Python programazio-lengoaiaren hastapenak

Lanbide Heziketako eta Batxilergoko ikasleentzako programazioaren oinarriak

EGILEA: EIDER GALARRAGA SOLABARRIETA

Eibarko IRALE

R400 IKASTAROA, 2017-2018

AURKIBIDEA

ATARIKOA

ATARIKOA

Irakurle:

Hemen aurkituko duzun lan honek oinarrizko programazioan hasierako pausoa emateko tresna izan nahi du. Oinarrizko programazioan murgiltzeko lehen pausoak emateko hainbat lengoaia dauden arren, Python programazio-lengoaia aukeratu dut, batez ere Pythonek eskaintzen dituen sinpletasunagatik eta ulergarritasunagatik. Bestalde, euskarazko material-gabeziaz ohartuta, hutsune hori betetzen laguntzeko asmoz ere idatzi dut hemen duzun txostena.

Hainbat mailatan erabiltzeko moduko lana dela uste dut. Lehenik, Batxilergoko Informatika ikasgaia ikasten dabiltzan ikasleentzat eta euren irakasleentzat. Izan ere, oinarrizko programazioa irakastea eskatzen du Curriculumak, eta, sarritan, ez da jakiten ikasgai hori emateko zer programazio-lengoaia erabili, eta horretarako dauden baliabideak ere urriak dira. Lan honek aukera bat gehiago eskaini nahi du Currilumari erantzuteko orduan.

Bigarrenik, Web Aplikazioen Garapeneko Goi Mailako Teknikaria titulua lortzeko ikasketetan ari diren ikasle eta irakasleentzat ere pentsatua dago, zehazki, 1. mailako Programazioa modulurako.

Lana bi atal nagusitan dago banatuta. Alde batetik, kontzeptu teorikoak lantzeko atala dago, eta bestetik, kontzeptu teoriko horiek praktikan jartzeko proposatutako ariketez osatutako atala.

Atal teorikoari dagokionez, hasieran, programazioaren oinarrizko kontzeptuak azaltzen dira: algoritmoak, programazioa, aplikazio baten bizi-zikloak eta programazio-lengoaien mailak, besteak beste. Ondoren, Pythoni buruzko oinarrizko ezagutzak zehazten dira, haren definizioa eta ezaugarriak. Gainera, horiekin batera, txosten hau diseinatzean, Pythonen lan egiteko aukeratu den lan-ingurunea ere azaltzen da: **Repl.it** ingurunea. Atal teorikoarekin bukatzeko, programazioan ezinbestekoak diren elementuak eta haien funtzionamendua deskribatzen dira: aldagaiak, kontrol-egiturak eta funtzioak.

Atal praktikoa, berriz, ariketaz osatutako zazpi azpiataletan banatzen da, eta azpiatalak honela multzokatu daitezke. Lehenengo, algoritmoen ariketak daude, programa bat egiteko ezinbestekoak baitira; alegia, oinarrizko ariketa da paperean edo buruan argi izatea zer

algoritmori jarraitu behar zaion, eta ariketek hori bideratzen laguntzen dute. Ondoren, Python programazio-lengoaian hasierako programak idazteko oinarrizko aginduei buruzko ariketak aurki daitezke. Eta, jarraian, zerrendei buruzkoak. Kontrol-egituretan murgiltzeko, baldintza eta begiztei buruzko ariketak multzokatzen dira ondoren, eta, jarraian, erabiltzaileak Pythonen funtzioak definitzeko ariketak. Alde praktikoarekin bukatzeko, “Askotariko ariketak” izeneko multzoa dator; Pythonen programatzeko ariketa orokorrak dira, eta ordura arte ikasitako guztia praktikan jartzeko balio dute. Ohar gisa esan kontzeptu teorikoak landu ahala dagozkien ariketak egitea gomendatzen dela.

1. ZER DA PROGRAMAZIOA?

Unitate honetan programazioaren oinarrizko kontzeptuak aztertuko dira besteak beste, zer den programazioa, zer den algoritmoa, nola gauzatu algoritmoen adierazpena eta, unitatea bukatzeko, aplikazio baten bizi-zikloaren nondik norakoak.

Hainbat programazio-legoaia daude, eta horietariko bat da Python programazio-lengoaia. Informatikan, programazioaz ari garenean funtzio oso zehatzaz ari gara, eta honela azaldu daiteke zein den programazioaren funtzio hori: erabiltzaileak emandako aginduak exekutatzen ditu ordenagailuak, eta, ondoren, agindu horien azken emaitza zein den adierazten du. Laburbilduz, esan daiteke programa bat agindu-segida ordenatu bat dela eta aipatutako agindu-segida horiek antolatzeko algoritmoak erabiltzen direla.

1.1. ZER DA ALGORITMOA?

Algoritmoa da arazo bat ebazteko edo ataza bat aurrera eramateko eman beharreko pausoak deskribatzen dituen prozesua, eta hainbat elementu izan ditzakete; esaterako, sekuentziak, erabakiak eta iterazioak.

Askotan arazo bat ebazteko algoritmo bat baino gehiago izaten da; horrelakoetan, exekuzio-denbora izaten da kontuan hartu beharreko ezaugarria.

Arazoa ebazteko aukera onena aztertutakoan, algoritmoa sortzen da, eta algoritmo hori nola adierazi ere pentsatu behar izaten da. Horretarako, hiru bide hauek daude:

1. Norberaren hizkuntzan idaztea.
2. Sasikodean idaztea.
3. Fluxu-diagramen bidez adieraztea.

Hona hemen, euskaraz idatzitako algoritmo batzuen adibideak:

1.1.1. adibidea: bete baso bat urez

Idatzi baso bat urez betetzeko jarraitu beharreko algoritmoa, eta, ondoren, konparatu azpiko adibidearekin:

1. Hartu basoa.
2. Ireki iturria.
3. Jarri basoa iturri azpian.
4. Itxaron basoa bete arte.
5. Erretiratu basoa iturritik.

1.1.1. irudia

1.1.2. adibidea : kalkulatu noten batezbestekoa

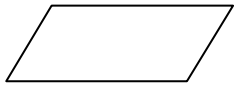
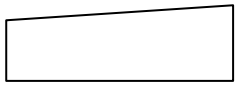
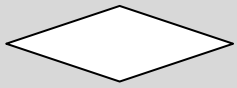
Idatzi hiru ikasgaitako noten arteko batezbestekoa kalkulatzeko algoritmoa, eta, ondoren, konparatu azpiko adibidearekin:

1. Gorde A ikasgaiko nota a_nota izeneko aldagaian.
2. Gorde B ikasgaiko nota b_nota izeneko aldagaian.
3. Gorde C ikasgaiko nota c_nota izeneko aldagaian.
4. Gorde a_nota, b_nota eta c_nota aldagaien arteko batura batura_abc izeneko aldagaian.
5. Gorde batezbestekoa _abc izeneko aldagaian, batura_abc/3 eragiketaren emaitza.

1.1.2. irudia

1.2. ALGORITMOEN ADIERAZPENA: FLUXU-DIAGRAMAK

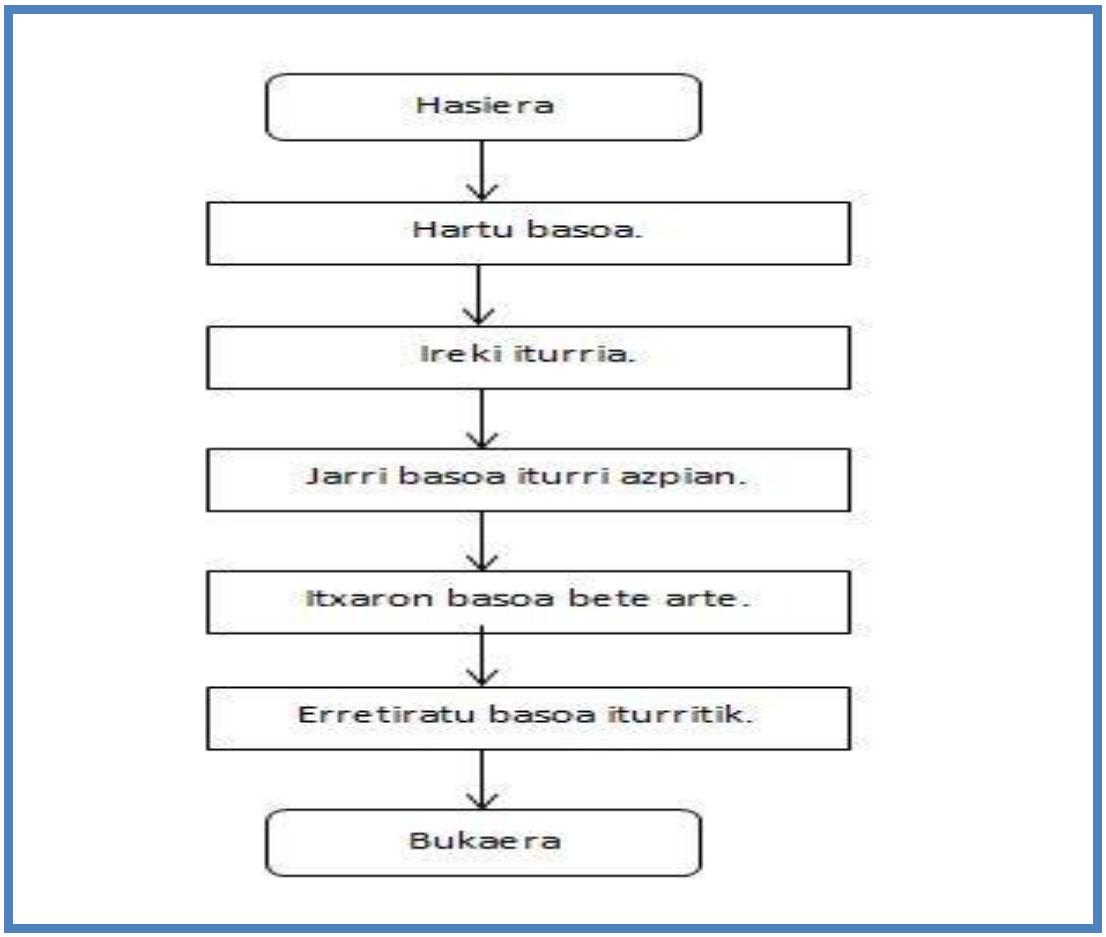
Hona hemen fluxu-diagrama egiteko erabili beharreko elementuen izena, ikurra eta funtzioa zehazten dituen taula.

IZENA	IKURRA	FUNTZIOA
Terminala		Diagramaren hasiera eta bukaera adieraztea.
Sarrera/Irteera		Informazioaren sarrera edo irteera sinplea adieraztea.
Prozesua		Informazioarekin edozein kalkulu edo eragiketa adieraztea.
Inprimagailura irteera		Informazioa inprimagailura bidaltzea.
Pantailara irteera		Informazio pantailan agertzea.
Teklatua		Teklatu bidez datuak eskuz sartzea.
Erabakia		Eragile logikoak edo konparazioak adieraztea.
Konektoreak		Diagrama bateko elementuak lotzea edo orriak lotzea.
Fluxu-geziak		Informazio-fluxuaren noranzkoa adieraztea.

Interesgarria da 1.1.1. adibidea eta 1.1.2 adibidea hartu eta fluxu-diagramen bidez nola adierazten diren aztertzea. Hona hemen bi adibide posible:

1.2.1. ariketa eta adibidea: bete baso bat urez

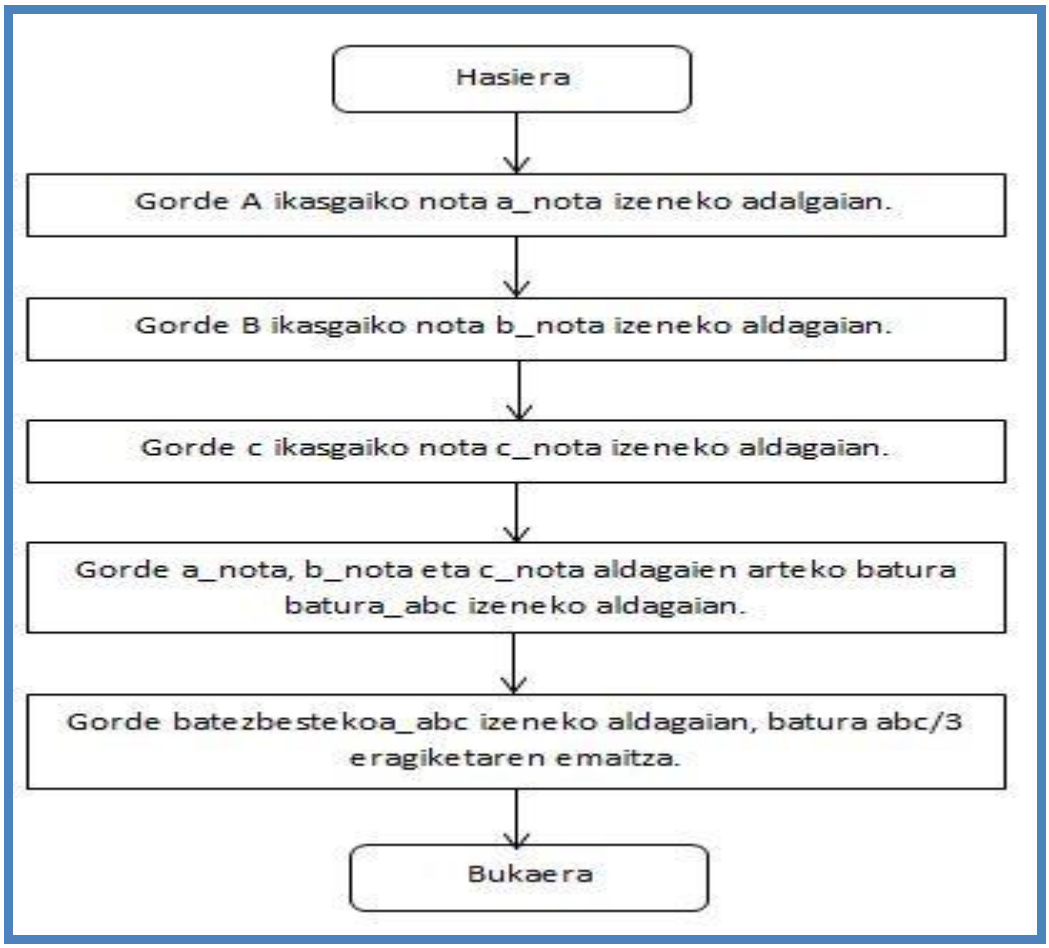
Idatzi baso bat urez betetzeko jarraitu beharreko algoritmoaren fluxu-diagrama, eta, ondoren, konparatu azpiko adibidearekin:



1.2.1. irudia

1.2.2. ariketa eta adibidea: kalkulatu noten batezbestekoa

Idatzi hiru ikasgaiko noten arteko batezbestekoa kalkulatzeko algoritmoaren fluxu-diagrama, eta, ondoren, konparatu azpiko adibidearekin:



1.2.2. irudia

1.3. APLIKAZIOEN BIZI-ZIKLOA

Programazioa, programa eta algoritmoak zehaztu ondoren, komeni da aplikazio baten bizi-zikloa zer den argitzea, aplikazioek ere bizi-itxaropena izaten dutelako.

Aplikazio informatikoen bizi-ziklo bat softwarea garatzeko jarraitu beharreko pausoak deskribatzeko modua da, beharrizan batetik abiatuta arazoak ebatzi arte irauten duena.

Askotariko aplikazioen bizi-zikloak dauden arren, hauek dira erabilienak: turrusta-eredua, espiral-eredua, prototipo bidezko garapena, garapen iteratiboa eta inkrementala. Hala ere, azkenaldian sortutako bizi-ziklo berriak indar handia ari dira hartzen.

Dena den, guztien artean oinarrizkoena teilakatutako bizi-zikloa da; horrelakoetan, honako pauso hauek eman behar dira:

1. **Analisia**. Ebatzi beharreko arazoaren inguruko informazioa batu, eta **zer** ebatzi zehazten da.

2. **Diseinua.** Ebatzi beharreko arazoaren inguruan behar adina informazio jasotakoan, arazo hori **nola** ebatzi erabakitzen da.

3. **Kodetzea**. Programaren diseinua kodetu egin behar da; hau da, aukeratutako programazio-lengoaiara itzuli behar da ordenagailuak uler dezan.

4. **Probak.** Programaren kodetzea bukatutakoan, ezinbestekoa da programaren funtzionamendu egokia probatzea. Horretarako, bermatu behar da programak eskatutakoa ongi ebazten duela egoera guztietan.

5. **Dokumentazioa.** Programak luzaro funtzionatzeko eta programa egoki mantentzeko, beharrezkoa da programaren nondik norakoak zehatz-mehatz idaztea eta dokumentatzea.

6. **Softwarea mantentzea**. Programa instalatu eta martxan jarri ondoren, ezinbestekoa da aldiro programa eguneratzea zaharkituta gera ez dadin.

2. PROGRAMAZIO-LENGOAIEN MAILAK

Programazioaren oinarrizko definizioak aztertu dira lehenengo puntuan, baina zer motatako programazio-lengoaiak daude programak idazteko?

Informatikan, programazio-lengoaia asko dago: makina-lengoaia, mihiztadura-lengoaia, programazio-lengoaia… Gainera, lengoaia horiek behe-mailako lengoaien edo goi-mailako lengoaien multzoetan sailkatu daitezke.

2.1. BEHE-MAILAKO LENGOAIAK

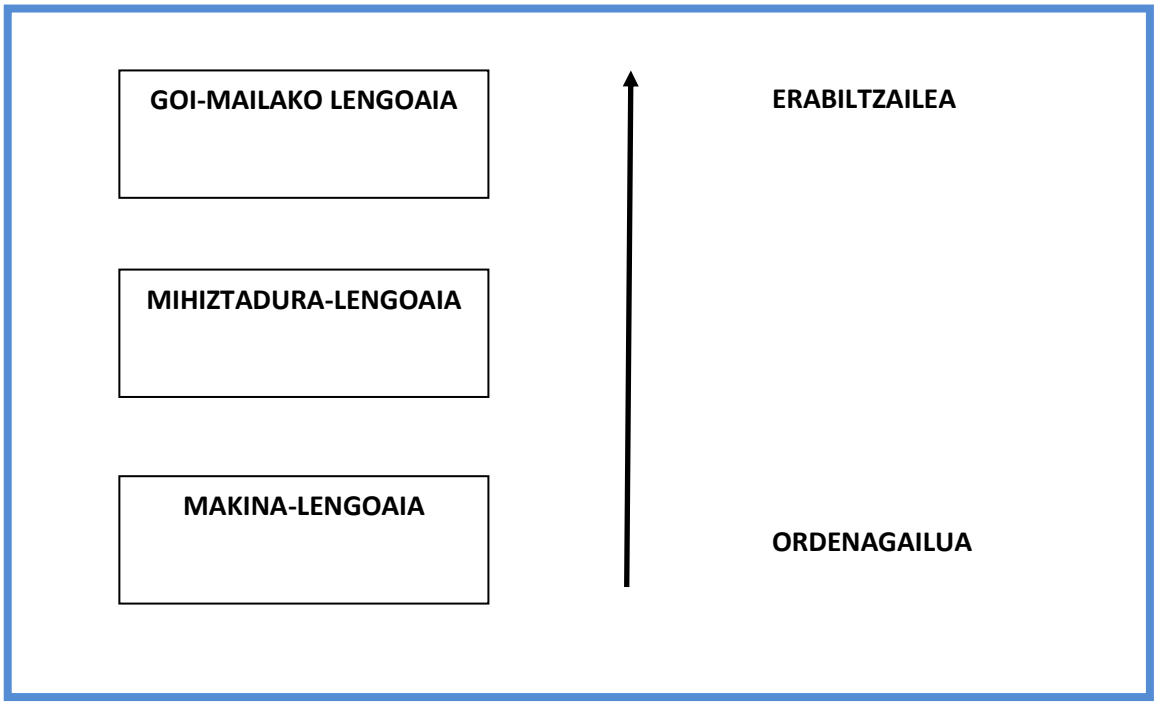
Behe-mailako programazio-lengoaiak zuzenean eragiten dio hardwareari, hau da, ordenagailuaren elementu fisikoen funtzionamenduari. Beste era batera esanda, ordenagailuak zuzenean, bitartekaririk gabe, ulertzen duen lengoaia da behe-mailako lengoaia.

Behe-mailako lengoaia kode bitarrez osatuta dagoen zenbaki-sistema bat da (bitak dira informazio-unitate txikienak). Bit bakoitzak 0 edo 1 balio logikoak izan ditzake, hau da, *off* (gezurra) edo *on* (egia) balioak. Horrela, bitez osatutako kateak sortzen dira, eta kate horiek ordenagailuarentzako instrukzioak adierazteko erabiltzen dira. Izan kontuan sistema bitarrez sortutako instrukzio-kateei behe-mailako lengoaia deitzen zaien arren ez dutela goi-mailako lengoaiek baino garrantzi gutxiago.

2.2. GOI-MAILAKO LENGOAIAK

Ordenagailuak ez ditu zuzenean ulertzen goi-mailako lengoaian jasotako aginduak. Hori dela eta, goi-mailako lengoaiatik behe-mailako lengoaiara itzuli behar da ordenagailuak mezua uler dezan. Horretarako, goi-mailako lengoaia ulergarri izateko honako pauso hauek eman behar dira:

1. Goi-mailako lengoaia mihiztadura-lengoaia bihurtu.
2. Mihiztadura-lengoaia makina-lengoaia bihurtu.



Goi-mailako lengoaiak erabiltzaileentzat errazagoak izaten dira, giza hizkuntzaren antzekoagoak direlako eta ez delako beharrezkoa ordenagailuaren ezaugarri fisikoak ezagutzea. Python, PHP, Java, Perl, C++ eta C# lengoaiak, esaterako, goi-mailako lengoaiak dira.

3. PYTHONEN OINARRIZKO EZAGUTZAK

Atal honetan, Pythonen oinarrizko ezagutzak aztertuko dira: definizioa eta ezaugarriak, zergatik erabili Python, nola jarri martxan Pythonen lan egiteko **Repl.it** ingurunea eta Pythonen egindako lehenengo programa.

3.1. DEFINIZIOA ETA EZAUGARRIAK

Python programazio-lengoaia bat da, eta Guido van Rossumn Herbehereetako konputazioan adituak asmatu zuen 80 hamarkada bukaeran. Perl programazio-lengoaiaren antzekoa den arren, sintaxi garbiagoa eta irakurterrazagoa du Python programazio-lengoaiak. Gainera, beste ezaugarri orokor batzuk ere baditu, honako hauek, esaterako:

- Python **interpretatutako lengoaia** da. Bitarteko programa bat –interpretea– erabilita exekutatzen den programari esaten zaio interpretatutako lengoaia. Konpilatutako lengoaiak baino malguagoak eta eramangarriagoak dira lengoaia-mota hauek.

- Lengoaia **dinamikoa** da. Pythonek ez du definitu behar zein den aldagai baten datu mota; izan ere, exekuzio-denboran esleitzen da aldagaiak erabiliko duen datu mota (ikus 4. puntua). Hau da, programatu bitartean, aldagaiaren datu mota aldatzeko aukera izaten da programaren edozein lerrotan. Horretarako, nahikoa da aldagaiari beste datu mota bateko balioa esleitzea.

- **Askotariko plataformetan** erabili daiteke. Pythonentzako interpretea plataforma askotan dago erabilgarri: UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS… Bestalde, plataforma bakoitzak liburutegi espezifikoak ditu baliabide gisa. Hala ere, liburutegi horiek erabiltzen badira, gerta liteke arazoak sortzea idatzitako iturburu-kodeak gainontzeko plataformetan exekutatzean. Horregatik, askotariko plataformetan erabiltzeko programak sortu nahi badira, egokiagoa da aipatutako liburutegi espezifikoak ez erabiltzea bestela, iturburu-kode horiek egokitu egin beharko dira gainerako plataformetan erabiltzeko.

- **Objektuei zuzenduta** dago. Pythonen, kontzeptu errealak –eguneroko objektu arruntak– izaten dira programazioan planteatutako arazoetako objektuak. Programa egikaritzeko, definitutako objektuen arteko elkarrekintza egon behar du.

Ezaugarri horiez guztiez gain, Pythonek baimendu egiten ditu agindu bidezko programazioa, programazio funtzionala eta objektuei orientatutako programazioa.

3.2. ZERGATIK ERABILI PYTHON?

Informatika-arloan eta webguneen garapenean diharduen orok ezagutu behar du Python programazio-lengoaia. Sintaxi sinplea eta argia duenez, sasikodea dirudi. Ondorioz, programatzen ikasi behar duenarentzat programazio-lengoaiarik egokienetariko bat da. Gainera, Pythonekin programa bat sortzea erraza, azkarra eta dibertigarria da: memoria erraz kudeatzen du, liburutegi-kopuru handia du eskura eta potentzia handiko lengoaia du. Horregatik guztiagatik, munduko hainbat enpresak Phyton erabili izan dute behin edo behin: Google-k, Yahoo-k, Nasa-k eta Industrias Light & Magic-ek, besteak beste.

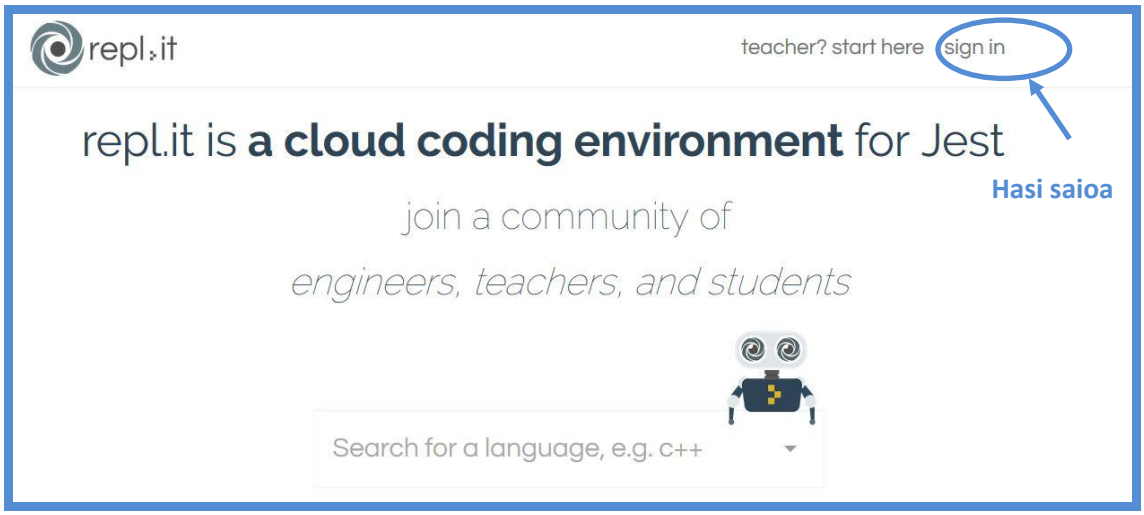
3.3. REPL.IT: LAN-INGURUNEA

Aurretik esan bezala, Python interpretatutako lengoaia da, eta, beraz, Pythonekin programatzeko, ezinbestekoa da kodea idatzi ondoren programa exekutatzeko interprete bat izatea. Izan kontuan Linux sistema-eragileak Python lehenetsita duela, ez ordea Windows-ek.

Pythonen programatzeko, garapen-ingurune bat instalatzea da ohikoena gaur egun. Hala ere, hodei-konputazioak ere eskaintzen du oinarrizko egiturak egiteko aukera.

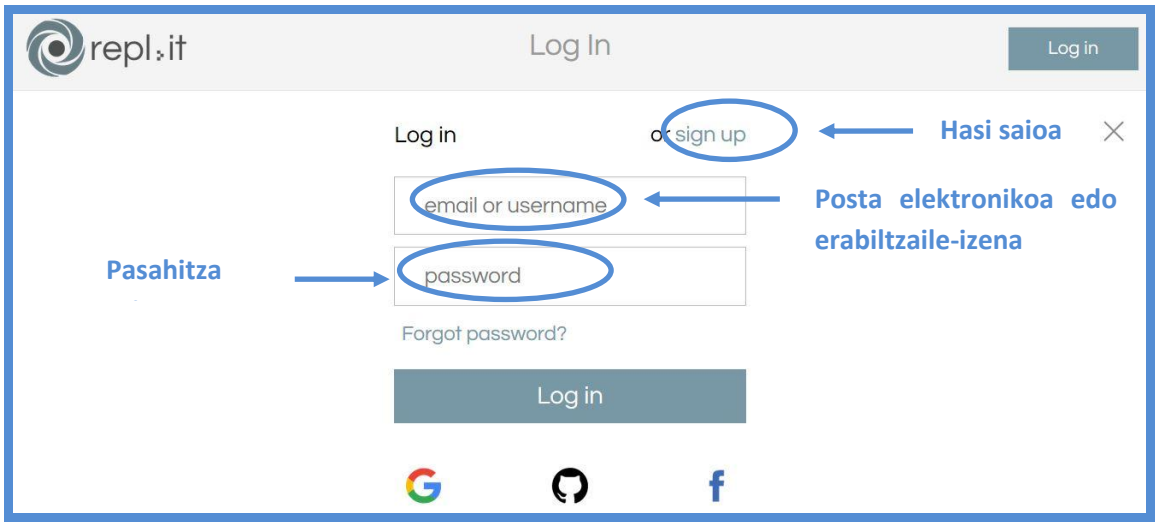
Ingeniari, irakasle eta ikasleentzat garatutako ingurunea da **Repl.it**. Hodeiko kodetze-ingurunea da eta horrek eskaintzen du instalaziorik gabe hodeian lan egiteko aukera. **Repl.it** ingurunean programatzeko, honako pauso hauei jarraitu behar zaie:

1. Sartu https://repl.it/ webgunean. (**Adi**! Ingelesez bakarrik dago).



3.3.1. irudia

2. Hurrengo pausoa da **repl.it** ingurunean nork bere kontua sortzea; horretarako, egin klik **Sign in** botoian.

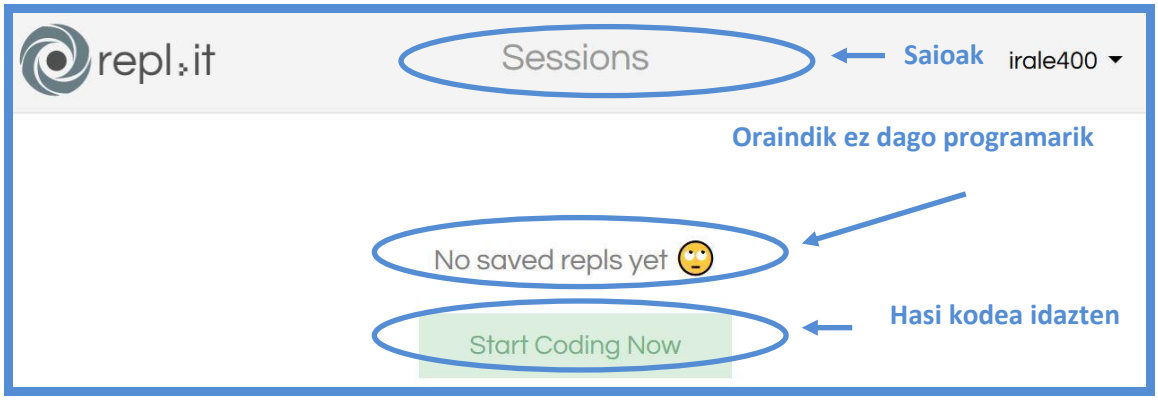


3.3.2. irudia

3. Idatzi erabiltzaile-izena, posta elektronikoa eta pasahitza.

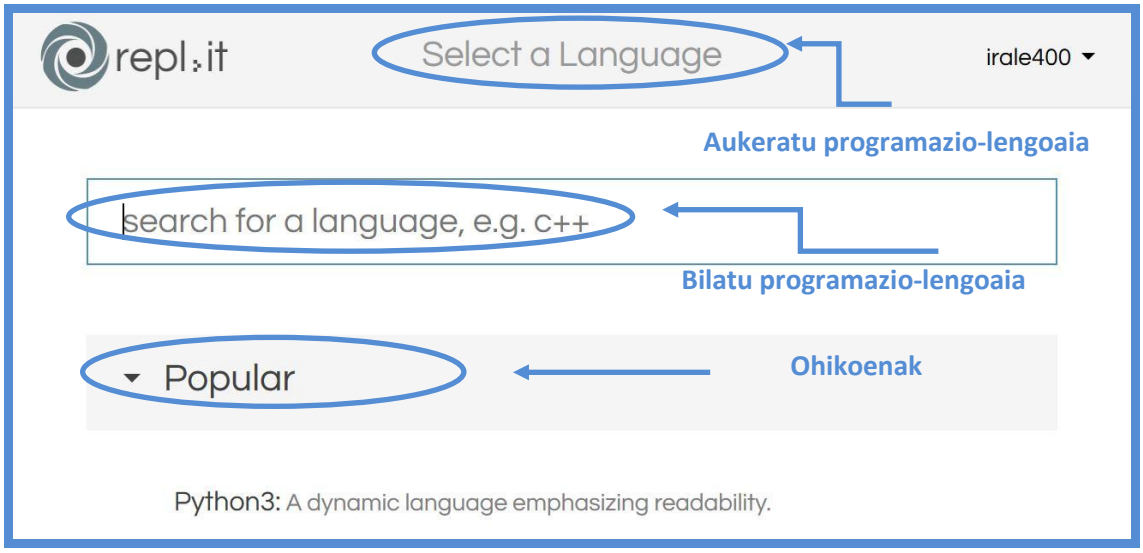
3.3.3. irudia

4. Saioaren barruan zaude. Oraindik saioa hutsik dago, ez baitago proiekturik gordeta.



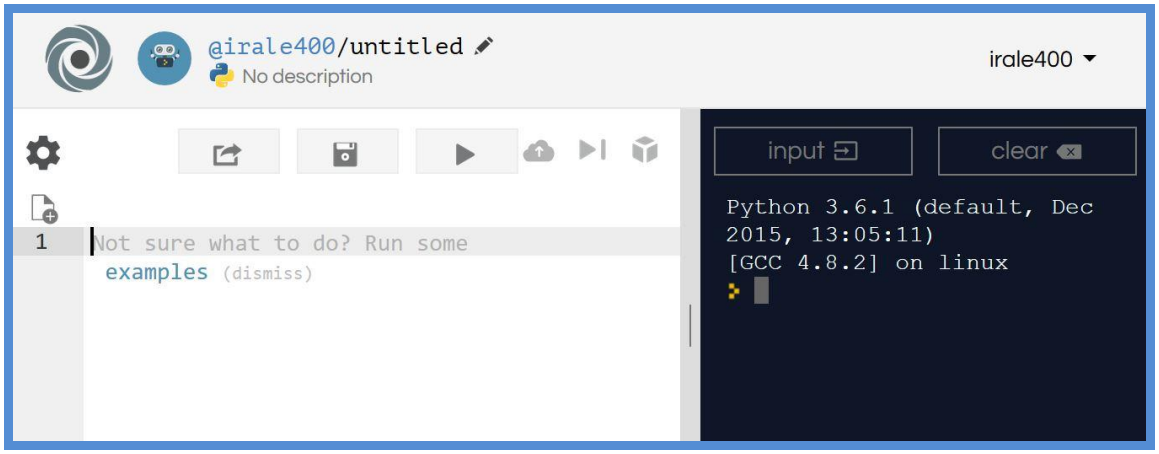
3.3.4. irudia

5. Aukeratu programatzeko erabiliko den programazio-lengoaia. Kasu honetan, aukeratu Python3. Izan kontuan hemendik aurrera erabiliko den sintaxia Python3 lengoaiari dagokiona dela.



3.3.5. irudia

6. Dagoeneko prest dago programak idatzi eta exekutatzeko ingurunea.

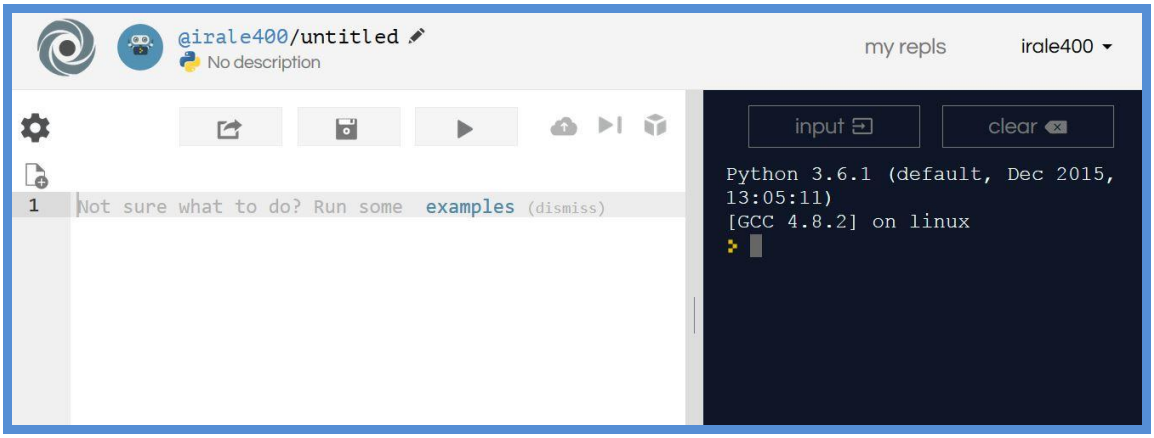


3.3.6. irudia

3.4. LEHENENGO PROGRAMA

Programazioaren alorrean ohikoa den moduan, lehen eginbeharra 'Kaixo, Mundua' agur-mezua pantailan bistaratzea izango da, eta, horretarako, honako pauso hauei jarraitu behar zaie:

1. Prestatu lan-ingurunea eta kargatu **Repl.it**.

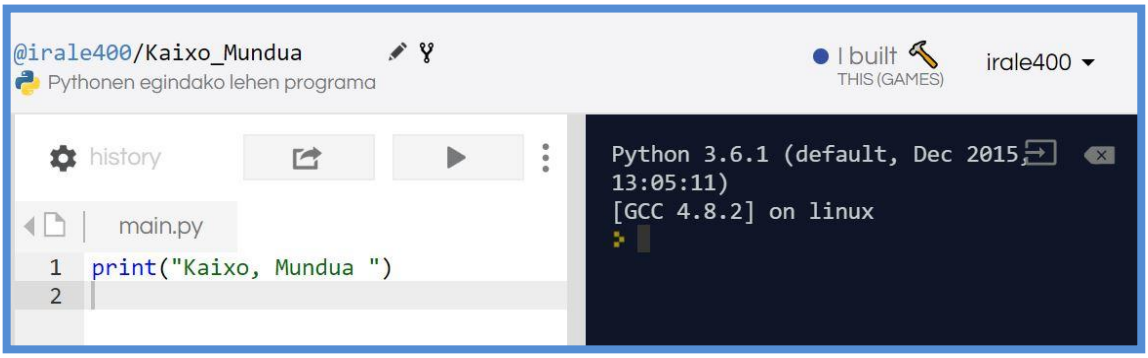


3.4.1. irudia

2. Zehaztu programaren izena eta deskribapena eta egin klik **save** botoian.

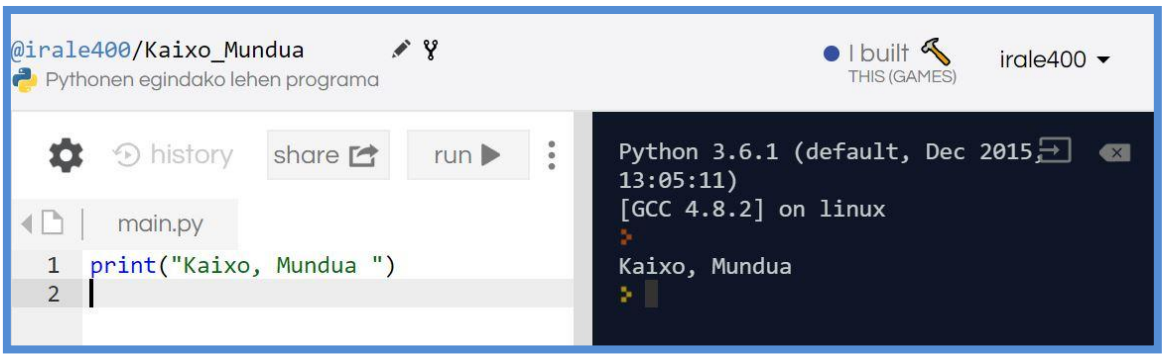
3.4.2. irudia

3. Idatzi honako agindu hau:



3.4.3. irudia

4. Jarri exekuzioa martxan, **play** botoian edo **Ctrl + Enter** botoian klik eginez.

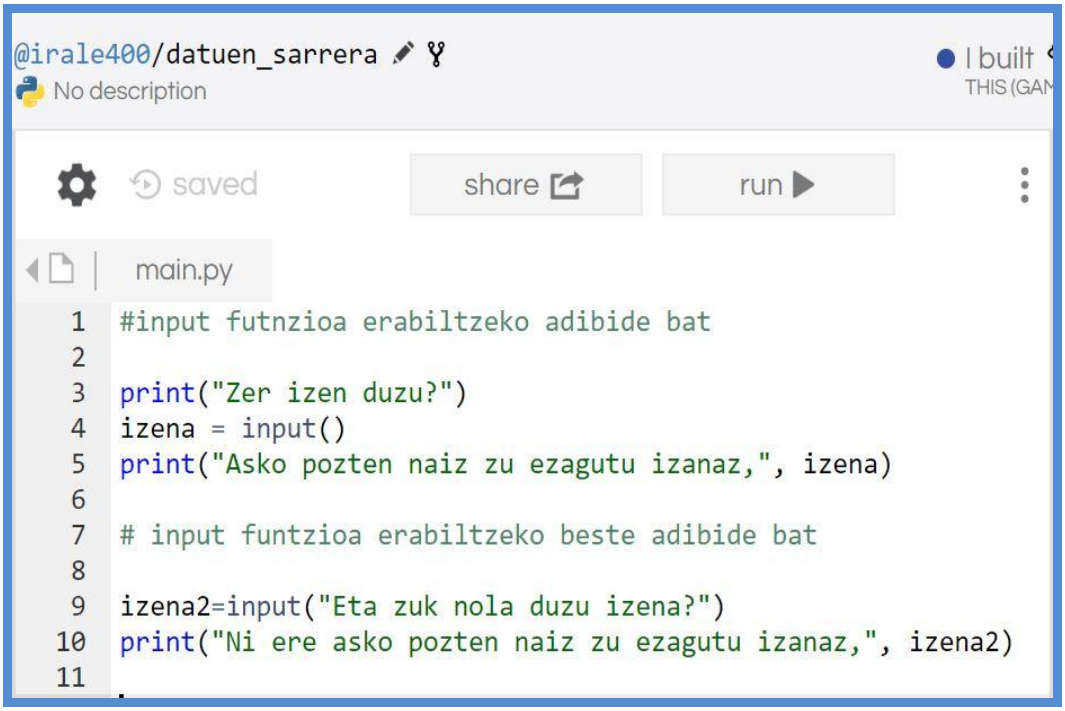


3.4.4. irudia

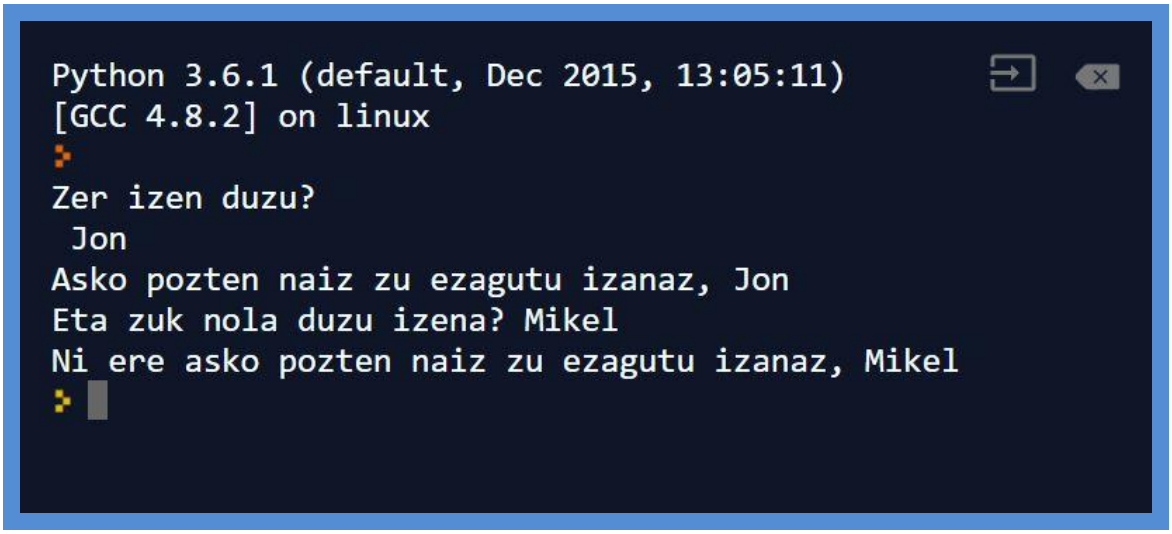
3.5. PROGRAMEN OINARRIZKO AGINDUAK

Edozein programazio-lengoaiatan programatzean, oinarrizko hiru agindu mota erabiltzen dira. Lehenengoa, teklatu bidez datuak edo balioak jasotzeko balio duen agindu mota. Bigarrena, aldagaien balioak (ikus 4. atala) esleitzen dituena. Azkena, programak lortu dituen emaitzak edo balioak pantaila bidez bistaratzen edo inprimatzen dituena.

1. Teklatu bidez datuak edo balioak jasotzeko, `input` funtzioa erabiltzen da eta, programan lortutako emaitzak edo balioak pantaila bidez inprimatzeko, berriz, `print` funtzioa. Pythonek lehenetsita ditu bi funtzio horiek. Hona hemen oinarrizko agindu bi horiek azaltzen dituen adibide txiki bat, baita exekuzioaren emaitza ere:



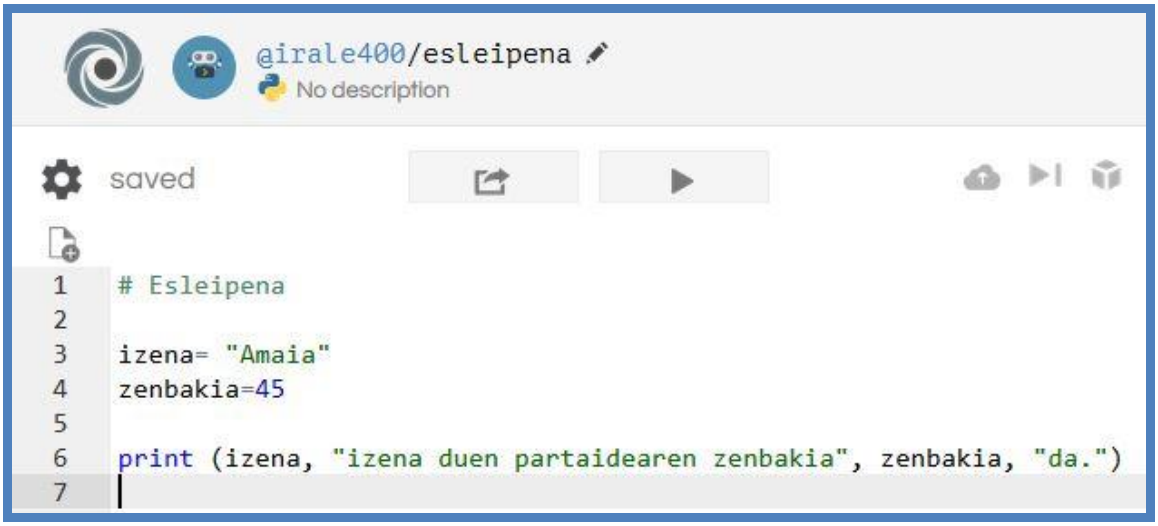
3.5.1.1.irudia



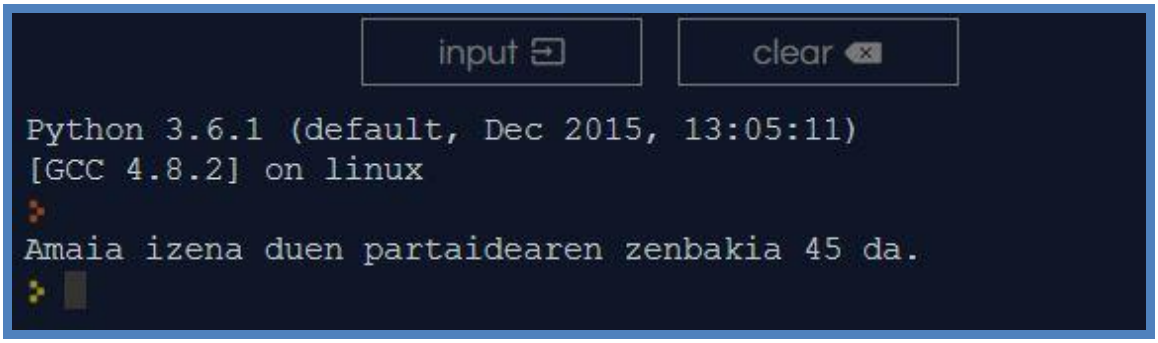
3.5.1.2.irudia

Adibidean ikus daitekeen bezala, `input` funtzioa bi modutara erabil daiteke. Bigarren modura erabiliz gero, kodeak lerro gutxiago izango ditu; nahiz eta programa txikietan gehiegi eragin ez, asko eskertzen da programa luzeak idaztean.

2. Aldagaiei balioak esleitzeari *esleipen* esaten zaio eta, balioa esleitzeko, = ikurra erabiltzen da. Ikus honako adibide hauetan nola egiten den:



3.5.2.1. irudia



3.5.2.2. irudia

4. ALDAGAIAK

4.1. ALDAGAIAK, KONSTANTEAK ETA IRUZKINAK

Informatikan balio asko erabiltzen dira programa bat kodetzeko, eta **aldagaiek** balio horiek ordenagailuaren memorian biltegiratzeko balio dute. Aldagaiek memoriako espazio zehatz bati egiten diote erreferentzia eta, programak aurrera egin ahala, hainbat balio har ditzakete. Aldagai bat definitzeko identifikatzailea definitu behar da lehenengo; hau da, gordeko duen balioarekin zerikusia duen izena zein izango den erabaki behar da eta baita balio hori zein datu motakoa izango den ere. Gainera, izenak esanguratsua izan behar du, lehenengo irakurraldian balioaren esanahia ulertzeko modukoa. Esaterako, hiru pertsonen artean nagusiena zein den kalkulatzeko programa kodetu behar bada, adinaren balioa gordeko duen aldagaiaren izena *adina* izatea izango litzateke egokiena; alegia, gordeko duen balioa deskribatu behar du aldagaiaren izenak. Hona hemen aldagaiaren izena zehazteko kontuan hartu beharreko zenbait irizpide:

- Aldagaia izendatzeko zenbakiak eta hizkiak erabil daitezke, baina izenak beti hizkiz hasi behar du.
- Letra larriak eta xeheak erabil daitezke, baina letra xeheak erabiltzea hobesten da.
- Pythonek letra larriak eta xeheak bereizten ditu, hau da, ez da berdina '**adina**' edo '**Adina**' identifikatzailea.
- Azpi marratxoa (**_**)erabil daiteke.
- Aldagai bat izendatzeko orduan, ezin dira erabili Pythonek beretzat erreserbatutako hitzak. Honako hauek besteak beste: `and`, `assert`, `break`, `class`, `contine`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `in` , `is`, `lambda`, `not`, `or`, `pass`, `print`, `raise`, `return`, `try` eta `while`.

> **Gomendioa**. Aldagaien izenek deskribatzaileak izan behar dute, hau da, ez dira anbiguoak izan behar, eta argi azaldu behar dute zer balio biltegiratzen duten. Adibidez, fruta-kopurua biltegiratu behar bada, **fruta_kop=15** aldagaiaren izena hobetsiko da **f=15** moduko izenen aurretik.

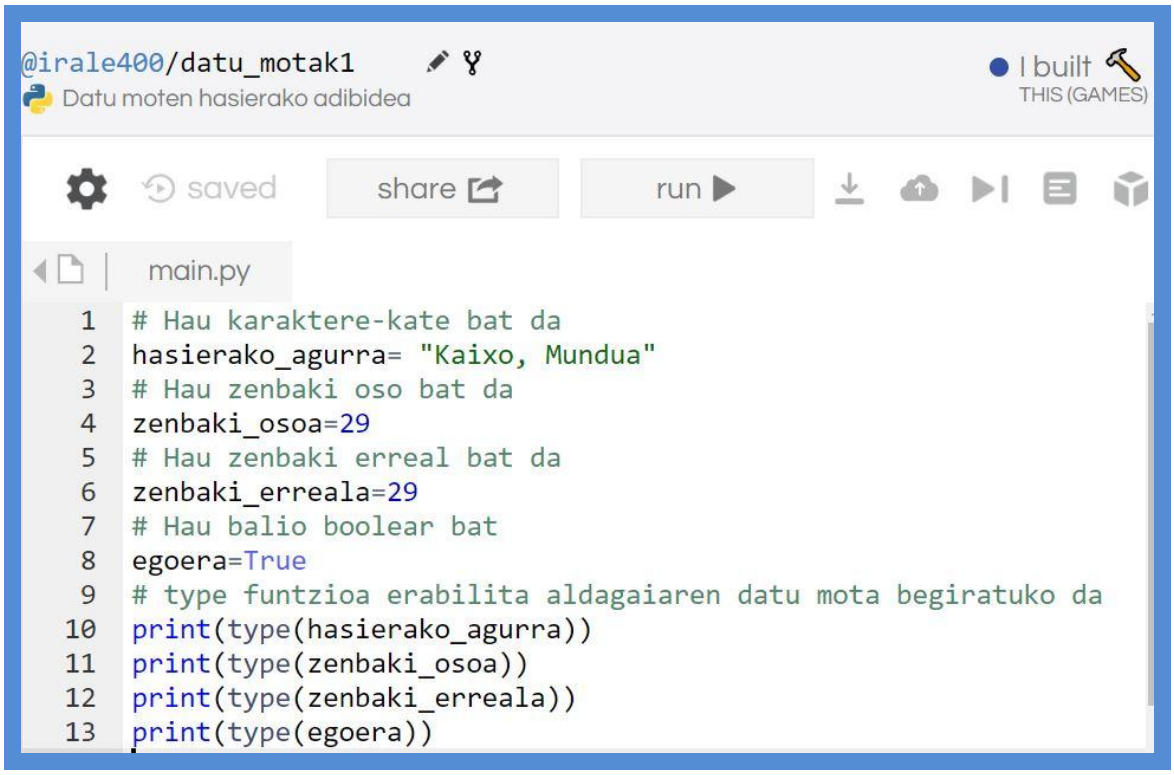
Programazio-lengoaia gehienetan aldagaiak aldez aurretik deklaratzen/erazagutzen dira; izan ere, aldagaiaren datu mota zein den zehazten da aldagaiak hartuko dituen balioak kontuan hartuz, eta aurrerantzean ez da aldatzen aldagaiaren datu mota. Aitzitik, Pythonen, aldagai berberak hainbat datu mota har ditzake momentuko beharren arabera, programazio-lengoaia dinamikoa baita. Hori dela eta, Pythonen ez dira aldagaiak erazagutu behar.

