul nu poate sa-i ofere celuilalt acest raspuns. Practic, cele doua thread-uri depind unul de celalalt (unul il blocheaza pe celalalt si viceversa)
In alte cuvinte, daca resursele necesare pt deblocarea unui proces sunt folosite de un alt proces blocat, atunci avem de a face cu o situatie de interblocare.

################################ CURS 12
##

1) Dati exemple de transformari intre spatii.
R:

2) Ce este banda Moebius?
R: Este un model de suprafață cu o singură față și o singură margine. Banda are proprietatea matematică de a fi neorientabilă

3) Care este ipoteza lui Church?
R: Fiecare functie poate fi real calculata in mod recursiv.

- Aceasta maniera de a apela functii in mod recursiv e specifica CALCULULUI FUNCTIONAL
Orice functie care e calculabila poate fi scrisa recursiv. (tot apeluri de functii)

4) Care este ipoteza lui Turing?
R: afirmă că o funcție pe numerele naturale poate fi calculată printr-o metodă eficientă dacă și numai dacă este calculabilă de o mașină Turing ??

4') Ce clasa de obiecte matematice descriu aceste doua formulari?
R:

5) Ce este o functie anonima?
R: O functie fara nume (definesc corpul functiei, dar nu ii dau un nume)
Ex: val capitalize = { str: String -> str.capitalize() }
- ce parametrii ar avea fct - ce ar returna functia?

Variabila capitalize poate fi folosita ca o functie (care primeste param string si returneaza ce e returnat in dreapta sagetii)

6) Ce este lambda?
R: Functiile lambda sunt functii pe care le putea crea fara sa le asignam un nume explicit
- Lambda a fost introdus de Church in 1930 si definea notiunea de functie gandita de Church. (Functie anonima)
- Functiile lambda sunt simple transformari (ca la matematica). De ex, transforma x si y in x+y.

7) Care sunt limitarile Java in cazul calculului functional?
R: Java a fost gandit ca un limbaj orientat obiect. Nu are f mult suport pt lambda.

8) Cum se poate utiliza o functie ca o proprietate?
R:

9) Ce tip de date au functiile lambda in Kotlin?
R:

10) Dati exemplu de o functie de nivel superior in Kotlin?
R: O fct de nivel superior este o functie primeste functii ca parametri sau returneaza o functie

11) Ce sunt efectele laterale?
R: O functie pura nu are efecte laterale.
Efectele laterale se refera la faptul ca o functie poate modifica variabile din exteriorul clasei/functiei. (la modul general, din exteriorul scope-ului unde a fost gandita)
- Un efect lateral apare atunci cand o functie modifica valoarea unei variabile din afara scope-ului in care a fost gandita functia.

12) Ce este o functie pura?

R: Valoarea returnata a functiei depinde strict de parametrii (argumentele) primite. Nu are efecte in exterior. (nu are efecte laterale)
- nici nu apeleaza in interiorul ei alta functie care ar putea genera efecte laterale

13) Ce face cuvantul cheie vararg?
R: Ofera posibilitatea de a lucra cu liste variabile de argumente pt o functie.

14) Explicati parametrii alias.
R: typealias Kg = Double (un fel de typedef)
- Sau: Intr-un constructor scriem: ObjConstructor(name = "Andrei", age = 19, ...) in loc de ObjConstructor("Andrei", 19, ...);

15) Ce este o functie de extensie?
R: Scriem noi functii pt o clasa, dar fara sa extindem clasa
Am extins functionalitatea clasei fara sa derivam clasa

16) Care este diferenta intre functia unei clase, functia unei clasa derivate si o functie de extensie a unei clase?
R:

17) Ce este un dispatch receiver?
R: Instanta unei clase care are functie de extensie se numeste dispatch receiver.

18) Cand exista posibilitatea unui conflict de nume in utilizarea unor functii de extensie?
R:

19) ---

20) Ce sunt functiile infix?
R: Ajuta codul sa arate mai natural (mai aproape de matematica).

21) La ce este buna functia "map"?
R: Functia "map" aplica acelasi algoritm pt toate elementele dintr-o colectie si returneaza colectia rezultata
- Dpdv matematic, aplicam o transformare asupra spatiului A care ne duce in spatiul B

22) Cand utilizam functia "filter"?
R: O utilizam atunci cand avem un set de elemente si vrem sa le filtram dupa o anumita conditie.

