**№1-LABORATORIYA ISHI**

**1.1. Dasturlash tillarida ma’lumotlarning standart turlari.**

**1.2. Funksiyalarni amalga oshirish.**

**Ishdan maqsad:** Ma’lumotlarning oddiy sozlangan va nostandart toifalarini o‘rganish va ularni tadqiq qilish, hamda matematik misollar uchun funksiyalar yozishni o‘rganish.

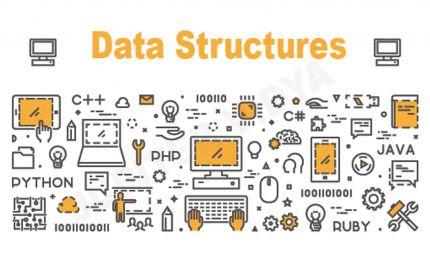
**Qo‘yilgan masala:** Haqiqiy, belgili, mantiqiy toifadagi ma’umotlarni e’lon qilish, nostandart toifalarni yaratish va ularga doir misollarning dasturini ishlab chiqish. Funksiyalar yaratish.

**Ish bajarish tartibi:**

* Tajriba ishi nazariy ma’lumotlarini o‘rganish;
* Berilgan topshiriqning algoritmini ishlab chiqish;
* Python dasturlash muhitida dasturni yaratish;
* Natijalarni tekshirish;
* Hisobotni tayyorlash va topshirish.

**NAZARIY QISM**

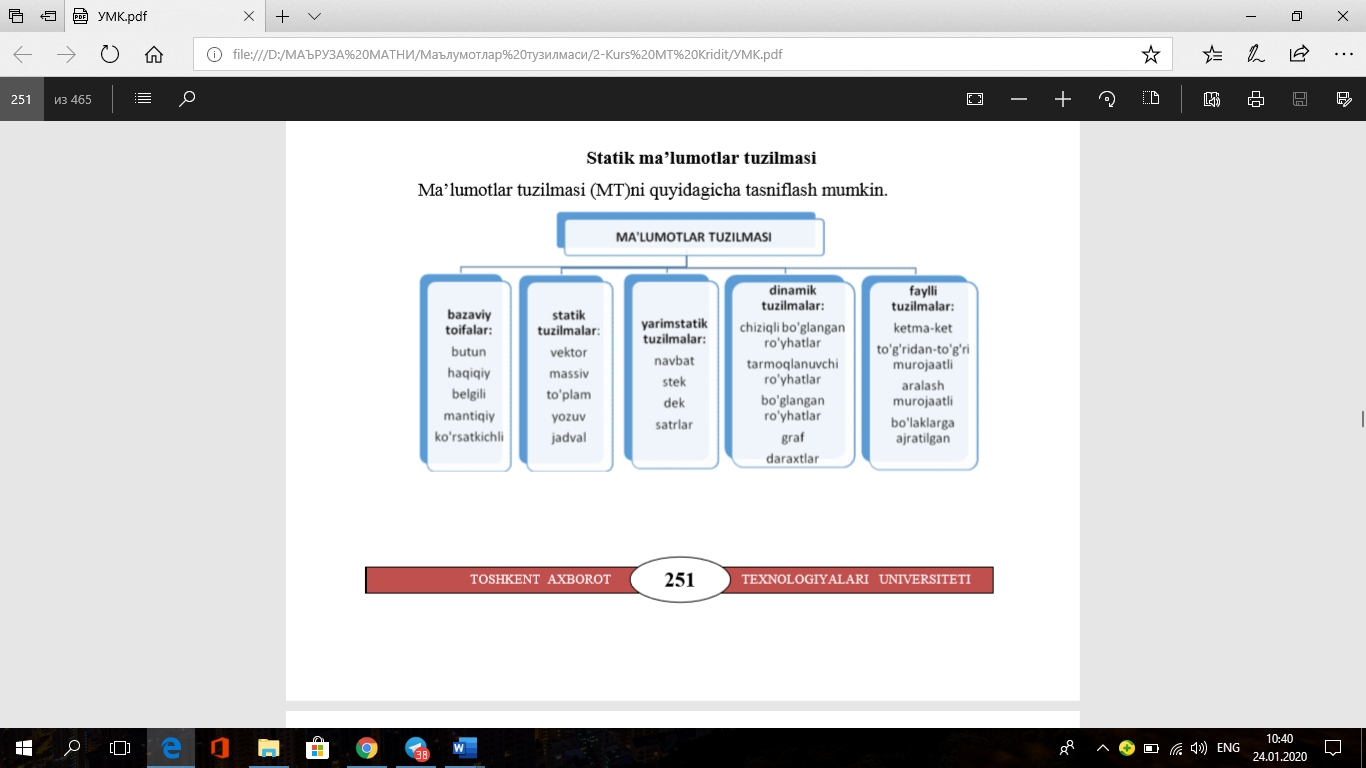
Ma’lumotlar tuzilmalari o‘zi nima?



Barcha dastur yoki dasturiy mahsulotning asosida ikkita birlik yotadi: ma’lumotlar va ular ustida qandaydir amallar bajaradigan algoritmlar. Algoritmlar ma’lumotlarni biz yoki dastur uchun foydali bo‘lgan axborot ko‘rinishiga keltirib beradi. Algoritmlar shu ma’lumotlar ustida amallarni (o‘qish, yozish, yangilash, o‘chirish) samarali va tez bajara olishi uchun biz shu ma’lumotlarni ma’lum bir strukturaga solgan holda saqlashimiz kerak bo‘ladi.

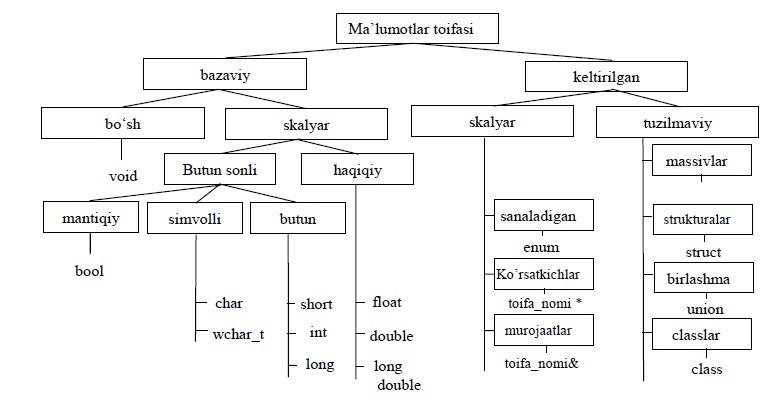
*Ma’lumotlar tuzilmasi* — bu ma’lumotlarni samarali o‘qish va o‘zgartirish imkonini beruvchi, ma’lumotlarni saqlash va boshqarishning bir formatga solingan shaklidir.

Soddaroq qilib aytganda, *ma’lumotlar tuzilmasi* — bu ma’lumotlarning ma’lum bir strukturaga solingan, ular o‘rtasida ma’lum bir bog‘lanishlar yaratilgan va ular ustida ma’lum amallar bajaruvchi funksiyalardan tashkil topgan guruh. Eng sodda ma’lumotlar tuzilmasiga misol qilib massiv (array)ni ko‘rsatishimiz mumkin.



*1-rasm. Ma’lumotlar tuzilmasining tasniflanishi*

***Ma’lumotlar toifalari.*** Ko‘plab dasturlash tillarida ma’lumotlar bazaviy va keltirilgan toifalarga ajratiladi. Ma’lumotlarning toifalarini 1.1-rasmdagidek klassifikatsiyalash mumkin.



*2-rasm. Toifalar klassifikatsiyasi*

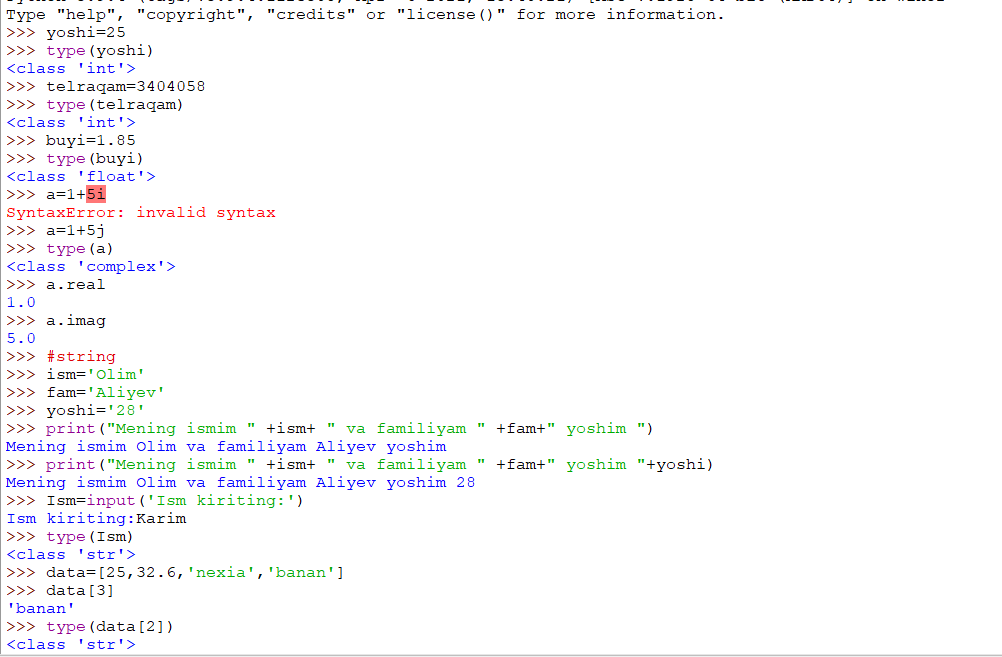
Ma’lumotlarning ixtiyoriy toifasi qiymatlar sohasi va ular ustida bajarilishi mumkin bo‘lgan amallar orqali tavsiflanadi. *void*kalit so‘zi hech qanday toifaga ega emaslikni anglatadi. Bunday toifadagi funksiyalar hech qanday qiymatni qaytarmaydi. Lekin asosiy dastur tanasi, ya’ni *main()*funksiyasi *void*toifasiga ega bo‘lolmaydi, u *int*toifasida bo‘lishi kerak.

***Standart ma'lumotlar turlari.*** Dasturlashda ma'lumotlar turi muhim tushunchadir.

O‘zgaruvchilar har xil turdagi ma'lumotlarni saqlashlari mumkin, va ular bilan turli xil amallarni qilish mumkin.

Python standart o‘rnatilgan quyidagi ma'lumotlar turlariga ega:

1. Matnli ma'lumot turi: *str*
2. Sonli ma'lumot turlari: *int, float, complex*
3. Tartibli ma'lumot turlari: *list, tuple, range*
4. Lug‘atli ma'lumot turi: *dict*
5. To‘plamli ma'lumot tulari: *set, frozenset*
6. Boolean (True/false) ma'lumot turi: *bool*
7. Binar(ikkilik) ma'lumot turlari: *byte, bytearray, memoryview*



Hozircha yuqoridagilar haqida qayg‘urmasangiz ham bo‘ladi, bu shunchaki qanday ma'lumotlar turlari borligi haqida tanishib chiqishingiz uchun. Ma'lumot turlari haqida keyinchalik alohida batafsil to‘xtalib o‘tamiz.

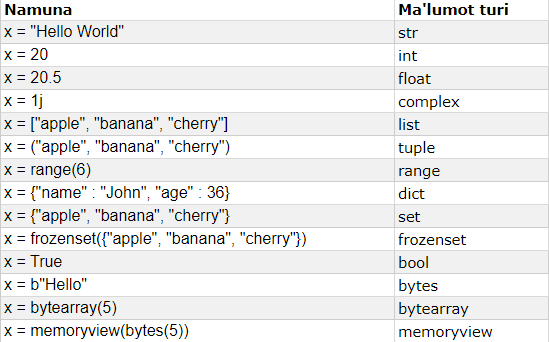
***Ma'lumotlar turini aniqlash.*** Ma'lumot turini bilib olish uchun type() funksiyasidan foydalanishingiz mumkin. Istalgan ma'lumotlarni turini bilib olishingiz mumkin:

*x = 10*

*print(type(x))*

Yuqoridagi misolda type(x) ifodasi x ning turini matn ko‘rinishida qaytaradi va uni konsolda yozib chiqarish uchun doimgidek print() funksiyasidan foydalanamiz.

***Ma'lumotlar turini belgilash.*** Pythonda, o‘zgaruvchiga qiymat tayinlaganingizda ma'lumotlar turi o‘rnatiladi:



***Maxsus ma'lumot turini o‘rnatish.*** Agar ma'lumotlar turini ko‘rsatmoqchi bo‘lsangiz, quyidagi konstruktor funksiyalaridan foydalanishingiz mumkin:



**1.2. Funksiyalarni amalga oshirish**

Funksiya – bu ko‘p marta ishlatiladigan dastur bo‘lagi. Funksiyalar ma'lum buyruqlar blokini ko‘rsatilgan nom bilan saqlash va shu blokni dasturning istalgan joyida, istalgan miqdorda bajarish imkonini beradi.

Funksiyalar ***def***zahira so‘zi orqali aniqlanadi. Bu so‘zdan so‘ng funksiya ***nomi***, undan so‘ng qavs va shu qavs ichida bir necha o‘zgaruvchilarni ko‘rsatish mumkin bo‘ladi va oxirida ikki nuqta (:) yoziladi. Shulardan so‘ng funksiyani tashkil qiluvchi buyruqlar bloki yoziladi. Quyidagi misolda buning oson ekanligini ko‘rish mumkin.

***Misol***:

Ikki sonnning yig‘indisini topuvchi funksiya yaratish.

*a,b=eval(input('ikkita sonni kiriting‘)) #a va b sonlarni kiritish*

*def funk(a,b): # funksiya e’lon qilish*

*return a+b #qiymat qaytarish*

*print(funk(a,b)) #funksiyaga murajaat qilish*

NATIJA:



***Misol*:**(function1.py nomi bilan saqlang)

*def Salom():*

*print('Salom, Dunyo!') # funksiyaga tegishli blok*

*# Funksiya oxiri*

*Salom() # funksiyani chaqirish*

*Salom() # ya'na bir marta funksiyani chaqirish*

***Natija:***

$ python function1.py

*Salom, Dunyo!*

*Salom, Dunyo!*

***Bu qanday ishlaydi:***

Biz *Salom*funksiyasini yuqorida aytib o‘tilgan qoida bo‘yicha aniqladik. Bu funksiya hech qanday paramert qabul qilmaydi shuning uchun qavs ichida xech qanday parametr yozilmadi. Funksiya parametri – bu qandaydir kiruvchi qiymatlar bo‘lib, tegishli natija olish uchun biz uni funksiyaga berishimiz mumkin.

E'tibor bering, bitta funksiyani ko‘p marta chaqirishimiz mumkin, demak, aynan bir xil dastur kodini qayta-qayta yozishga hojat yo‘q.

***Pythondagi funksiya*** - bu argumentlarni qabul qiladigan va qiymatni qaytaradigan obyekt. Odatda funksiya **def** ifodasi yordamida aniqlanadi.

Keling, eng oddiy funksiyani aniqlaylik:

***def*** *add(x, y):*

***return*** *x + y*

Return bir qiymat qaytaradi. Bizning holatda, funksiya x va y yig‘indisini qaytaradi.

Endi biz uni chaqirishimiz mumkin:

***>>>*** *add(1, 10)*

*11*

***>>>*** *add('abc', 'def')*

*'abcdef'*

Funksiya har qanday murakkablikda bo‘lishi mumkin va har qanday obyektlarni qaytarishi mumkin (ro‘yxatlar, tuplar va hatto funksiyalar!):

***>>> def*** *newfunc(n):*

***...*** ***def*** *myfunc(x):*

***...*** ***return*** *x + n*

***...*** ***return*** *myfunc*

***...***

***>>>*** *new = newfunc(100) # new - это функция*

***>>>*** *new(200)*

*300*

Funksiya qaytarish iborasi bilan tugamasligi mumkin va funksiya [None](https://pythonworld.ru/tipy-dannyx-v-python/none.html) qaytarmaydi :

***>>> def*** *func():*

***...*** ***pass***

***...***

***>>>*** *print(func())*

*None*

**Funksiya argumentlari**

Funksiya ixtiyoriy miqdordagi argumentlarni qabul qilishi yoki umuman qabul qilmasligi mumkin. O‘zboshimchalik bilan ko‘p argumentlarga ega bo‘lgan funksiyalar, pozitsion va nomlangan argumentlarga ega bo‘lgan funksiyalar majburiy va ixtiyoriydir.

***>>> def*** *func(a, b, c=2): # c - необязательный аргумент*

***...*** ***return*** *a + b + c*

***...***

***>>>*** *func(1, 2) # a = 1, b = 2, c = 2 (по умолчанию)*

*5*

***>>>*** *func(1, 2, 3) # a = 1, b = 2, c = 3*

*6*

***>>>*** *func(a=1, b=3) # a = 1, b = 3, c = 2*

*6*

***>>>*** *func(a=3, c=6) # a = 3, c = 6, b не определен*

*Traceback (most recent call last):*

*File "", line 1, in*

*func(a=3, c=6)*

*TypeError: func() takes at least 2 arguments (2 given)*

Funksiya, shuningdek, pozitsion argumentlarning o‘zgaruvchan sonini olishi mumkin, keyin \* ism oldiga qo‘yiladi:

***>>> def*** *func(\*args):*

***...*** ***return*** *args*

***...***

***>>>*** *func(1, 2, 3, 'abc')*

*(1, 2, 3, 'abc')*

***>>>*** *func()*

*()*

***>>>*** *func(1)*

*(1,)*

Misoldan ko‘rinib turibdiki, args - bu funksiyaga berilgan barcha argumentlarning [yig‘indisi](https://pythonworld.ru/tipy-dannyx-v-python/kortezhi-tuple.html) va siz o‘zgarmaydigan bilan xuddi tuple kabi ishlashingiz mumkin.

Funksiya ixtiyoriy ravishda nomlangan argumentlarni qabul qilishi mumkin, keyin \*\* ism oldiga qo‘yiladi:

**>>> def** func(\*\*kwargs):

**...**  **return** kwargs

**...**

**>>>** func(a=1, b=2, c=3)

{'a': 1, 'c': 3, 'b': 2}

**>>>** func()

{}

**>>>** func(a='python')

{'a': 'python'}

Kvarg o‘zgaruvchisida biz [lug‘atni](https://pythonworld.ru/tipy-dannyx-v-python/slovari-dict-funkcii-i-metody-slovarej.html) saqlaymiz, u bilan biz xohlagan narsani qila olamiz.

***Funksiya parametrlari.*** Funksiyalar parametrlar, ya'ni funksiyaga berilishi mumkin bo‘lgan qiymatlar qabul qila oladi va ular ustuda biror amal bajarishi mumkin. Bu parametrlar o‘zgaruvchilarga o‘xshaydi. Faqat ulardan farqi bu o‘zgaruvchilarning qiymati funksiyani chaqirish vaqtida o‘rnatiladi. Funksiya ish boshlagan vaqtda bularga qiymat biriktirilgan bo‘ladi.

Parametrlar funksiya aniqlanayotgan vaqtda qavs ishida vergul bilan ajratilgasn holda ko‘rsatiladi. Ularga qiymatni funksiyani chaqirganimizda biriktiramiz. Ushbu atamalarga e'tibor bering: funksiya e'lon qilinayotgan vaqtda ko‘rsatilgan nomlar *parametrlar*, funksiyani chaqirayotganimizda unga berilgan qiymatlar esa *argumentlar*deyiladi.

***Misol*:**( func\_param.py nomi bilan saqlang)

*def printMax(a, b):*

*if a &gt; b:*

*print(a, 'katta')*

*elif a == b:*

*print(a, 'teng‘, b)*

*else:*

*print(b, 'katta')*

*printMax(3, 4) # qiymatlarni to‘g‘ridan-to‘g‘ri berish*

*x = 5*

*y = 7*

*printMax(x, y) # o‘zgaruvchilarni argument sifatida uzatish.*

***Natija*:**

$ python func\_param.py

*4 katta*

*7 katta*

***Mahalliy (локальные) parametrlar.*** Funksiyaning ichida e'lon qilingan o‘zgaruvchilar huddi shu nomdagi funksiya tashqarisida e'lon qilingan o‘zgaruvchilar bilan xech qanday bog‘liklikka ega emas, ya'ni bu o‘zgaruvchilar mahalliy o‘zgaruvchilar hisoblanadi. Bu o‘zgaruvchining *ko‘rinish maydoni(область видимости)*deyiladi. Har bir o‘zgaruvchining ko‘rinish maydoni o‘zgaruvchi aniqlangan amallar bloki va shu o‘zgaruvchi e'lon qilingan nuqta bilan chegaralangan.

***Misol*:**(func\_local.py nomi bilan saqlang)

*x = 50*

*def func(x):*

*print("x teng", x)*

*x = 2*

*print("Mahalliy x qiymatini", x, "ga o‘zgartiramiz")*

*func(x)*

*print("x qiymati qanday bo‘lsa, shunday turibdi", x)*

***Natija*:**

$ python func\_local.py

*x teng 50*

*Mahalliy x qiymatini 2 ga o‘zgartiramiz*

*x qiymati qanday bo‘lsa, shunday turibdi 50*

Python dasturlash tilida funksiyani def kalit so‘zi bilan e’lon qilinadi.

Biror ishni qayta qayta hisoblashning oldini oladi va yaratgan funksiyaga murojaat qilish orqali bajariladi.

Misol:

Ikki sonnning yig‘indisini topuvchi funksiya yaratish.

*a,b=eval(input('ikkita sonni kiriting‘)) #a va b sonlarni kiritish*

*def funk(a,b): # funksiya e’lon qilish*

*return a+b #qiymat qaytarish*

*print(funk(a,b)) #funksiyaga murajaat qilish*

NATIJA:



Funksiyani amalga oshirishda quyidagicha bajarish ham mumkin:

*a,b=eval(input('ikkita sonni kiriting‘)) #a va b sonlarni kiritish*

*def funk(a,b): # funksiya e’lon qilish*

*print(a+b) #qiymat chop etish*

*funk(a,b) #funksiyaga murajaat qilish*

**Topshiriqlar**

1. Uch xonali son berilgan. Berilgan sonning raqamlari yig‘indisini topuvchi dastur tuzing.

| **Input:** | **Output:** |
| --- | --- |
| N kiriting: 867 | 21 |

1. To‘rt xonali son berilgan. Berilgan sonning raqamlari ko‘paytmasini topuvchi dastur tuzing.

| **Input:** | **Output:** |
| --- | --- |
| N kiriting: 4368 | 576 |

1. To‘g‘ri burchakli uchburchakning katetlari berilgan. Berilgan uchburchakning medianasini topuvchi dastur tuzing *(Yaxlitlashda format() dan foydalaning)*.

| **Input:** | **Output:** |
| --- | --- |
| b,c ni kiritng: 5,6 | 15.25 |

1. Uchburchakning tomonlari berilgan. Uning a tomoniga tushuvchi medianasini topuvchi dastur tuzing *(Yaxlitlashda format() dan foydalaning)*.

| **Input:** | **Output:** |
| --- | --- |
| b,c ni kiritng: 7,5,9 | 6.38 |

1. Shar radiusi berilgan. Uning hajmini topuvchi dastur tuzing *(Yaxlitlashda format() dan foydalaning)*.

| **Input:** | **Output:** |
| --- | --- |
| r ni kiriting: 7.5 | 31.42 |