

JOB INTERVIEW SAVCL- JAVCBALAR



ALIMARDON BOQIJONOV

Savol-1:

HTTP VA HTTPS FARQLARI

HTTP (HyperText Transfer Protocol) va HTTPS (HyperText Transfer Protocol Secure) protokollarining asosiy 10 farqi quyidagicha:

1. Amal qilish: HTTP protokoli veb-sahifalarni yuklash, yuborish va olish uchun ishlataladi, odatda to'plamli veb-muqobil (stateless) tarzda amalga oshiriladi. HTTPS esa veb-sahifalarni xavfsiz tarzda yuklash, yuborish va olish uchun ishlataladi, ma'lumotlarni shifrlab o'tkazadi va kimdirning ma'lumotlarni kuzatishini qiyinchilikka o'tkazadi.
2. Xavfsizlik: HTTP protokolida ma'lumotlar shifrlanmaydi, ya'ni uzatilayotgan so'zlar, ma'lumotlar va talablar otuz ikkilik (plaintext) shaklda yuboriladi. Buning ta'siri bilan, uzatilayotgan ma'lumotlar kuzatuvchi tomonlar tomonidan o'qilishi oson bo'ladi. HTTPS protokoli esa ma'lumotlar shifrlanadi, ya'ni ma'lumotlar shifrlanib yuboriladi va faqat o'zi ushbu ma'lumotlarni o'qish, foydalanish va tahlil qilish imkoniga ega bo'ladi.
3. Portlar: HTTP va HTTPS protokollari bir xizmat sifatida amal qilish uchun biror portni ishlataladi. HTTP protokoli uchun oddiyligi uchun 80-chi port ishlataladi, HTTPS protokoli uchun esa shifrlanganligi uchun 443-chi port ishlataladi.
4. SSL/TLS: HTTPS protokolining asosiy farqlaridan biri shifrlash usullari bo'lib, odatda Transport Layer Security (TLS) yoki avvalgi versiyasi Secure Sockets Layer (SSL) protokolidan foydalaniladi. Bu protokollar orqali ma'lumotlar shifrlanib yuboriladi va foydalanuvchi va server o'rtaсидаги aloqada xavfsizlik ta'minlanadi.
5. Sertifikatlar: HTTPS protokoli orqali foydalanuvchi va server o'rtaсидаги xavfsizlikni ta'minlash uchun sertifikatlar ishlataladi. Sertifikatlar xavfsizlikni tasdiqlovchi asosiy vositadir va ularga serverning identifikatori, shifrlash klyuchi va sertifikat haqidagi ma'lumotlar kiritiladi.

6. URL struktura: HTTP va HTTPS URL larining strukturasida ham farq mavjud. HTTP URL lar "http://" bilan boshlanadi, HTTPS URL lar esa "https://" bilan boshlanadi. Bu bilan birga, HTTPS URL larida yuborish va so'rovlarni shifrlash uchun "TLS" qismi ham mavjud bo'ladi.
7. Kuzatuvchilar: HTTP protokoli orqali yuborilgan ma'lumotlar osonlik bilan kuzatilishi mumkin, masalan, kuzatuvchi vositalar orqali. HTTPS protokoli esa ma'lumotlarni shifrlash sababli, ma'lumotlarni kuzatish qiyinchilikka o'tkazadi.
8. Xosting: Ba'zi hosting xizmat provayderlar HTTPS protokoliga qo'lda qolib, faqat HTTP protokoliga yo'naltirishlar bilan xizmat ko'rsatishadi. Bu holatda, foydalanuvchilar tomonidan HTTP URL lar kiritilsa ham, ularga avtomatik ravishda HTTPS protokoliga o'tish amalga oshiriladi.
9. Sayt ruxsatnomalari: HTTPS protokoli sayt ruxsatnomalari (site certificates) talab qiladi. Sayt egasi ruxsatnoma olishi uchun SSL sertifikatini olishi kerak. Bu sertifikat saytning identifikatorini va shifrlash klyuchini o'z ichiga oladi.
10. Xavfsizlik ta'minoti: HTTPS protokoli yordamida foydalanuvchi va server o'rtaсидаги ma'lumotlar shifrlanadi, shuning uchun ma'lumotlarni o'qish, o'zgartirish va o'rnatish kuzatuvchilar tomonidan qiyinchilikka o'tadi. Shuningdek, HTTPS protokoli orqali identifikatsiyalash va kimdirning saytga bo'lgan to'lojni himoya qilish imkoniyatiga ega bo'ladi.

Ushbu farqlar HTTP va HTTPS protokollari orasidagi asosiy farqlardir. HTTPS protokoli ma'lumotlar shifrlanishi sababli xavfsizlikni ta'minlaydi, shuningdek, sertifikatlar orqali serverni tasdiqlash imkonini beradi. Bunda HTTP protokoli bilan taqqoslanib, ma'lumotlar to'plamlarini himoya qilish uchun HTTPS protokolini ishlatalish tavsiya etiladi.



HTTP VS HTTPS

Savol-2:

OOP nima?

OOP (Object-Oriented Programming) yoki obyektoriyentlangan dasturlash, dasturlashning bir paradigmasidir. Bu usulda, dasturlashda ishlatiladigan obyektlar va ularning bir-biriga munosabatlari asosiy qismni tashkil etadi.

OOP paradigmasingin asosiy konseptlari quyidagilardir:

1. Ob'ekt (Object): Ma'lum bir vazifani bajarish uchun ma'lumotlarni va ular bilan amal qilish uchun funksiyalarni ichiga olgan yegane to'plam. Obyektning o'ziga xos holati (xususiyatlar) va xarakteristikasi (metodlar) mavjud bo'ladi.
2. Klass (Class): Ob'yektlarning asosiy tushunchasi yoki qismi. Klass, obyektlarning xususiyatlarini (ma'lumotlarni) va metodlarini (funksiyalarni) tashkil etadi. Ob'yektlar klassning bir misoldir.
3. Encapsulation (Jamlanish): Obyektning ma'lumotlarini va metodlarini bir jamaa ichida jamlash. Bunda obyektning ma'lumotlari faqat unga oid metodlar orqali kirish mumkin, shuning uchun ma'lumotlar xavfsizligi ta'minlanadi.
4. Inheritance (Miras oluvchi): Bir klassdan boshqa klasslarni yaratish uchun yana boshqa klassni asos qilib oluvchi konsept. Miras oluvchilik orqali bir klassning xususiyatlarini va metodlarini boshqa klasslarda ham qayta ishlash mumkin.
5. Polymorphism (Ko'plik qobiliyat): Bir obyektning bir qismi yoki funksiyasining bir nechta turdag'i ko'rinishlari bo'lishi. Bunda bir obyektning boshqa obyektlarga o'xshash usulda ishlashi va turli klasslarning metodlarining birlikda ishlashi mumkin.

OOP paradigmasi dasturlashni kengaytiradi va kodni muharrir va boshqalar uchun o'rganishni osonlashtiradi. U kodni yanada tozalaydi, qayta ishlashni osonlashtiradi va ularni bir-biridan ajralib turadigan, to'g'ridan-to'g'ri bog'langan obyektlar to'plami sifatida tavsiflaydi. Bu esa kodni qayta ishlashni osonlashtiradi va dasturchilarining yoki jamoalarining o'zaro ishlashini osonlashtiradi.

Savol-3:

SOLID nima?

SOLID - bu Dasturlashga asos solishtirilgan bir prinsip to'plami hisoblanadi. U 5 ta asosiy prinsipdan iborat bo'lib, qulaylik, toza kod, va kengaytirishga asoslangan dasturlash jarayonlari uchun bir ko'nikmaga aylanadi. Har bir harif SOLID prinsiplari birning o'zining belgilash prinsipi bilan ta'minlanadi.

1. Single Responsibility Principle (Yagona Mas'uliyat Prinsipi): Har bir klass yoki funksiya faqat bir yagona mas'uliyatni bajarishi kerak. Bu prinsip orqali klasslar va funksiyalar yagona vazifalarni bajarish uchun yaratiladi, bu esa kodning toza va boshqa modullar bilan mos kelishini ta'minlaydi.
2. Open-Closed Principle (Ochiq-Yopiq Prinsipi): Dasturchi kodni o'zgartirmasdan funksionallikni kengaytirishga imkon beradi. Klas va modullarni o'zgartirish yoki yangilash shu moduldan tashqari bo'lgan kodlariga ta'sir qilmaydi.
3. Liskov Substitution Principle (Liskov Almashtirish Prinsipi): Har qanday klassning obyektlari klasning to'liq funksionalini o'zgartirmasdan almashtirish imkonini beradi. Bu prinsip o'ziga xos interfeyslar va turli klasslarning bir birini almashtirishini ta'minlaydi.
4. Interface Segregation Principle (Interfeys Qisqartirish Prinsipi): Kliyent yalnız o'ziga kerak bo'lgan funksiyalarga ega bo'lishi kerak, boshqalariga ega bo'lishi shart emas. Bu prinsip orqali interfeyslar kliyentning faqat kerakli funksiyalarni taqdim etishiga yo'l qo'yadi, bu esa toza va modulli kod yozishni ta'minlaydi.
5. Dependency Inversion Principle (Tegishli Inversiya Prinsipi): Klasslar va modular qisqa modular yoki interfayslarga, ya'nii abstraksiyalarga bog'lanishlari kerak. Tegishli inversiya prinsipi orqali koddagi bog'lanishlar ortga yo'naltiriladi va bo'ysunish va qayta ishlashni osonlashtiradi.

SOLID prinsiplari kodning toza, to'g'ri va kengaytirishga tayanadigan qurilmalarni yaratishda qo'llaniladi. Ular dasturchilar uchun yaxshi tasniflanish, qayta ishlash, o'zgartirishlar va yangilashlarni osonlashtirishni ta'minlaydi. Bu prinsiplar yordamida kodning boshqa dasturchilar tomonidan tushunilishi va o'zgartirilishi ham osonlashadi.

Savol-5:

Microservice nima?

Microservice arxitekturasi, dasturlarni kichik, o'zaro bog'langan xizmatlar (microservices) to'plami sifatida tuzishga asoslangan dasturlash arxitekturasidir. Bu arxitekturada, dasturlar katta, monolitik kod o'rniغا, ko'p paydo bo'lgan xizmatlarga bo'linadi. Har bir xizmat o'zining o'zgartirishlarni o'zida saqlaydi va ularga o'ziga xos ish qilish va bog'lanish turlari mavjud bo'ladi.

Microservice arxitekturasi quyidagi asosiy xususiyatlarga ega:

1. Xizmatlar o'zaro o'zgartirishlarga tayanadi: Har bir xizmat o'zining o'zgartirishlarini o'zida saqlaydi va qo'shimcha xizmatlar bilan bog'lanish orqali o'zgarishlarni ulardan so'ray oladi. Bu, har bir xizmatning o'zining ozodlik va o'zgartirishlarga ishonchini beradi.
2. Kichik va mustaqil xizmatlar: Har bir xizmat biror muayyan vazifani bajaradi va keng qo'llanishda muvaffaqiyatli ishlashi uchun mustaqil o'zgartirishlarga ega bo'ladi. Bu, xizmatlarning qurulishi, testlanishi va boshqarilishi osonlik bilan amalga oshirilishi imkonini beradi.
3. Boshqaruv: Xizmatlar o'zining boshqaruv interfeyslariga ega bo'lishi mumkin. Boshqaruv interfeysi orqali xizmatlarni boshqarish, monitoring qilish, ma'lumotlarni saqlash va tahlil qilish imkoniyatlariga ega bo'lib, barcha xizmatlarga bir-biriga aloqada bo'lmaydi.
4. Scalability (o'sish): Xizmatlar o'zining ozodligi va mustaqil bo'lishi tufayli, har bir xizmatning o'sishini va ko'payishini osonlashtiradi. Ko'p xizmatlar qo'shilishi va xozirgi xizmatlarni ushlab turish jarayonlarida o'sish muhimdir.
5. Bog'lanishlar: Xizmatlar bog'lanish orqali birlashtiriladi. Ular RESTful API, HTTP, mesoservislarni, zeromq, AMQP kabi protokollarni qo'llab-quvvatlashlari mumkin. Xizmatlarning birlashtirilishi va aloqalar bilan ishslash, avtomatlashtirilgan va tashqi integratsiyalarni osonlashtiradi.

Microservice arxitekturasi yaratish, qo'llab-quvvatlash, boshqarish va o'zgartirishni osonlashtirishni ta'minlaydi. Uning afzalliklari dasturchilarga keng imkoniyatlar beradi va quyidagi muammolar bilan bog'liq bo'lgan kompleks dasturlash loyihalarini hallashga yordam beradi: ko'p qatlamlı dasturlash, modifikatsiya va o'zgartirishlarni boshqarish, ko'payib chiqish va iste'molchilar tomonidan qo'llanish ko'rsatkichlarini yaxshilash, va boshqalar.

Savol-6:

Docker nima?

Docker, ilovalarni avtomatlashtirilgan va portativ mahsulotlar sifatida ishlatalish uchun yaratilgan platformadir. Docker o'rtida qurilgan ilovalarni "konteynerlarga" (containers) o'rnatish, ishlatalish va boshqarishga imkon beradi. Konteynerlar, ilovaga to'g'ridan-to'g'ri o'rnatigan, isolatsiyalangan, xususiyatlar bilan tushunilgan muhitlar hisoblanadi.

Docker, ilovalarni alohida platformalar uchun ishlab chiqish, to'xtatish va o'chirishni osonlashtiradi. Har bir konteyner o'z o'zining fayllari, kutubxonalari va konfiguratsiyalarini o'zida saqlaydi, shuningdek o'zning barcha zarur modullarni o'z ichiga oladi. Konteynerlar o'zlarining hosil qilinish shartlari va tomonlari bilan yagona "konteynerizatsiya" o'qibatida yaratiladi.

Docker foydalanuvchilarga ilovalarini ishga tushirish va taqdim etishda katta imkoniyatlar beradi. Uning qulay interfeysi va boshqaruv panellari bor, shuningdek, ilovalar va xizmatlar bilan bog'liq muhim operatsiyalarni avtomatlashtirish, shalayni ta'minlash imkoniyatiga ega. Docker, ishga tushirilgan konteynerlarni ulkan miqdorda ishlab chiqish, ulardan foydalanish va ulardan o'zaro aloqani boshqarishni osonlashtiradi.

Docker, xususiyatlaridan biri, ilovalarni konteynerlarda qo'llab-quvvatlash orqali ilova ishga tushirish va qo'shishning osonligi va tezligini ta'minlaydi. Uning ishlatalishi juda keng o'rnatilgan va sohalarda xavfsizlik, so'nggi vaqt amalga oshirish, qo'llab-quvvatlash va ishlab chiqish jarayonlarini osonlashtiradi. Docker, dasturchilar va tizim boshqaruvchilari uchun yirik imkoniyatlarga ega bo'lgan bir virtualizatsiya platformasidir.

Savol-7: Kubernets nima?

Kubernetes (ya'ni "k8s") - bu yevropa standartida eng kuchli, yetarli boshqaruv vositasidir va shunchaki konteynerlashgan ilovalarni hamkorlash, boshqarish va ishlatishni osonlashtiruvchi bir dagi ilovalar tizimi. Kubernetes, ilovalarni yoritish, balanslashtirish, ishga tushirish va tarqatishni avtomatlashtirishga imkon beradi.

Kubernetes ma'nosini to'liq boshqaruv qiladigan bir jarayon platformasi sifatida o'rnatiladi. Uning asosiy maqsadi, ilovalarni tez va ishonchli tarzda qo'llab-quvvatlash va boshqarishda ko'p serverlar va konteynerlar bilan bog'liqliknii avtomatlashtirishdir. Kubernetes, ilovalarni yoritish uchun mavjud resurslarni (serverlar, xotiralar, tarmoqlar) samarali va moslashtiradi, konteynerlarni ko'paytirish, avtomatik tarqatish, ilovalarni qo'llab-quvvatlash va skaling qilishni osonlashtiradi.

Kubernetes kuchli boshqaruv mekanizmlariga ega bo'lgan, mazkur imkoniyatlarga erishish uchun quyidagi kabi xususiyatlarga ega:

1. Konteynerlarni boshqarish: Kubernetes, Docker, rkt, Podman va boshqa konteynerlar platformalarini qo'llaydi va bu yordamida ilovalarni boshqaradi.
2. Moslashtirish: Kubernetes, ilovalarni moslashtirish va qo'shimcha resurslar (xotiralar, tarmoqlar, qurilmalar)ni yoritishda yordam beradi.
3. Avtomatik tarqatish: Kubernetes, ilovalarni avtomatik tarqatish imkonini beradi, shuningdek, ilovani ko'paytirish va qisqartirish jarayonlarini boshqarishni osonlashtiradi.
4. Balanslashtirish: Kubernetes, trafikni ko'paytirish uchun ko'rsatkichlarni beradi va ilovalarni ko'paytirilgan holatda avtomatik ravishda belgilangan resurslarga bo'lgan trafikni tarqatishni ta'minlaydi.
5. Xavfsizlik: Kubernetes, ilovalarni xavfsizlik sohasida qo'llaniladigan yordamchi texnologiyalarni (kimyo, xavfsizlik xizmatlari) integratsiya qilishga imkon beradi.

Kubernetes, yirik ommaviy proyektlar va korporativ ishlab chiqarishlar uchun qat'iy tavsiya etiladi, chunki uni ishlatish, ilovalarni yaxshilash va har tomonlama boshqarish imkoniyatlarini osonlashtiradi. Bu orqali, Kubernetes iste'molchilariga ilovani yuqori darajada yuzaga keltirish, xavfsiz va qo'shimcha imkoniyatlarga ega bo'lgan konteynerlashgan ilovalarni boshqarishda yordam beradi.

Savol-8: Replica nima?

"Replica" so'zi IT sohasida bir necha kontekstda foydalaniladi. Shu tarzda, "replica" kontsepti Docker bilan bog'liq emas, lekin u Kubernetes va boshqa sohalarda ishlataladi.

Kubernetes yoki "K8s"ning asosiy konseptlaridan biri "Pod"dir. Pod, bir yoki bir nechta birlikda ishlaydigan konteynerlardan iborat bo'lib, ularga bir tizim resurslarini taqdim etadi. Kubernetes muhitida, "replica" esa bir Podning nusxasi yoki ko'paytmasi deb tavsiflanadi.

Replicalar, ilovalarning tashqi resurslarni qo'llab-quvvatlash va balanslashtirish uchun ishlataladi. Bir replica, bitta yoki bir nechta Podlarni yaratib, ularga bitta aloqador xizmat nomi beradi. Kubernetes, ilovani tarqatgan paytda, belgilangan replica sonini saqlayadi va Pods tuzishiga murojaat qiladi. Bu qilib, ilova so'rovlarni qabul qiladigan bir nechta konteynerlarni boshqarishga imkon beradi. Agar biron bir Podning xatoligi yuz berishi yoki band qolishi kerak bo'lса, Kubernetes avtomatik ravishda uni tiklash va qayta ishga tushirish imkoniyatiga ega bo'ladi.

Docker esa bir hostda yoki serverda faqat bitta konteyner ishga tushirish va boshqarish imkoniyatini beradi. Bunda konteynerlar bir-biridan o'ziga aloqador resurslarni taqsimlash, tarqatish yoki ilovani balanslashtirish imkoniyatiga ega emaslar. Docker, o'ziga xos konteyner yaratish va boshqarish uchun yaratilgan bir platforma sifatida ishlaydi. Kubernetes esa birga ilovalarni yoritish, balanslashtirish, skaling qilish va boshqarish uchun yaratilgan ilovalar tizimi sifatida xizmat qiladi va "replica" kontseptini o'z ichiga oladi.



Savol-9: Dekorator nima?

Dekorator Python tilida funksiyalarni yoki klasslarni o'zgartirishsiz yoki ularga qo'shimcha xususiyatlar qo'shish uchun ishlataladigan bir konstruktordir. Bu, Pythonning o'ziga xos vaqtinchalik imkoniyatlaridan biridir va funksiyalar va klasslar bilan ishlashda kuchli bo'lib chiqadi.

Dekoratorlar, funksiyalarni yoki klasslarni o'zgartirishsiz o'zgarishlarga qo'shimcha xususiyatlar qo'shishda keng qo'llaniladi. Bu, funksiyalar yoki klasslarning ish faoliyatini ko'rib chiqish, ularga yangi funktsiyalar qo'shish, boshqa funksiyalar bilan ishlashni boshqarish, xatolarini tekshirish va h.k. kabi qo'shimcha vazifalarni o'z ichiga oladi.

Dekoratorlar funksiyalar yoki klasslarni modifikatsiya qilishda juda qulay bo'ladi. Ular yaxshi ko'rgan misoldan foydalanib, shuningdek, kodni qisqa va qulay yozishga imkon beradi. Dekoratorlarning sintaksi, @ belgisi bilan boshlanadi va funksiya nomi yoki klass nomi bilan birga ishlataladi.

Misol uchun, quyidagi misoldek dekorator funksiyani yaratadi:

```
python Copy code
def log_decorator(func):
    def wrapper():
        print("Funksiya boshlanishi")
        func()
        print("Funksiya tugadi")
    return wrapper

@log_decorator
def hello():
    print("Salom, Dunyo!")

hello()
```

Bu misolda `log_decorator` dekoratori funksiya `hello` ni chaqirishdan oldin va keyin qo'shimcha ishlar bilan o'zgartiradi. Natija sifatida funksiya boshlanishi va tugashi orqali "Salom, Dunyo!" matni chiqadi.

Savol-10:

Pythonda Dataclass nima?

Python tilida "dataclass" deb nomlanadigan modul, Python standart kutubxonasi ichida joylashgan bir funksiyadir. "dataclass" funksiyasi obyektlarni ma'lumotlarni saqlash va ishlash uchun ma'lumotklasslariga aylantirish uchun ishlatiladi.

"dataclass" funksiyasi, obyektlarni boshqarish uchun kerakli funktsiyalarni va xususiyatlarini avtomatik ravishda yaratadi. Bu, obyektlarning qiymatlari orqali avtomatik ravishda `__init__`, `__repr__`, `__eq__`, `__hash__` va boshqa metodlarni yaratadi.

Quyidagi misol kod orqali "dataclass" funksiyasini ko'rib chiqamiz:

python

 Copy code

```
from dataclasses import dataclass

@dataclass
class Person:
    name: str
    age: int

person = Person("John", 25)
print(person) # Natija: Person(name='John', age=25)
```

Ushbu misolda "Person" deb nomlangan "dataclass" yaratiladi va "name" va "age" deb nomlangan ikkita xususiyati mavjud. "dataclass" funksiyasi avtomatik ravishda `__repr__` metodini yaratadi va bu metod yordamida obyektlar matn ko'rinishida ifodalanganadi. Misoldagi `print(person)` ifodasi "Person(name='John', age=25)" chiqishini beradi.

"dataclass" funksiyasi, ma'lumotlarni saqlash va ularga qo'shimcha funktsiyalar qo'shishni osonlashtiradi va Python 3.7 versiyasidan boshlab standart kutubxonada mavjud bo'lib, kodni qisqa va qulayroq yozishga imkon beradi.

Savol-11:

OOPning 4ta ustuni

OOP (Object-Oriented Programming) – bu dasturlash paradigmasi, dasturlarni obyektlar va ularning o'zaro munosabatlari asosida yozishga asoslanadi. OOP, kodni organizatsiyalash, qayta ishlanishi, o'zgartirilishi va boshqarilishi uchun qulayliklar ta'minlaydi. Ustunliklari quyidagicha:

1. Encapsulation (Qayta ishlanishi): Obyektlar ma'lumotlarini va funksiyalarini yig'ib o'z ichida saqlayadi. Bu, ma'lumotlarni boshqa obyektlardan yashirish va ularga faqat talab qilinadigan interfeys orqali murojaat qilish imkonini beradi.
2. Inheritance (Merosi): Bir obyektning boshqa bir obyekttan xususiyatlarini va funksiyalarini olmashtirish imkonini beradi. Bu yordamida kodni qayta ishlash va qayta ishlash talablari to'g'risida qulaylik yaratiladi.
3. Polymorphism (Ko'plik): Bir obyektning bir amalni turli shakllarda bajarishi imkonini beradi. Bu, boshqa obyekt turlari bilan mos kelishuvchi metodlar yaratish va ularni bajarish imkonini beradi.
4. Abstraction (Ajralish): Ma'lumotlarni va funksiyalarini o'z ichiga olgan obyektlar yordamida asosiy ma'lumotlarni ifodalash imkonini beradi. Bu, kompleks dasturlarni oson va yordamchi qilishga yordam beradi.

OOP, kodni modular va yengil qiladi, qayta ishlashni osonlashtiradi, kod qayta ishlash va ishlab chiqishga to'g'ri kelishuvlar qilishni osonlashtiradi. Bu esa dasturchilar uchun kodni boshqarish va kengaytirishni osonlashtiradi.

Savol-12:

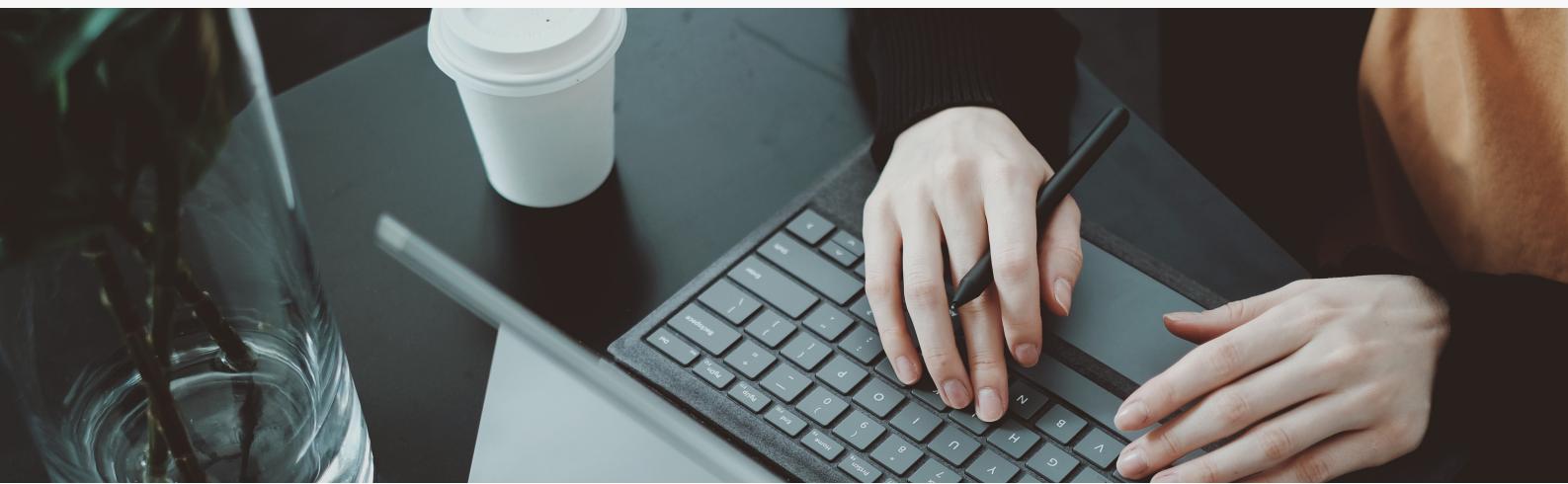
Websocket nima?

WebSoket, istemciler va serverlar orasida ikki tomonloma kommunikatsiya uchun mo'ljallangan bir kommunikatsiya protokolining nomidir. Bu protokol orqali istemciler va serverlar o'rtasida haqiqiy vaqtli ma'lumotlar bir-biriga yuboriladi. WebSoket, ma'lumotlar almashishini amalga oshirish uchun o'zgaruvchan vaqtli davrlar bilan bog'liq aloqani tashkil qiladi.

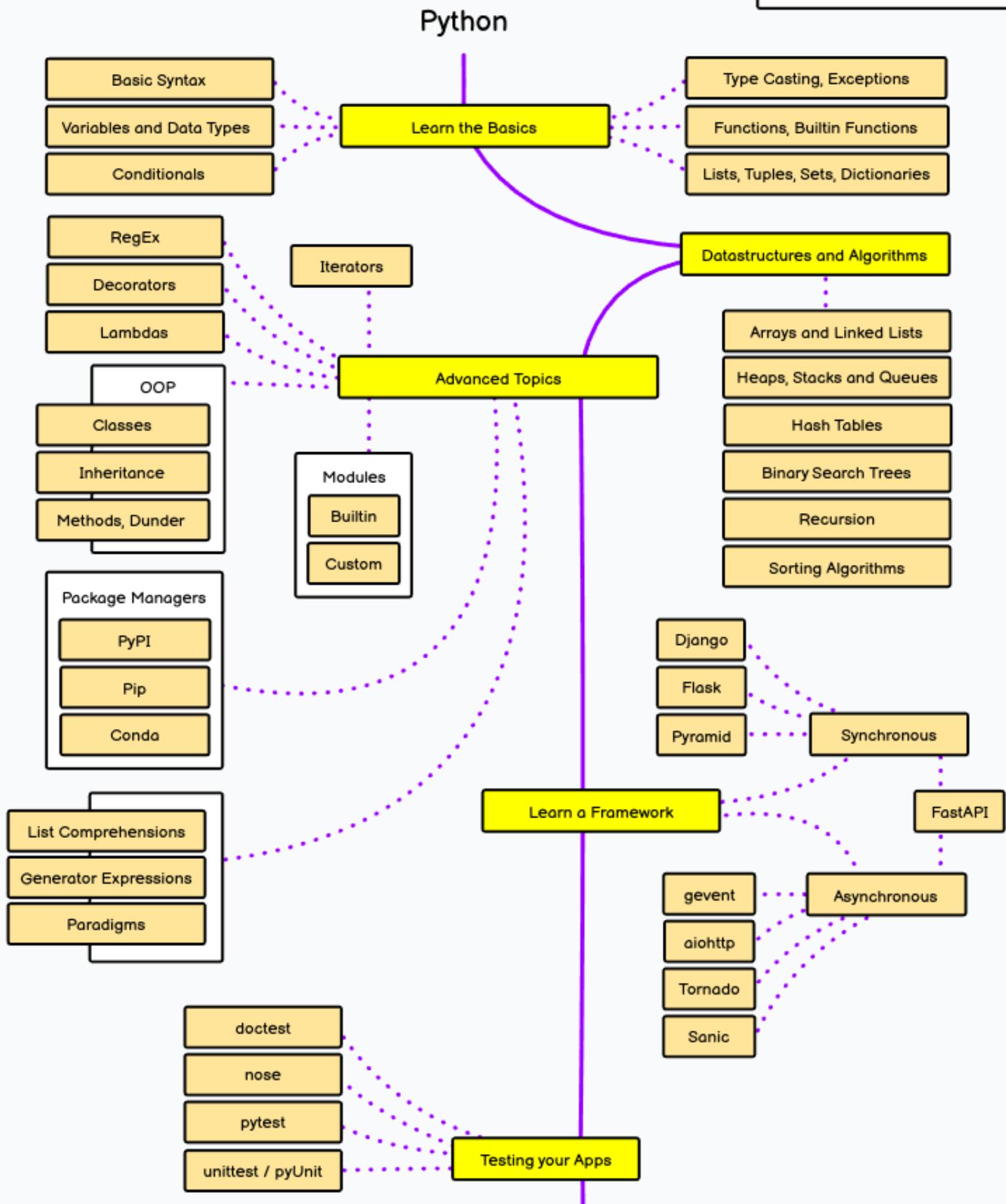
WebSoketning asosiy xususiyatlari quyidagicha:

1. Real-time Communication (haqiqiy vaqtli kommunikatsiya): WebSoket protokoli yordamida ma'lumotlar haqiqiy vaqtida istemcilar va serverlar orasida almashishadi. Bu, yagona so'rov/javobga asoslangan klassik HTTP protokolidan farq qiladi.
2. Tovushli (Full-duplex) aloqa: WebSoket protokoli orqali istemci va serverlar bir vaqtida ikki yo'nalishda ma'lumotlar almashishi mumkin. Bu, istemci va serverlar orasidagi uzatilmasiz kommunikatsiyani ta'minlaydi.
3. Protokol: WebSoket, HTTP protokoli asosida ishlaydi, lekin o'zining maxsus talablariga ega bo'lgan aloqa davrini o'z ichiga oladi. Uchlash davri davomida HTTP protokoliga o'girib, keyin esa aloqa WebSoket protokoliga o'tadi.

WebSoket, dinamik veb ilovalar, chat dasturlari, o'yinlar, reallik ma'lumotlar, hisobotlar va boshqa sohalarda haqiqiy vaqtli aloqalar yaratish uchun keng qo'llaniladi.



Python yo'l xaritasi





Alimardon Boqijonov-Python
Dasturchi, Mentor, INFIN1TY.UZ
asoschisi va yaxshi do'st.

Kuch va Qudrat Allohdandir.
Maqsad qo'yib harakat qilsang
Albatta unga erishasan.