23) Ce face functia "flatmap"?
R: Am valori pe mai multe dimensiuni si vreau sa fac proiectie pe o singura dimensiune.
- Ex: ((1,2), (3,4,5), (6,7)) -> (1,2,3,4,5,6,7) - ca si cum am elimina parantezele -> dispar tuplele, raman doar elementele intr-o singura dimensiune

24) Ce rol au functiile "drop" si "take"?
R: Drop -> extragere submultimi dintr-o multime.

25) Ce este functorul?
R: Functorul = o clasa ce defineste o modalitate de a transforma datele (o clasa care redefineste functia "map")

26) Ce sunt functiile Curry?
R: Transformarea unei functii cu mai multe argumente in mai multe functii cu un singur argument.
- Aplicarea functiilor partiale (argumentele sunt transmise secvential)

########################## CURS 13
##3

1) Ce este calculul Lambda?
R: Este un model matematic formal in care folosim functii cu parametri, fara nume, pt calculul unor expresii.

2) Cum s-ar putea rescrie un if cu un lambda utilizand doua obiecte?
R:
```
true = lambda x, y: x
false = lambda x,y: y
if = lambda p, x, y: p(x,y)
```

3) La ce se refera *args si **kwargs?
R: Permit introducerea unui nr variabil de argumente la o functie
```
args   - tupla
        kwargs - dictionar
```

4) La ce se refera functiile de prim nivel?
R: In python totul este obiect (chiar si functiile).
Functiile de prim nivel sunt ca niste obiecte cu atribute, pe care le putem inspecta
- ne permit sa aplicam mecanismul lui church (functiile sunt obiecte)

5) La ce se refera datele imutabile?
R: Datele imutabile se refera la datele care nu-si pot modifica continutul

6) La ce se refera functiile pure?
R: Functiile pure sunt cele fara efecte laterale.
- trebuie sa avem grija la variabile globale
- sa nu facem modificari in corpul functiei respective

7) La ce se refera abordarea impacheteaza - proceseaza - despacheteaza?
R:

8) Cum se realizeaza evaluarea la cerere?
R:

9) Ce este un generator recursiv?

R: yield from

10) Pentru ce utilizam modulul iterTools?
R: Pt a folosi functii care genereaza si proceseaza serii si secvente de numere (pt partea hardware sunt f bune deoarece se pot genera repede si usor semnale)

11) Cand utilizam iteratorii infiniti?
R: Sunt utili pt generarea semnalelor (sinus, cosinus etc.)

12) Cum se calculeaza eroarea prin acumulare?
R: Se aduna putin cu putin.

13) Care-i diferenta intre functiile "cycle" si "accumulate"?
R: cycle -> repeta argumentele la infinit
accumulate -> efectueaza o operatie asupra fiecarui argument

14) Cum se poate utiliza modulul iterTools pt a crea functii de ordin 2?
R:

15) Cand se utilizeaza functie "tee"?
R:

16) Cand utilizam operatorii "any" si "all"?
R: any -> macar una dintre "ele" sa respecte o anumita conditie (conditii cu || intre ele) -> returneaza true daca macar o conditie e true
all -> toate dintre "ele" trb sa respecte o anumita conditie (conditii cu && intre ele) -> returneaza true daca toate conditiile sunt true

17) Explicati data flatten.
R: Proiectarea unui set de date reprezentat in mai multe dimensiuni pe o singura dimensiune. (De ex lista de tuple)

18) Ce este scurtcircuitul (nu ala electric)?
R: De ex intr-un if in care avem mai multe conditii cu "OR", daca prima cond e adevarata atunci nu mai trb verificate si celelalte -> scurtcircuit

19) Cum putem evalua deciziile in calculul functional?
R:

20) Ce este un "closure"?
R: Functiile in python sunt referinte catre obiecte. Deci ele pot suporta si apeluri de alte obiecte???
-inchiderile in python retin si contextul in care au fost incheiate

21) Care operatori pe colectii sunt mai rapizi si in ce domeniu?
R:

22) Care este diferenta intre un decorator in stil macro si unul in stil OOP?

R: - stil macro (pe functie)
  - still OOP (pe clasa)

23) Cum putem implementa un state machine (FSM)?
R: