Python

1. **Proqramlaşdırma dili.**

Proqramlaşdırma dili nədir?

Bildiyimiz kimi, bir insanla ünsiyyət qurmaq üçün bizə müəyyən bir dil lazımdır, kompüterlərlə ünsiyyət qurmaq üçün proqramçılara da lazım olan dilə **proqramlaşdırma dili** deyilir.

Proqramlaşdırma dili proqramçıların (developerlərin) kompüterlərlə əlaqə saxlamaq üçün istifadə etdiyi kompüter dilidir. Bu, müəyyən tapşırığı yerinə yetirmək üçün hər hansı bir xüsusi dildə (C, C++, Java, Python) yazılmış təlimatlar toplusudur.

Proqramlaşdırma dilinin növləri.

1. **Low-level programming language**

**Low-level proqramlaşdırma dili** maşından asılı olan (0s və 1s) proqramlaşdırma dilidir. Prosessor kompilyator və ya tərcüməçiyə ehtiyac duymadan aşağı səviyyəli proqramları birbaşa işlədir, beləliklə, aşağı səviyyəli dildə yazılmış proqramlar çox sürətli işlədilə bilər. Lakin bu dildə ola proqramlardakı səhvləri tapmaq çətin olur. Low-level programming language daha iki hissəyə bölünür -

1. **Machine language**

**Machine language** low-level proqramlaşdırma dilinin növüdür. Buna maşın kodu və ya obyekt kodu da deyilir. Maşın dilini oxumaq kompüter üçün daha asandır, çünki o, adətən ikili və ya onaltılıq formada (əsas 16) göstərilir. Proqramları çevirmək üçün tərcüməçi tələb olunmur, çünki kompüterlər maşın dili proqramlarını birbaşa başa düşür.

Maşın dilinin üstünlüyü ondan ibarətdir ki, o, proqramçıya proqramları yüksək səviyyəli proqramlaşdırma dilindən daha sürətli icra etməyə kömək edir.

2. **Assembly Language**

**Assembly Language (ASM)** həm də xüsusi prosessorlar üçün nəzərdə tutulmuş aşağı səviyyəli proqramlaşdırma dilinin bir növüdür. O, simvolik və insan üçün başa düşülən formada təlimatlar toplusunu təmsil edir. Assembly Language-i maşın dilinə çevirmək üçün assemblerdən istifadə edir.

Assembly dilinin üstünlüyü proqramın icrası üçün daha az yaddaş və daha az icra müddəti tələb etməsidir.

1. **High-level programming language**

**High-level** **proqramlaşdırma dili** **(HLL)** istifadəçi üçün istifadəçi yönümlü proqramları və veb-saytlar hazırlamaq üçün nəzərdə tutulmuşdur. Bu proqramlaşdırma dili proqramı maşın dilinə çevirmək (proqramı icra etmək) üçün kompilyator və ya tərcüməçi tələb edir.

High-level proqramlaşdırma dilinin əsas üstünlüyü ondan ibarətdir ki, oxumaq, yazmaq və saxlamaq asandır.

High-level proqramlaşdırma dilinə Python, Java, JavaScript, PHP, C#, C++, Objective C, Cobol, Perl, Pascal, LISP, FORTRAN və Swift proqramlaşdırma dilləri daxildir.

Yüksək səviyyəli dil daha da üç hissəyə bölünür -

**1. Procedural Oriented programming language**

(Prosedur yönümlü (prosedural) proqramlaşdırma dili)

**Prosedur yönümlü proqramlaşdırma (POP) dili** strukturlaşdırılmış proqramlaşdırmadan irəli gəlir və prosedur çağırışı konsepsiyasına əsaslanır. Proqramı rutinlər və ya funksiyalar adlanan kiçik proseduraları bölür.

**Prosedur yönümlü proqramlaşdırma dili** proqramçı tərəfindən IDE, Adobe Dreamweaver və ya Microsoft Visual Studio kimi proqramlaşdırma redaktorundan istifadə etməklə həyata keçirilə bilən proqram yaratmaq üçün istifadə olunur.

POP dilinin üstünlüyü ondan ibarətdir ki, o, proqramçılara proqram axınını asanlıqla izləməyə kömək edir və kod proqramın müxtəlif hissələrində təkrar istifadə oluna bilər.

**2. Object-Oriented Programming (OOP) language**

(Obyekt yönümlü proqramlaşdırma dili)

Obyekt yönümlü proqramlaşdırma (OOP) dili obyektlərə əsaslanır. Bu proqramlaşdırma dilində proqramlar obyekt adlanan kiçik hissələrə bölünür. Proqramı təkrar istifadə edilə bilən, səmərəli və istifadəsi asan etmək üçün proqramda encapsulation, inheritance, polymorphism, abstraction kimi tikinti blokları (building blocks) vardır.

Obyekt yönümlü proqramlaşdırmanın əsas üstünlüyü ondan ibarətdir ki, OOP daha sürətli və daha asan yerinə yetirilir, saxlanılır, dəyişdirilir. Burada elementlər bir-biri ilə daha sıx əlaqəli olur.

Qeyd: Obyekt yönümlü proqramlaşdırma dili aşağıdan-yuxarıya yanaşmanı izləyir.

**3. Middle-level programming language**

Middle-level proqramlaşdırma dili low-level proqramlaşdırma dili ilə high-level proqramlaşdırma dili arasında yerləşir. O, həmçinin intermediate proqramlaşdırma dili və psevdodil kimi də tanınır.

Orta səviyyəli proqramlaşdırma dilinin üstünlükləri yüksək səviyyəli proqramlaşdırmanın xüsusiyyətlərini dəstəkləməsi, istifadəçi dostu bir dil olması və maşın dili və insan dili ilə sıx əlaqəli olmasıdır.

İmperative və declarative programming.

**İmperativ proqramlaşdırma** nədir? Bu suala cavab vermənin ən qısa yolu, imperativ proqramlaşdırma dilindən istifadə edərək bir şeyi NECƏ etmək istədiyimizi söyləməyə çalışdığımızı söyləməkdir. Bu məqsədlə biz tətbiqin cari vəziyyətini dəyişə bilən ifadələrdən istifadə edirik. İmperativ proqramlaşdırma dilində hərəkəti detalına qədər bilməliyik.

**Deklarativ proqramlaşdırma** nədir? Deklarativ proqramlaşdırma termini çox vaxt İmperativ proqramlaşdırmanın əksi kimi istifadə olunur. Qısaca desək, bu paradiqma sizə NƏ etmək istədiyinizi bəyan etməyə imkan verir. Deklarativ proqramlaşdırma dilində bu dil hər şeyi özü edir və yazmaq daha asandır.

Kompilyasiya və interpretasiya.

(İnterpreter və compiler (Computer Program Executer))

Translyasiyanın iki qaydası var: interpretasiya və kompilyasiya. İnterpretasiya – şifahi tərcüməyə oxşayır. Giriş proqramının hər bir təlimatı tərcümə olununur və yerinə yetirilir. Bu qaydada təkrar təlimatlar hər dəfə kodlaşdırılır. Kompilyasiya isə yazılı tərcüməyə bənzəyir. Proqram yerinə yetirilməzdən qabaq proqramın bütün tərcüməsi yığılır.

**İnterpreter** proqramı oxumaq və icra etmək qabiliyyətinə malik sistem proqramı kimi adlandırıla bilər, daha çox proqramları icra edə bilərsiniz. Bu icraya yüksək səviyyəli dilin mənbə kodu, əvvəlcədən tərtib edilmiş proqramlar və skriptlər daxildir. Qeyd etmək lazımdır ki, interpreter proqramı sətir-sətir icra edir, yəni bir ifadəni bir anda tərcümə edir. Bu xüsusiyyət proqramçılara debug zamanı hər hansı xüsusi xətti yoxlamağı asanlaşdırır, lakin bütün proqramın ümumi icra müddətini ləngidir.

**Kompilyator** proqramlaşdırma dilində (mənbə dili, adətən HLL) yazılmış mənbə kodunu başqa kompüter dilinə (hədəf dil) çevirən sistem proqramıdır (kompüter proqramı toplusu). O, müəyyən proqramlaşdırma dilində yazılmış hər bir ifadəni emal edir və onları maşın dilinə çevirir. Proqramı yazdıqdan sonra proqramçı kodu mənbə ifadələrini ehtiva edən faylın adını göstərən müvafiq dil tərtibçisində işlədir.

1. **Alqoritm.**

**Alqoritm** — verilmiş məsələni həll etmək üçün ilkin verilənlərlə icra olunan hesabi və məntiqi əməliyyatların sonlu sayda ardıcıllığıdır.

Proqramçı isə proqramlaşdırma dillərini bilməklə, qarşıya qoyulan məsələnin kompüterdə həllini həyata keçirmək üçün proqram yazır və onu kompüterdə yerinə yetirir.

Kompyuterin alqoritmi başa düşməsi üçün proqramlaşdırma dillərindən istifadə edilir. Məsələ həll edərkən əvvəlcə yerinə yetiriləcək əməliyyatların alqoritmi tərtib edilir, daha sonra bu əməliyyatlar hər-hansı alqoritm (proqramlaşdırma) dilində əmrlər şəklində yazılır. Tərtib olunmuş proqram xüsusi əlavələr (translyator proqramlar) vasitəsilə yerinə yetirilir və ya maşın koduna çevrilir.

Alqoritmin növləri (tipləri).

**Xətti alqoritmlər** hesablama prosesini ifadə edən bir neçə ardıcıl əməliyyatlardan ibarət olur və onlar yazıldığı ardıcıllıqla da icra olunur. (heç bir şərt və dövr yoxdur)

**Budaqlanan alqoritmlərin** tərkibində bir və ya bir neçə məntiq mərhələsi olur. Bu mərhələdə müəyyən kəmiyyətlərin hər hansı bir şərti ödəyib-ödəmədiyi yoxlanılır və ona uyğun olaraq sonrakı gedişin istiqaməti seçilir. Yəni nəzərdə tutulan şərt ödənilirsə, bir istiqamətə, həmin şərt ödənilmirsə, əks istiqamətə doğru hərəkət edilir. Beləliklə, alqoritmdə budaqlanma baş verir.(şərt var)

**Dövrü alqoritm** hesablama prosesində tez-tez eyni əməliyyatlar qrupunun çoxlu sayda təkrar olunması lazım gəlir. Bu halda dövr alqoritmindən istifadə olunur. Dövrlər sadə və mürəkkəb olur. Sadə dövrlü alqoritmin bir dövrü olur. Əgər hər hansı bir alqoritmdə bir neçə daxili dövr iştirak edirsə, onda belə dövrlərə mürəkkəb dövr deyilir. (dövr var)

Alqoritmin izah növləri (təsvir üsulları).

**Mətn şəkildə (sözlə)**

**Blok-sxem**

**Cədvəl**

**Proqram (kodla)**

1. **Data Types.**

Proqramlaşdırmadan **data type** vacib bir anlayışdır.

Dəyişənlər müxtəlif növ məlumatları saxlaya bilər və müxtəlif növlər fərqli işlər görə bilər.

Python bu kateqoriyalarda defolt olaraq aşağıdakı **data type-lara** malikdir:

**Text Type: str**

**Numeric Types: int, float, complex**

**Sequence Types: list, tuple, range**

**Mapping Type: dict**

**Set Types: set, frozenset**

**Boolean Type: bool**

**Binary Types: bytes, bytearray, memoryview**

Məlumat Tipinin əldə edilməsi

type() funksiyasından istifadə etməklə istənilən obyektin data type-ını əldə edə bilərsiniz:

x = 5  
print(type(x))

>>> <class 'int'>

1. **Operatorlar.**

Python Operators.

Dəyişənlər və qiymətlər üzərində əməliyyatlar yerinə yetirmək üçün operatorlardan istifadə olunur.

Aşağıdakı nümunədə iki dəyəri bir araya toplamaq üçün + operatorundan istifadə edirik:

print (10 + 5)

>>> 15

Operator qrupları.

* Arithmetic operators. (Arifmetik operatorlar)
* Assignment operators. (Təyinat operatorları)
* Comparison operators. (Müqayisə operatorları)
* Logical operators. (Məntiqi operatorlar)
* Identity operators. (Şəxsiyyət operatorları)
* Membership operators. (Üzvlük operatorları)
* Bitwise operators. (Bitwise operatorları)

**Python Arithmetic Operators:**

+ - \* / % \*\* //

**Python Assignment Operators:**

= += -= \*= /= %= //= \*\*= &= |= ^= >>= <<=

**Python Comparison Operators:**

== ! = > < >= <=

**Python Comparison Operators:**

**and** Hər iki ifadə doğrudursa, True qaytarır.

**or** İfadələrdən biri doğrudursa, True qaytarır.

**not** Nəticəni tərsinə çevirin, nəticə doğrudursa, False qaytarır.

**Python Identity Operators:**

**is** Hər iki dəyişən eyni obyektdirsə, True qaytarır.

**is not** Hər iki dəyişən eyni obyekt deyilsə, True qaytarır.

**Python Membership Operators:**

**in** Əgər y-nin x obyektində müəyyən edilmiş dəyəri olan ardıcıllığı varsa True qaytarır.

**not in** Obyektdə müəyyən edilmiş dəyərə malik ardıcıllıq yoxdursa True qaytarır.

**Python Bitwise Operators:**

**& -** AND

**| -** OR

**^ -** XOR

**~ -** NOT

**<< -** Zero

**>> -** Signedrightshift

Operatorların üstünlüyü.

1. Qüvvət
2. Vurma, bölmə
3. Toplama, çıxma
4. **Typecasting.**

Dəyişən Tipini Müəyyən edin.

Dəyişən üçün növ təyin etmək istədiyiniz zamanlar ola bilər. Bu casting ilə edilə bilər. Python obyekt yönümlü bir dildir və buna görə də onun primitiv növləri də daxil olmaqla məlumat növlərini müəyyən etmək üçün siniflərdən istifadə edir.

Beləliklə, python-da casting konstruktor funksiyalarından istifadə etməklə həyata keçirilir:

int() - tam ədəddən, float literalından (bütün onluqları silməklə) və ya sətir literalından (sətir tam ədədi təmsil etməklə) tam ədəd qurur.

float() - tam ədəddən, float literalından və ya sətir literalından float ədədi qurur (sətirin float və ya tam ədədi təmsil etməsi şərti ilə).

str() - sətirlər, tam ədədlər və float literalları daxil olmaqla, müxtəlif məlumat növlərindən sətir qurur.

**Typecasting** 2 cür olur:

**Explicit** (aşkar) və **implicit** (qeyri-aşkar).

Implicit Type Conversion.

Bu üsullarda Python məlumat növünü avtomatik olaraq başqa bir məlumat növünə çevirir.

# Python automatically converts

# a to int

a = 7

print(type(a))

>>> <class 'int'>

Explicit Type Casting.

Bu üsulda Python tələb olunan əməliyyatı yerinə yetirmək üçün dəyişən məlumat növünü müəyyən məlumat növünə çevirmək üçün istifadəçinin iştirakı tələb edir. (int(), float(), str() və s. ilə edir)

1. **Hansı xətalar ola bilər?**

Bəzi xətalar.

**ArithmeticError** Rəqəmsal hesablamalarda xəta baş verdikdə yüksəlir.

**AttributeError** Atribut arayışı və ya təyinatı uğursuz olduqda yüksəldilir.

**Exception** Bütün exceptionlar üçün baza sinfidir.

**ImportError** Import edilmiş modul mövcud olmadıqda yüksəldi.

**IndexError** Ardıcıllığın indeksi mövcud olmadıqda qaldırılır.

**KeyboardInterrupt** İstifadəçi Ctrl+c, Ctrl+z və ya Delete düymələrini sıxdıqda yüksəlir.

**MemoryError** Proqramın yaddaşı bitdikdə yüksəlir.

**NameError** Dəyişən mövcud olmadıqda qaldırılır.

**OSError** Sistemlə əlaqəli əməliyyat xətaya səbəb olduqda yüksəlir.

**OverflowError** Rəqəmsal hesablamanın nəticəsi çox böyük olduqda qaldırılır.

**ReferenceError** Zəif istinad obyekti mövcud olmadıqda qaldırılır.

**SyntaxError** Sintaksis xətası baş verdikdə qaldırılır (mötərizə, dırnaq səhvləri).

**TabError** Abzas nişanlar və ya boşluqlardan ibarət olduqda qaldırılır

**SystemError** Sistem xətası baş verdikdə yüksəlir.

**TypeError** İki müxtəlif növ birləşdirildikdə yüksəlir.

**UnboundLocalError** Təyinatdan əvvəl yerli dəyişənə istinad edildikdə yüksəldilir.

**ValueError** Müəyyən edilmiş məlumat növündə səhv dəyər olduqda yüksəldilir.

**ZeroDivisionError** Bölmədəki ikinci operator sıfır olduqda qaldırılır.

1. **If,Elif,Else.**

Tərif və İstifadə.

**if** açar sözü şərti ifadələr (if ifadələri) yaratmaq üçün istifadə olunur və yalnız şərt True olduqda kod blokunu icra etməyə imkan verir.

Şərt False-dursa kodu yerinə yetirmək üçün **else** açar sözündən istifadə edin.

**else** açar sözü şərti ifadələrdə (if ifadələrində) istifadə olunur və şərt False olduqda nə edəcəyinə qərar verir.

else açar sözündən başqa try...except blokundan da istifadə edilə bilər.

**elif** ifadəsi çoxlu ifadələri yoxlamağa və şərtlərdən biri True olaraq qiymətləndirilən kimi kod blokunu icra etməyə imkan verir.

Else kimi, elif ifadəsinin işlənməsi istəyə bağlıdır. Lakin else-dən fərqli olaraq, if-dən sonra ixtiyari sayda elif ifadəsi ola bilər.

1. **Exception Handling.**

**try** bloku kod blokundakı səhvləri sınamağa imkan verir.

**except** bloku xətanı idarə etməyə imkan verir.

**Finally** bloku try...except bloklarının nəticəsindən asılı olmayaraq kodu icra etməyə imkan verir.

Exception Handling.

Səhv və ya exception olduqda, Python normal olaraq dayanacaq və xəta mesajı yaradacaq.

Bu xətaları try ifadəsi ilə idarə etmək olar:

* The try block will generate an exception, because x is not defined:

try:  
  print(x)  
except:  
  print("An exception occurred")

try bloku xəta yaratdığından, except bloku icra ediləcək.

try bloku olmadan proqram çökəcək və xəta verəcək.

* İstədiyiniz qədər exception blokları təyin edə bilərsiniz, məs. xüsusi növ xəta üçün xüsusi kod blokunu icra etmək istəyirsinizsə:

Print one message if the try block raises a NameError and another for other errors:

try:  
  print(x)  
except NameError:  
  print("Variable x is not defined")  
except:  
  print("Something else went wrong")

* Heç bir xəta olmadıqda icra olunacaq kod blokunu müəyyən etmək üçün else açar sözündən istifadə edə bilərsiniz:

In this example, the try block does not generate any error:

try:  
  print("Hello")  
except:  
  print("Something went wrong")  
else:  
  print("Nothing went wrong")

* The finally block, if specified, will be executed regardless if the try block raises an error or not.

try:  
  print(x)  
except:  
  print("Something went wrong")  
finally:  
  print("The 'try except' is finished")

* Raise an exception

As a Python developer you can choose to throw an exception if a condition occurs.

To throw (or raise) an exception, use the raise keyword.

Raise an error and stop the program if x is lower than 0:

x = -1  
  
if x < 0:  
  raise Exception("Sorry, no numbers below zero")

You can define what kind of error to raise, and the text to print to the user.

x = "hello"  
  
if not type(x) is int:  
  raise TypeError("Only integers are allowed")

1. **While,For,Iterasiya nedir?**

Python Loops.

Python has two primitive loop commands:

While loops and for loops

The while Loop.

With the while loop we can execute a set of statements as long as a condition is true.

Print i as long as i is less than 6:

i = 1  
while i < 6:  
  print(i)  
  i += 1

Note: remember to increment i, or else the loop will continue forever.

The break Statement.

With the break statement we can stop the loop even if the while condition is true:

Exit the loop when i is 3:

i = 1  
while i < 6:  
  print(i)  
  if i == 3:  
    break  
  i += 1

The continue Statement.

With the continue statement we can stop the current iteration, and continue with the next:

Continue to the next iteration if i is 3:

i = 0  
while i < 6:  
  i += 1  
  if i == 3:  
    continue  
  print(i)

The else Statement.

With the else statement we can run a block of code once when the condition no longer is true:

Print a message once the condition is false:

i = 1  
while i < 6:  
  print(i)  
  i += 1  
else:  
  print("i is no longer less than 6")

The For Loop.

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Print each fruit in a fruit list:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)

The range() Function

To loop through a set of code a specified number of times, we can use the range() function. The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Using the range() function:

for x in range(6):  
  print(x)

Break and continue are also used in for loop.

1. **Lokal və global dəyişənlər.**

Global Variables.

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

Create a variable outside of a function, and use it inside the function

x = "awesome"  
  
def myfunc():  
  print("Python is " + x)  
  
myfunc()

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

The global Keyword

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = "awesome"  
  
def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

1. **Funksiya nədir? Hasnı problemləri həll edir?**

Python functions.

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

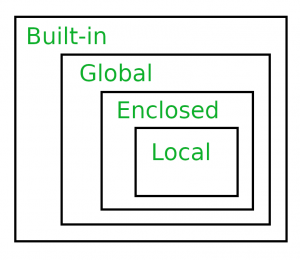
A function can return data as a result.

Funksiya həər hansı bir işi (və ya bir neçə işi) görən kod blokudur.

Artıq kod yazmağın qarşısını alır. Strukturluq qazandırır, oxunarlığı artırır.

* Funksiya bir işi görməlidir. Belə funksiyalar ideal funksiyalar adlanır.
* Funkjsiyanın adı və parametrləri (argumentləri)gördüyü işə uyğun olmalıdır.
* Funksiyaların üzərində nə işi gördüyü barəsində comment(#) olmalıdır.
* Funksiya daxilində input, print yazılmamalıdır.
* Parametr sıraya görə tanınır, ada görə yox.
* Return break kimidir, funksiyadan çıxarır.

1. **LEGB scope nədir ?**



1. **Lambda function.**

Python Lambda.

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

*lambda arguments : expression*

The expression is executed and the result is returned:

* Add 10 to argument a, and return the result:

x = lambda a : a + 10  
print(x(5))

* Lambda functions can take any number of arguments:

Multiply argument a with argument b and return the result

x = lambda a, b : a \* b  
print(x(5, 6))

* Summarize argument a, b, and c and return the result:

x = lambda a, b, c : a + b + c  
print(x(5, 6, 2))

1. **map, filter, zip, reduce ?**

These functions can save you a lot of time when working with iterables ,e.g Sets, Lists, Strings etc.

Qeyd: Assosativ massivlər içində bir neçə listlər olan listdir.

Filter.

Müəyyən şərtlərə cavab verən dəyərləri çıxarmaq üçün filtrdən istifadə edin.

Misal

Siyahıdakı bütün cüt dəyərləri qaytarmaq istədiyirsiniz.

arr = [ 32,11,55,21,9]

normal = map( lambda x: (x - min(arr))/ (max(arr) -min(arr)), arr)

normal = list(normal)

// Output: [0.5, 0.043478260869565216, 1.0, 0.2608695652173913, 0.0]

Qeyd edək ki, bu halda biz normallaşdırma düsturumuzu siyahıdakı hər bir elementə tətbiq edən lambda funksiyasından istifadə etdik.

Map.

Elementlər toplusunun üzərində gəzərək dəyişiklik edib qaytarır.(update etmək üçündür)

ZIP.

Zip ilə siz birdən çox təkrarlananları bir məlumat strukturunda birləşdirə bilərsiniz

Misal

Oyunçu adlarını son iki oyundakı xallarına uyğunlaşdırın.

names = [ “Ja”, “Beel”, “Madison”]

last\_game = [1,4,9]

previous\_game = [3,4,5]

player\_performance = list( zip( names, last\_game, previous\_game ) )

// Output: ('Ja', 1, 3), ('Beel', 4, 4), ('Madison', 9, 5)]

Eyni indeksə malik elementlər birlikdə qruplaşdırılır.

Reduce.

Reduce allows you to perform cumulative tasks on your iterables.

Example

Calculate the product of all entries in a list.

list\_b = [ 1,3,2,5]

product = reduce( lambda a, b: a\* b , list\_b)

// product >> 30

1. **Containers nədir?**

Konteyner başqa bir şey saxlayan edən bir şeydir.

Məsələn, listlər, tupple-lar, dictionary-lər, setlər və.s.

1. **Massiv anlayışı.**

Massiv – bir adla adlandırılmış nizamlanmış kəmiyyətlər yığımıdır. Massivin hər bir elementinə müraciət onun öz nömrəsi ilə olur. Yəni hər bir elementin öz nömrəsi var. Bu nömrəyə massivin indeksi deyilir. Hər elementin massivdə vəziyyəti onun indeksi ilə müəyyən olunur. Buna görə də massiv nizamlanmış olur.

1. **Module nədir ?**

Modul başqa proqramçının yazdığı hazır kod blokudur.

Məsələn, datetime, os və.s.

1. **String nədir ?**

Strings.

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the print() function.

Multiline Strings.

You can assign a multiline string to a variable by using three quotes:

You can use three double quotes:

a = """Lorem ipsum dolor sit amet,

consectetur adipiscing elit,

sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua."""

print (a)

1. **String metodları?**

**capitalize()** Converts the first character to upper case

**count()** Returns the number of times a specified value occurs in a string

**encode()** Returns an encoded version of the string

**endswith()** Returns true if the string ends with the specified value

**find()** Searches the string for a specified value and returns the position of where it was found

**index()** Searches the string for a specified value and returns the position of where it was found

**isalpha()** Returns True if all characters in the string are in the alphabet

**isdecimal()** Returns True if all characters in the string are decimals

**isdigit()** Returns True if all characters in the string are digits

**isprintable()** Returns True if all characters in the string are printable

**isspace()** Returns True if all characters in the string are whitespaces

**istitle()** Returns True if the string follows the rules of a title

**isupper()** Returns True if all characters in the string are upper case

**lower()** Converts a string into lower case

**replace()** Returns a string where a specified value is replaced with a specified value

**rfind()** Searches the string for a specified value and returns the last position of where it was found

**rindex()** Searches the string for a specified value and returns the last position of where it was found

**split()** Splits the string at the specified separator, and returns a list

**splitlines()** Splits the string at line breaks and returns a list

**startswith()** Returns true if the string starts with the specified value

**strip()** Returns a trimmed version of the string

**title()** Converts the first character of each word to upper case

**translate()** Returns a translated string

**upper()** Converts a string into upper case

1. **OOP nedir, prinsipləri?**

Object-Oriented Programming (OOP) language

(Obyekt yönümlü proqramlaşdırma dili)

Obyekt yönümlü proqramlaşdırma (OOP) dili obyektlərə əsaslanır. Bu proqramlaşdırma dilində proqramlar obyekt adlanan kiçik hissələrə bölünür. Proqramı təkrar istifadə edilə bilən, səmərəli və istifadəsi asan etmək üçün proqramda encapsulation, inheritance, polymorphism, abstraction kimi tikinti blokları (building blocks) (prinsipləri) vardır.

Enkapsulyasiya.

Enkapsulyasiya obyektin öz məlumatlarını və metodlarını gizlətmək üçündür. 3 Access modifier-i var: Public, Protector, Private.

Public (.) – Belə infoya rahat çatılır.

Protector (.\_) - Yalnız class və törəmə class daxilində görsənir.

Private (.\_\_) – Belə infoya çatmaq üçün getter-lərdən(nispektorlar)

istifadə olunur. Getter-lər infoya dolayı yolla çatmaq üçündür.

Və setter-lər (modifikatorlar) var. Onlar datanı müəyyən şərtlərlə set etmək üçündür. (datanı yoxlayır, sonra yazır)

İnheritance.

İnheritance törəmə deməkdir, bir class digər classdan törənə bilir.

Törənən class törəndiyi classın bütün əlamətlərini daşıyır. Kod artıqlığının qarşısını alır, strukturuq qazandırır, oxunarlığı artırır.

Derived class (subclass, child class) base classdan (super class, parent class) törənir.

Note: Every derived class is base class, but every base class is not derived class.

Polimorfizm.

(Çoxformalılıq deməkdir)

Polimorfizmin faydası • Polimorfizmin əsas məqsədi daha az kod ilə daha üstün məhsuldarlıq əldə etməkdir. Yazılan kodun oxşar xassələrə malik bir neçə sinif üçün təkrarlanmadan ümumi yazılmasına nail olmaq.

“” Poly-çoxlu, morph-forma, davranış deməkdir. Bir neçə obyekt arasından ən düzgün olanın seçilməsində polymorphism method-dan istifadə edilir. Bir az lori dildə izah edim. Fərz edin ki şirkətdə işləyirsiniz və nəqliyyat departamentinə zəng edib deyirsiniz ki, şirkətin rəisinə digər şirkət tərəfindən orta həcmdə heykəl hədiyyə edilib və rəis deyir ki olduğunuz yerə heykəli daşıya biləcək minik vasitəsi göndərilsin. Şirkətin balansında olan minik vasitələri sinfində sedan, SUV, yük maşını və traktor var. Polymorphism bu yerdə işə düşür. Əsas dəyərlər minik vasitəsinin seçilməsi, sonra orta həcmdə yük daşıya bilən avtomobilin seçimini həyata keçirir. Daha orta həcmli heykəl üçün traktor göndərmir : ) ””

Statik Metod.

Classla bağlı infonu obyekti yaratmadan götürməyə imkan verir.

Statik metod sinfin obyektinə deyil, sinfə bağlıdır.

1. **Fayllar.**

File.

Fayllarla işləmə hər hansı bir veb tətbiqinin vacib hissəsidir.

Python faylları yaratmaq, oxumaq, yeniləmək və silmək üçün bir neçə funksiyaya malikdir.

File Handling

The key function for working with files in Python is the open() function.

The open() function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file:

**"r" - Read - Default value.** Opens a file for reading, error if the file does not exist

**"a" - Append** - Opens a file for appending, creates the file if it does not exist

**"w" - Write** - Opens a file for writing, creates the file if it does not exist

**"x" - Create** - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

**"t" - Text** - Default value. Text mode

"**b" - Binary** - Binary mode (e.g. images)

1. Ələvə sual - Konstruktor nədir?

Konstruktor obyekt yaradılan zaman çağırılan metoddur. Bu üsul sinifdə müəyyən edilmişdir və əsas dəyişənləri işə salmaq üçün istifadə edilə bilər.