Intrebari Curs 13

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

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.

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:

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: - Utilizam lambda cand avem expresii destul de simple pt care nu merita sa cream o functie

- mapReduce are 2 etape -> mapare si reducere (exemplificam pe exemplu din lab)

1) mapare -> se elimina semnele de punctuatie si se fac litere mici, se sparge textul dupa spatiu si se fac perechi de genul

- ("ana", 1), ("are", 1), ("mere", 1), ...

2) shuffle & sorting (etapa intermediara) -> sorteaza

3) reducere -> suma valorilor pt elementele cu aceeasi cheie

In urma acestor operatiuni a rezultat practic un Word Counter.

Intrebari Curs 12

1.Dati exemple de transformari intre spatii

R:

2. Ce este banda "moebils"?

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 care poate fi calculata poate fi scrisa recursiv

4. Care e ipoteza lui Turing?

R:

5. Ce este o functie anonima?

R: Este o functie fara nume

6. Ce e lambda?

R: Functii pe care le putem crea fara a le asigna un nume explicit

7. Care sunt limitarile Java in cazul calculului functional?

R:

8. Cum se poate utiliza o functie ca o proprietate?

R:

9. Ce tip de date au functiile lambda in kotlin?

R:

10. Dati un exemplu de o functie de nivel superior in kotlin

R: O functie care accepta parametri si poate returna o alta functie fold

11. Ce este un efect lateral?

R: Este acel efect ce consta in modificarea variabilelor existente

in exteriorul blocului functii

12. Ce este o functie pura?

R: O functie care nu are efecte laterale(nu schimba variabilele din exteriorul ei)

13. Ce face cuvantul cheie vararg?

R: Permite primirea unui numar variabil de argumente

14. Explicati parametrii alias

R: Ca un typedef, redenumim tipuri existente

15. Ce este o functie extensie?

R: Ce este o functie ce extinde o clasa, ca si cum am face o noua clasa la care mai adaugam o functie

16. Care e diferenta dintre functia unei clase, functia supraincarcata intr-o clasa derivata

si functia extensie acelei clase?

R:

17. Ce este un dispatcher receiver?

R:

18. Cand exista posibilitatea unui conflict de nume in utilizarea functiilor de extensie?

R:

19. Ce sunt functiile de extensie pt obiecte?

R:

20. Ce sunt functiile infix?

R:

21. La ce este buna functia map?

R: Returneaza o lista ce contine rezultatele aplicatii unei transformari listei initiale

22. Cand utilizam functia filter?

R: Cand dorim sa selectam doar anumite elemente dintr-un container

23. Ce face functia flatMap?

R: Imi "aplatizeaza", de exemplu, mai multe liste intr-o lista. Transforma dintr-o lista de liste intr-o lista

24. Ce rol au functiile drop si take?

R:

25. Ce este functorul?

R: Este o clasa ce poate fi apelata

26. Ce sunt functiile curry?

R:

Intrebari 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: Cuvantul cheie suspend practic ne spune ca acea functie poate fi intrerupta oricand, dar poate fi si reluata oricand. Lucrul asta este util daca vrem sa facem o planificare de genul Round - Robin

9) Ce face instructiunea "run blocking"?

R: Incepe o noua corutina si cumva desparte tot ce este in afara corutinei si ce este in interiorul corutinei. Practic blocheaza thread-ul care executa codul din afara acestei corutine si il deblocheaza cand s-a terminat de facut tot ce era in interiorul corutinei

10) Ce scope-uri exista pt corutine?

R: Global, LifeCycle, ViewModel

11) Ce este un thread local?

R: Thread Local este o clasa ce se foloseste la a declara variabile ce se pot citi sau scrie de acelasi thread. De exemplu, avem 2 thread-uri care acceseaza acelasi cod care contine o referinta spre o variabila de tip thread local, atunci niciunul din thread-uri nu va vedea modificarile facute de celalalt thread la acea variabila.

12) Cum se poate asigura coerenta (integritatea) datelor?

R: Prin sincronizarea corutinelor, prin lock-uri, prin variabile atomice

13) Ce face instructiunea "async"?

R: Incepe o noua corutina, returneaza o valoare de tip Deffered, cumva promite ca va primi rezultatul mai tarziu. Folosim await pentru a-i primi valoarea unei variabile de tip Deffered

14) Cum se poate crea un thread in Kotlin?

R: Mostenind clasa Thread sau implementand interfata Runnable sau apeland functia thread() din kotlin

15) Ce tipuri de dispatcheri exista?

R: Main Dispatcher, IO Dispatcher, Default Dispatcher, Unconfined Dispatcher

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

R:

17) Ce este excluziunea mutuala?

R: Se refera la faptul ca la un moment dat de timp doar o corutina executa o bucata de cod. In Kotlin excluziunea mutuala se implementeaza in asa fel incat nu sunt blocante: adica daca un proces are acces acum la "lacat" pt o bucata de cod atunci el asteapta ca alte procese sa-si faca treaba iar el va primi acel "lacat" cand va fi liber ca apoi sa ruleze bucata lui de cod

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: Wait - opreste un thread pana functia notify ii spune sa revina in executie

Notify - trezeste din somn un anumit thread

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

R:

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

23) Modalitati de sincronizare

R: Variabile atomice, lock-uri, actori

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: In cazul memoriei comune fiecare thread ia datele din acelasi loc din memorie, pot aparea probleme

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)

26) Cum pot mapa corutine pe threaduri?

R: Corutinele isi pot porni executia pe un thread si pot termina pe orice altul

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)

Intrebari Curs 9

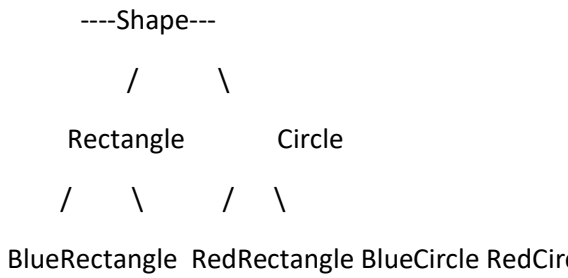
TutorialsPoint

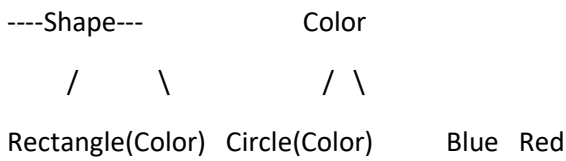
1) Explicati modelul Bridge.

R: Modelul Bridge este un model structural care iti permite sa separi o clasa mare in doua ierarhii separate, abstractizare si implementare care pot fi dezvoltate independent una de cealalta. Practic decuplam abstractizarea de implementare.

2) Cand utilizam modelul Bridge?

R: Cand avem ierarhii preferam sa folosim acest model





3) Explicati modelul Composite.

R: Modelul composite este un model de partitionare si descrie un grup de obiecte care sunt tratate ca si o singura instanta a unui acelasi tip de obiect

4) Cand utilizam modelul Composite?

R: Cand vedem ca mai multe tipuri de obiecte sunt tratate in acelasi fel, repetandu-se codul pentru fiecare dintre ele, atunci e o idee buna sa folosim modelul composite, tratandu-le pe toate omogen.

5) Explicati modelul Facade?

R: Modelul Facade ascunde complexitatea sistemului si pune la dispozitie o interfata utilizatorului ce poate fi folosita pentru a utiliza functionalitatile sistemului

6) Cand utilizam modelul Facade?

R: 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. II arati doar ce poate reutiliza el.

7) Explicati modelul Decorator.

R: Modelul Decorator este un model structural ce permite adaugarea de noi functionalitati unei clase, fara a-i afecta functionalitatile anterioare si modul cum relationeaza cu alte clase.

8) Cand utilizam modelul Decorator?

R: Ex. un context Manager (in python, context manager poate fi folosit ca un decorator)

9) Explicati modelul Proxy. (proxy = un intermediar)

R: ((Intermediarul proceseaza cererea noastra si o modifica sub o anumita forma pt a ne oferi rezultatul))

- Practic, proxy proceseaza o anumita cerere, un task pe care i-l dam.

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!".

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: Se ocupa de descrierea modului de interactiune intre clase

Descrie fluxul de control al unei aplicatii

13) Explicati modelul Chain of Responsibility.

R: Clientul trimite o cerere, iar aceasta este trimisa de la clasa la clasa pana se gaseste una care o poate procesa.

14) Cand utilizam un lant de responsabilitati?

R: De exemplu cand dorim tratarea unei cereri intr-o anumita ordine.

De exemplu, daca avem un request pe web, la tratarea lui il putem trece prin mai multe middleware-uri

de exemplu un middleware de securitate, care sa verifice daca persoana este logata, iar abia apoi ii putem trimite informatiile despre contul bancar, de exemplu

15) Explicati modelul Observer.

R: Observer este un model de comportament in care avem un Subject si mai multi observatori, iar cand se schimba ceva intr-o instanta a unui obiect de tip Subject toti observatorii lui trebuie anuntati, fiinda acestia depind de subject

16) Cand folosim modelul Observer?

R: Cand avem o relatie one to many iar componentele many depind de componenta principala.

17) Explicati modelul automatului finit.

R: Are la baza modelul State.

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: Gestioneaza realizarea unei actiuni

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.

Intrebari Curs 8

1) Ce este un model de proiectare (un design pattern)?

R: Modele de proiectare

2) Cum a aparut termenul de model de proiectare (termenul de design pattern)?

R: -> Cristopher Alexander

3) Cum a evoluat conceptul de design pattern?

R: 1987 - Cunningham si Beck - limbaj

1990 - Gasca celor patru - catalog

1995 - Gasca celor patru - carte

4) Explicati modelele Real si Zuligoven ??? (not sure)

R: -model conceptual:descrie in termeni si concepte domeniul aplicatiei

-model de proiectare:descrie proiectarea software folosind constructii software

-model de programare:se fol. constructii de limbaj pentru a descrie forme

5) Unde sunt utile modelele de proiectare?

R: Modelul aplicatiei

GoF

OOP + ADT

6) Care sunt elementele unui model de proiectare?

R: -numele:trb. ales a.i. sa descrie pe scurt problema de proiectare

-problema:se descriu cazurile in care se aplica acest model

-solutia:descrie elem. care compun proiectul, relatiile si colaborarile dintre ele

-consecintele:rez. obtinute si compromisurile care trb facute cand se aplica modelul

7) Ce este o arhitectura inchisa?

R:

8) Ce sunt modelele creationale?

R:

9) Explicati fabrica de obiecte (factory method).

R:

10) Cand se utilizeaza fabrica de obiecte?

R: Clasa nu poate anticipa ce tipuri de obiecte trebuie generate

11) Explicati fabrica de fabrici de obiecte.

R:

12) Ce este singleton si unde se utilizeaza?

R:

13) Explicati modelul Builder.

R: Permite crearea de obiecte complexe pornind de la unele simple

14) Cand se utilizeaza modelul Builder?

R: Obiectul nu poate fi creat intr-un singur pas

Evitarea crearii de prea multi constructori pt acelasi obiect

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.

17) Ce sunt modelele structurale?

R: Modalitatea de combinare a claselor si obiectelor

18) Explicati modelul Adapter.

R: Dorim folosirea unei librarii, dar interfata pe care vrem sa o folosim

nu este

19) Modelul POD

-Decuplam abstractizarea de modelul ei

Intrebari Curs 7

1. Ce sunt functiile parametrizate?

R: Functii bazate pe generice(argumentele sunt parametrizabile)

Any(Kotlin) = Object(Java)

Unit(Kotlin) = void(Java)

2. Ce sunt tipurile parametrizate?

R: Ca si containerele din C++, de ex List<Int>

parametrii au un tip(se specifica intre paranteze ascutite)

3. La ce se refera polimorfismul limitat superior?

R: Nu putem duce orice tip de date in orice tip de date

De ex pt comparable: se limiteaza la subgrupul asteptat care contine elemente ce pot fi comparate

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: Tipuri de date paramatrizabile(au un parametru <T> care poate fi inlocuit cu Int, Double sau chiar ADT-uri)

ex Class<Int>

6. Modificatorii de tip?

R: out -> poate fi folosit doar ca tip de return

in -> poate fi folosit doar ca parametrii la functii

7. Ce este declaration site variance?

R: abilitatea de a specifica variance annotation la declararea clasei

iesire -> produs(out) -> covarianta

intrare -> consumat(in) -> contravarianta

7'. Ce este variance annotation?

R: un modificator aplicat unui parametru/argument generic, cu scopul de a-i declara varianta. In Kotlin sunt 2 variance annotations: out si in

8. Explicati problema claselor duale producator - consumator

R: Clase care folosesc un parametru covariant out si un parametru contravariant in.

9. Explicati type projection

R: E util atunci cand cineva a declarat o clasa invarianta si eu am nevoie sa fie folosita intr-un mod covariant sau contravariant. Practic specificam tipul de varianta la utilizare

11. Ce sunt functiile generice?

R: Sunt functii ce au si ele la randul lor tipuri parametrice

12. Ce sunt constrangerile generice?

R: Limitarea superioara a constrangerilor

13. Ce este type erasure? (pierderea tipului)

R: La cast type instantele cu tip generic nu stiu de ce tip sunt

14. Ce sunt tipurile de date algebrice?

R: Similar grupurilor din algebra - un set inchis de tipuri si functiii

functiile pot folosi doar anumite tipuri(cu ceil)

15. Ce sunt clasele "sealed"?

R: Clasa a carei functionalitate e restransa la fisierul in care e descrisa.

- contine operatiile asociate

16. Ce sunt colectiile?

R: Niste entitati ce permit tratarea unitara a mai multor elemente ca o multime matematica.

17. Ce operatii ne ofera colectiile lista?

R: add, remove, set, contains, get, isEmpty, size, indexOf

lastIndexOf, removeAt, clear()

18. Ce sunt listele in kotlin?

R: Colectii generice ordonate de elemente

19. Care e diferenta intre list si mutableList ?

R: Se pot adauga si sterge elemente

20. Ce sunt colectiile set?

R: O colectie de obiecte fara duplicate

21. Ce operatii ofera colectiile Set?

R: size, contains, add, remove

containsAll, isEmpty, retainAll

22. Ce este LinkedHashSet?

R: Implementarea unui set folosind un Hash_table

lista dubla inlantuita, hash table

Ma protejeaza de ordinea aleatorie a lui hashSet, pastreaza ordinea

de la inserare

contains = O(1)

size, add = O(1)

23. Ce este un treeSet?

R: TreeSet este o interfata din SortedSet care foloseste un arbore ca metoda de stocare

log(n) - add, remove, contain

24. Ce este o colectie Map?

R: O colectie ce contine perechi de elemente

key -> value

suporta extragerea optimizata

cheile sunt unice

relatii 1 la 1

25. Ce operatii ne confera colectiile de tip Map?

R: containsValue, containsKey, isEmpty, size

keys, value, entries

put, remove, putAll

26. Cum se poate parcurge un map cu un for?

R: for elem in map.keys{

print("[%v]: %v", elem, map[elem])

}

27. Avantaje LinkedHashmap? Avantaje TreeMap?

R: LinkedHashMap = HashMap + LinkedList (lista dublu inlantuita)

TreeMap -> implementare pe arbori red-black

28. Ce este colectia Array?

R: Pentru Backward compatibility

E un container ce retine un nr fix de elemente de un tip dat

29. Care este relatia dintre colectiile Java si colectiile Kotlin?

R: Colectiile Kotlin is mai bune. Se distinge o data dintre mutable si immutable

30. Ce sunt tipurile platforma?

Marcate cu !

Deseori librariile din Java(fiind un limbaj null-unsafe)

au metode ce returneaza un tip! deoarece Kotlin nu-si da seama daca rezultatul e nullable sau nu.

Practic ! spune ca poate fi si nullable si non-nullable, venind dintr-oplatforma care nu are aceste clasificari

Se foloseste in string-uri, punem de exemplu

"%d". format(x)

31. Cand nu este imutabilul garantat in Kotlin?

In kotlin, e garantat de o interfata, dar Kotlin relaxeaza principiile fata de java, deci pot aparea probleme la combinarea celor 2 limbaje.

Practic, atunci cand interoperam cu java(tipuri cochilie).

Mutable - se poate modifica

Immutable - nu se modifica, creeaza o noua colectie cu update-ul facut

Intrebari Curs 6

1.Enumerati domeniile posibile de utilizare python

R: Backend, front end, inteligenta artificiala, securitate, aplicatii web, hardware low level

2. Care sunt paradigmlele de programare ale lui Python?

R: functionala

orientate obiect

imperativa

procedurala

3. Ce tipuri de conversii explicite pt tipurile de date suporta python?

R: integer,string,complex, floating point, long

4. Explicati instructiunea try-except

R: except exception

5. Ce este o exceptie?

R: Este un eveniment care apare in timpul executiei programului si care opreste fluxul normal al instructiunilor. Cand programul gaseste o situatie care nu-i convine trimite o exceptie. Exceptia e un obiect python care reprezinta o eroare

6. La ce este utilizata pass?

R: De exemplu pentru a spune unei functii ca va fi interpretata mai tarziu.

7. Ce diferenta sunt in python fata de C in termeni de operatori simpli?

R: putere, //, membership si identitate la op pe biti

in python nu avem incrementari

8. Ce tipuri de operatori de atribuire sunt suportati in python?

R: =,+=,*=,%=,/= etc

9. Diferenta dintre op de apartenenta si cei de identitate

R: daca un obiect e intr-o secventa

daca au aceeasi referinta

10. Ce standard de caractere este implicit in python?

R: UTF-8

11. De ce se poate executa un cod in interiorul unui sir in python?

R: Deoarece python este interpretat

cu exec si eval

12. Ce este eval?

R: Evalueaza un string ca o expresie python si returneaza rezultatul

13. Care e diferenta dintre o tupla si un dictionar?

R: Dictionarul e mutable, tuple e immutable

14. Care sunt atributele unui obiect de tip fisier?

R: close(),mode(),name(),softspace(),next(),read(),

readline(),seek(),tell(), write()

15. Care e metoda corecta de tratete a op cu fisiere?

R: Deschidem fisierul intr-un try tratem erorile si inchidem la final fisierul.

Daca sunt exceptii le tratam cu except

16. Cum se trateaza apelul unei functii in python?

R: Se opreste executia, se trimit valorile la parametrii formali, functia isi face treaba si returneaza valoare

17. De ce pot intoarce valori multiple in python?

R: Cu tuple, poate sa despacheteze tuple(iterable unpacking)

18. Cum pot simula transferul prin referinta la iesirea din functie in python?

R: tablouri, lista

19. La ce ne folosesc cuvintele cheie utilizate ca parametru in python?

R: ca sa fim mai expliciti

20. Care sunt diferentele dintre C++ si suportul primar de OOP oferit de python?

R: this vs self(se trimite implicit, dar trebuia primit explicit)

constructorul este __init__()

indentare la python

nu avem ;

in python avem conceptul ala Duck typing si astfel exista cumva o interfata

generata automat

21. Cum se poate adauga rapid persistenta in python?

R: fisier, baza de date, shelve

22. De ce as fi nevoit sa construiesc un GUI de la 0 in python?

R: vrei sa modifici comportamentul default al acelui GUI

(de exemplu vrei sa schimbi background-ul si nu iti permite GUI existent)

Intrebari Curs 5

1. Ce este un eveniment?

R: 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

R: Tastatura,Mouse -> OS -> coada evenimente -> programe -> OS -> monitor

3. Cum se gestioneaza dpdv arhitectural un eveniment?

R: 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?

R: Sincron : trimit mesaj, astept confirmare

asincron : trimit toate mesajele, nu mai astept confirmare

mouse tastatura : asincron

5. Care sunt sursele comune pt evenimentele sincrone si asincrone?

R: Mouse-ul si tastatura

6. Cum se trateaza un eveniment dpvd al programatorului?

R: Se trimite la distribuitor care le trimite la un anumit gestionant

Polling (citire secventiala intr-o bucla infinita a tuturor comenzilor dispozitivelor de intrare)

7. Care sunt avantajele procesarii orientate pe evenimente?

R: mai portabile

permit tratarea rapida a erorilor

se pot 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?

R: 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?

R: AppUser - actiuni

Event Object - genereaza evenimente

Event Mapper - inregistreaza gestionarul de evenimente si distribuie

Event Handler - proceseaza evenimentele

10. Ce este dispecerul in EDP?

R: Cel care controleaza evenimentele

Distribuie evenimentele

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

R: What you see is what you get

12. Explicati look-n-feel?

R:

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

R:

14. La ce se refera PIC correlation?

R:

15. Ce este un GUI chat?

R:

16. Explicati MVC-ul

R: 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?

R: Este o biblioteca folosita la creare de software

SimpleDirectMediaPlayer

18. Care sunt modulele SDL si echivalentele lui cu DIVX?

R: 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

R: 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

R:

21. Ce este si unde este necesara organizarea matriceala a entitatilor intr-un GUI python?

R: Fiecare element este pe o linie si o coloana.

22. Explicati maniera arborescenta de creare a meniurilor in tkINTER

R: Creez submeniurile si apoi le asamblez intr-un meniu mare

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

R:

24. Explicati tratarea evenimentelor in Android

R:

25. Cum se pot adauga servicii in Android?

R:

Polling : bucla infinita

- Interrupt-driven : asteapta aparitia unor intreruperi
- Widget : obiecte din cadrul unui GUI orientat obiect
- Sincron : trimit mesaj, astept confirmarea ca primul a ajuns, apoi trimit al doilea mesaj. eu rezerv resurse care saprimeasca 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

Intrebari Curs 4

1. Cum definiti proiectarea aplicatiilor software?

R: Cu ajutorul diagramelor UML

Proiectarea software reprezinta un proces de rezolvare a unor probleme, obiectivul fiind sa se gasesaca si sa se descrie o cale de a implementa necesitatile functionale ale sistemului, tinand cand de constrangerile cleintului

2. Care e diferenta intre must have si nice to have ?

R: produsele principale(must have), cele secundare(nice to have)

3. Definiti o componenta?

R: Entitate fie fie soft fie hard care are un rol bine determinat si poate fi inlocuita cu alta componenta

4. Din ce este alcatuit un sistem daca il analizam din prisma analizei modulare? Subsisteme, componenete si module

R: Subsisteme, componente si module

5. Ce este modelul domeniului?

R: System,susbsysteme implementat de Component, care este definit la nivelul limbajului de catre module si framework

6. Ce este UML(unified modeling language)

R:Este un limbaj grafic adoptat mondial folosit pentru reprezentarea,vizualizarea si documenatarea componentulor unui sistem software de dimeniusni mari

7. Ce este modelul de proiectare si implementare in cascada? AVANTAJ

R: Fiecare nou proces 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 AGILE si WATERFALL?

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 :

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 task-urilor: __,planificarea

R: Harta PERT(care sunt legaturile dintre task-uri),

Planificarea(cum poate fi aceasta realizata in durata de timp rezervata?)

Harta organizationala(care sunt rolurile in proiect?)

11.Eumerati mecanismele centrale ale unei aplicatii OOP

R:

12. Cum se defineste un model dpdv al UML:

R: Un model este o descriere completa a unui sistem dintr o perspectiva particulara

12'. Tipuri de proiectare specifice

R: proiectare arhitecturala

proiectarea claselor

proiectarea interfetei vizuale

proiectarea bazelor de date

proiectarea algoritmilor

proiectarea protocoalelor

13. La ce este buna abordarea bazata pe componenete in proiectare?

R: Componentele pot fi inlocuite ulterior cu unele mai bune, acest lucru fiind posibil datorita nivelului ridicat de incapsulare.

14. Dati exemple de aplicare a principiului cresterii coeziunii

R: Grupeaza ce este apropiat cat mai mult

15. Exemple de aplicare a principiului reducerii cuplarii(cuplare = interdependente in

R: De exemplu: reducem numarul de dependente intre clase, daca fiecare clasa ar fi un nod intr-un graf ne-am dori sa minimizam numarul de arce, sa fie cat mai independente pentru a ne fi usor sa modificam lucrurile ulterior

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?

R: proiecteaza pentru reutilizare, proiecteaza prin reutilizare

18. La ce se refera proiectarea pt flexibilitate?

R:Anticiparea activa a modificarilor pe care un proiect le poate suferi si pregatirea din timp pt acestea

R: Anticiparea schimbarilor ce pot aparea pe parcurs. (reducerea cuplarii + cresterea coeziunii)

19. Cum se gestioneaza problemele de inlocuire sau disparitie a unor componente

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? Programam aplicatia astfel incat sa fie cat mai portabila pe mai multe sisteme de operare

R: Utilizarea de limbaje interpretate(de ex Java), ce nu sunt dependente de arhitectura hardware

21. La ce se refera proiectarea pt testabilitate?

R: Iei masuri pentru usurarea testarii

Masuri precum:

- utilizarea design patternurilor

- utilizarea unor feature-uri care acceota linie de comanda

21') Scopurile proiectarii OOP

R:

-> 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. La ce se refera coeziunea claselor?

R:plasarea metodelor care opereaza pe anumitr date cat mai aproape de datele respective

23. Ce este principiul de raspundere unica?

R:Fiecare clasa ar trebui sa faca o singura chestie

24. Ce este principiul separarii interfetelor?

R:Clientii nu trebuie sa fie obligati sa depinda de interfete de care nu au nevoie

25. Principiul inchis/deschis?

R:Clasele trebuie sa fie deschise la extindere si inchise la modificare

26. Ce este BPP-ul?

R: Bune practici de proiectare (Divide and conquer)

27. Principiul substitutiei Liskov?

R: Clasele derivabile trebuie sa fie substituibile in clasele de baza. Adica clasele derivate trebuie sa aduca functionalotati noi, nu sa le modifice pe cele vechi

28. Principiul dependentei inverse?

R: Modulele de nivel arhitectural superior nu trebuie sa depinde de cele de nivel inferior. Ambele trebuie sa depinda de abstractii, care la randul lor nu depind de implementarile concrete

29. TDD = Test-Driven Development

R:

scrie(un test)

executa(toate testele)

scrie(codul tinta)

executa(testele)

refactorizeaza(codul)

Intrebari Curs 3

1.Ce este descompunere functionala dpdv al oop?

R:Inseamna ca atunci cand proiectam un program sa ne facem un plan, sa distingem niste functionalitati iar apoi sa le modulam si sa le algoritmizam impartindu-i intr-un numar de pasi

Ex: Sa se acceseze descrierea unor forme existente intr-o baza de date apoi sa se afiseze aceste forme

1. Identifica lista de forme in baza de date

2. Deschide lista de forme din baza de date

3. Ordoneaza lista de forme conform cu un set de reguli

4. Afiseaza formelele pe monitor

2.Care sunt principalele probleme ale specificatiilor de la client?

R: Clientul habar n-are ce vrea si da info incomplete, gresite, ii mai vin idei pe parcurs, greseli de comunicare/intelegere

3.Care sunt motivele pt care clientul nu da ce trebuie?

R: Diferente de comunicare, clientul omite informatii care crede el ca sunt evidente desi sunt relevante pentru proiectant, nu se gandeste inainte la toate aplicatiile

4.Ce efect au coeziunea scazuta si cuplarea stransa?

R: coeziune - cat de apropiate sunt operatiile dintr-o metoda

cuplarea - cat de stransa este legatura dintre doua metode

integritate interna(coeziune stransa)

relatii directie, vizibile, flexibile si directe(strans cuplate)

Coeziunea scazuta = operatiile dintr-o metoda sunt strans legate intre ele si o modificare mica se propaga peste tot

5.Care sunt pasii din proiectarea OOP?

R: Izoleaza obiectele din lumea reala, abstractizeaza obiectele, determina responsabilitatea grupului fata de alte grupuri

6.Ce este brainstorming-ul?

R: aruncari spontane de idei, persoanele care participa ar trebui sa stie pt a da idei pertinente

7.Ce pasi implica metodologia de proiectare OOP?

R: brainstorming - pentru a localiza clasele posibile, se considera toate ideile

filtrarea claselor - pt a gasi duplicatele si a elimina pe cele in plus

scenarii - necesare pentru a fi siguri ca s-a inteles colaborarea intre obiecte

algoritmi - sunt proiectati pentru a defini toate actiunile pe care clesele trebuie sa le poata efectua

8.La ce se folosesc fisierele CRC?

R: Pt a mentine informatiile primare despre clase,subclase, superclase, responsabilitati, colaborari, cum relationeaza intre ele clasele

9.Care este flow-ul general pt rez si proiect unei probleme in OOP?

?R: Se grupeaza elementele in clase

10.Cum creeaza Kotlin obiectele asociate unei clase?

?R: Instantiere prin constructori

11.Care este rolul lui this?

R: Ne ajuta sa ne referim la obiectul respectiv(in metodele unei clase de ex)

this -> instanta curenta

cand vrem sa ne referim la instanta externa

12.Care este rolul lui scope?

R:

13.Cum se poate face controlul fluxului de executie ca o expresie?

R: cu if,else, try,catch

14.Ce este NULL? Ce rol are NULL?

?R: Null este un pointer la nimic. Null nu se poate converti la un tip

...

15.Explicati verificarea si conversia de tip in Kotlin

R: Implicita(dar poate fi facuta explicita).

Verificam cu "is" = "instance of"(Java)

Kotlin face cast a unei variabile la ultima verificare de tip

16.Cum se poate utiliza when ca un switch case?

R: In loc de default se pune else; la case se pune doar valoarea pe care sa o ia variabila din when()

17.Cum se poate utiliza when ca o expresie?

?R: Asignam valoare unei variabile cu when

18.Ce este clasa anonima?

R: Este o clasa ce se poate utiliza pt apeluri singulare si nu este recomandata

19.Ce sunt clasele pt gestiunea datelor?

R: Clasele de tip enum si data(pastreaza date)

20.Ce sunt metodele statice in Kotlin?

R: Nu avem in clase metode statice(in Kotlin). La nivel de pachet avem metode statice.

21.Ce sunt obiectele companion si unde se folosesc?

R: Functiile companion care apartin unei clase iau locul metodelor statice, putand fi apelata fara un obiect instatiat

22.Suporta Kotlin mostenirea simpla direct?

R: Mostenirea multipla se simuleaza

Se suporta mostenirea simpla directa. Clasele fara parinte vine de la clasa de baza(object, aia de baza) Pt a face o clasa derivabila avem cuvantul cheie "open"

23.Ce sunt functiile cu expresie unica?

R: Functie ce contine o simpla expresie si folosind inferenta de tip isi da seama pe baza evaluarii expresiei ce tip are de returnat

24.Ce sunt functiile membru?

R: Functii in interiorul unei clase, obiect sau interfata.

25.Cand sunt recomandate functiile locale?

R: Functii mici, definite in interiorul altor functii

Cand iesim din scope-ul in care au fost definite nu le mai putem folosi

Dorim sa ascundem detalii de implementare a unei functii mai mari

Cand nu mai avem nevoie de ele in afara scope-ului

26.Ce sunt functiile top-level?

R: Functiile globale din C++

27.Ce este list cromprehension si cum se poate face in Kotlin?

R: Facem liste pe o linie, e mai rapid, am facut in graba

izoleaza obiecte din lumea reala, abstractizeaza obiectele care au prop si comportamente similare

determina resp grupului dpdv al interactiunii cu alte grupuri

Metodologie POO

brainstorming - pentru a localiza clasele posibile

filtrarea claselor - pt a gasi duplicatele si a elimina pe cele in plus

scenarii - necesare pentru a fi siguri ca s-a inteles colaborarea intre obiecte

algoritmi - sunt proiectati pentru a defini toate actiunile pe care clesele trebuie sa le poata efectua

Intrebari curs 2

studentul la curs se afla in starea s0 plictisit

pauza = 1

studentul tranzitioneaza in stare 2 numit relaxat

pauza = 0

studentul revine la starea s0 plictisit

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)

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

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 valoarea 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: Loggings(trimitere de mesaje), mecanismul 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

link ascuns = dino.zip

lambda calculabila = calculabila conforma modelului turing

mutable types = cand dorim sa ii asociem o noua valoare pur si simplu se schimba(de exemplu lista in python)

immutable types = cand dorim sa ii asociem o noua valoare, de fapt se creeaza un nou obiect(cu alt id) spre care variabila noastra pointeaza

dynamic typing = nu e nevoie sa declari tipul obiectului, isi da el singur seama(ex python: a = 7, isi da seama ca e vorba de un int); fac type-checking la run-time

static typing = trebuie sa declari tipul obiectului la declarare(ex: int a = 7); fac type-checking la compile time

strongly typed = nu face conversii implicite(de exemplu 1 + "2" = eroare)

weakly typed = face conversii implicite(de exemplu in javascript: 1 + "2" = "12")

Intrebari Curs 13

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

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.

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:

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:

- Utilizam lambda cand avem expresii destul de simple pt care nu merita sa cream o functie

- mapReduce are 2 etape -> mapare si reducere (exemplificam pe exemplu din lab)

1) mapare -> se elimina semnele de punctuatie si se fac litere mici, se sparge textul dupa spatiu si se fac perechi de genul

- ("ana", 1), ("are", 1), ("mere", 1), ...

2) shuffle & sorting (etapa intermediara) -> sorteaza

3) reducere -> suma valorilor pt elementele cu aceeasi cheie

In urma acestor operatiuni a rezultat practic un Word Counter.