Aldagaiez gain, programazio-lengoaia gehienek **konstanteak** erabiltzeko aukera ere eskaintzen dute. Konstanteek memoriako espazio zehatz bati egiten diote erreferentzia, baina kasu horietan, behin balioa esleituz gero, ez dago berriro aldatzeko aukerarik; hau da, konstantea den aldagaiak programa osoan zehar balio berbera izango du. Adibidez, euroa eta Pi balioa bezalako balioak gordetzeko erabiltzen dira konstanteak.

Aldagaiak zer diren zehaztu ondoren, Pythonen erabiltzen diren datu motak zeintzuk diren zehaztu behar da. Pythonen oinarrizko datu motak honako hiru multzo hauetan banatzen dira:

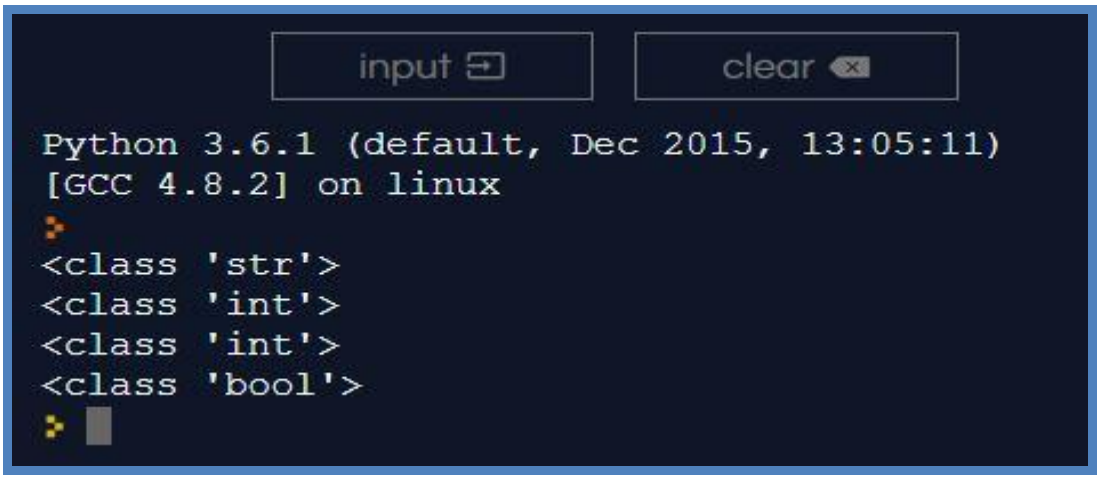
- **Zenbakiak.** Zenbaki osoak eta zenbaki errealak; adibidez, 7 eta 7.7 zenbakiak.
- **Karaktere-kateak**. ˈKaixo, Munduaˈ bezalako karaktere-kateak.
- **Balio boolearrak**. *True* (egia) eta *False* (gezurra) balioak.

Honako adibide honetan, aipatutako datu moten multzo bakoitzeko aldagai bat nola sortu ikus daiteke:



4.1.1. irudia

Hona hemen bere kodearen exekuzioaren emaitza:



4.1.2. irudia

Aurreko adibideak programan egingo diren **iruzkinak** nola jarri ikusteko ere balio du, izan ere, Pythonek # ikurrarekin hasten diren karaktere-kateak ez ditu exekutatzen. Iruzkinak oso erabilgarriak dira programaren kodearen inguruko oharrak zehazteko, batez ere, programak lerro asko dituenean.

4.2. ZENBAKIAK: OSOAK, ERREALAK ETA ERAGILE ARITMETIKOAK

Programazioan, osoak, errealak edo konplexuak izan daitezke zenbakiak; hala ere, unitate honetan zenbaki osoak eta errealak bakarrik aztertuko dira.

Zenbaki **osoak** hamartarrik ez duten zenbaki positibo edo negatiboak dira, eta Pythonen `int` edo `long` datu motak erabilita adieraz daitezke. `Long` datu motak `int` datu motak baino zenbaki handiagoak biltegiratzeko aukera ematen du. Dena den, soilik beharrezkoa denean erabiltzea komeni da, memoria ez xahutzeko.

Bestalde, zenbaki **errealak** adierazteko, Python programazio-lengoaian, `float` datu mota erabiltzen da.

Zenbaki mota horiekin hainbat eragiketa egin daitezke eta, horretarako, **eragile aritmetikoak** behar dira. Hona hemen eragile aritmetikoak, haien deskribapenak eta kasu bakoitzeko adibide bana:

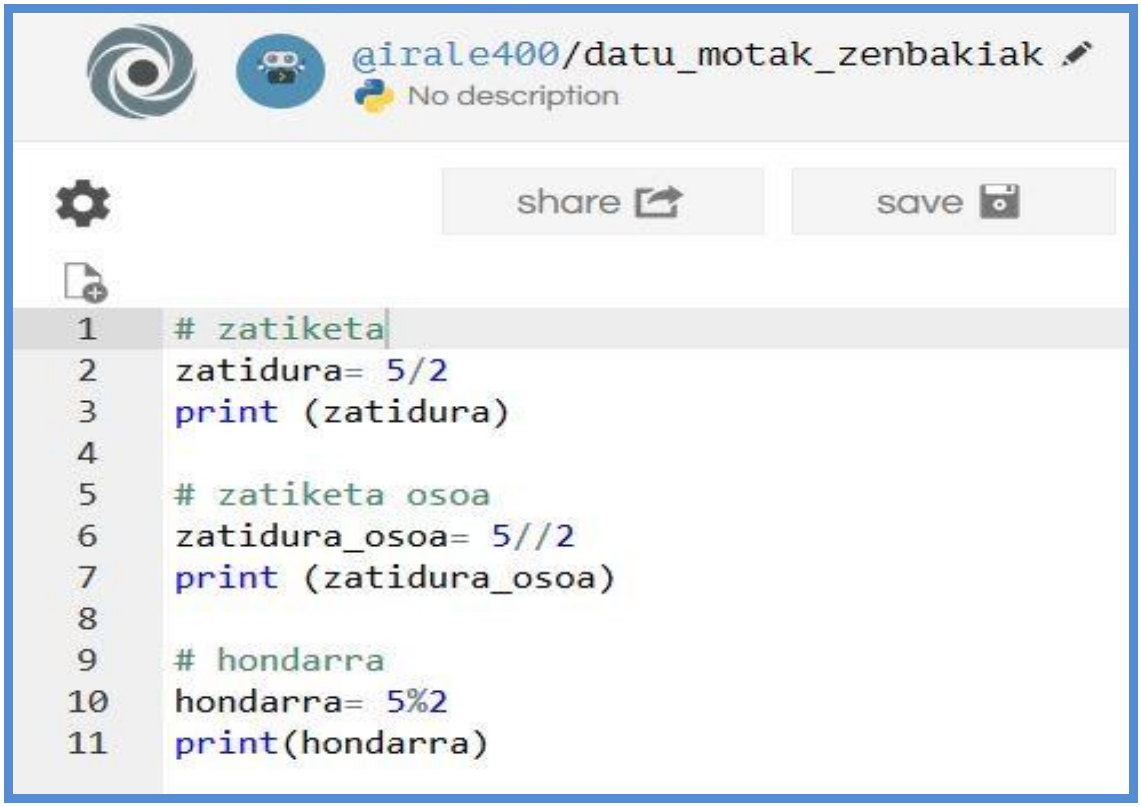
Eragile aritmetikoak	Deskribapena	Adibidea
+	Batuketa	e = 4 +2 # e aldagaiaren balioa 6 da.
-	Kenketa	e = 4 -2 # e aldagaiaren balioa 2 da.
-	Ukazioa	e = -2 # e aldagaiaren balioa -2 da.
*	Biderketa	e = 4 *2 # e aldagaiaren balioa 8 da.
**	Berreketa	e = 4 **2 # e aldagaiaren balioa 16 da.
/	Zatiketa	e = 4.5 / 2 # e aldagaiaren balioa 2.25 da.
//	Zatiketa osoa	e = 4.5 // 2 # e aldagaiaren balioa 2 da.
%	Hondarra	e = 5 % 2 # e aldagaiaren balioa 1 da.

Hondarra kalkulatzeko erabiltzen den eragileak nola funtzionatzen duen ulertzeko orduan, zalantzak sor daitezke; izan ere, hondarra kalkulatzeko erabiltzen den eragileak eragingai biren arteko zatiketaren hondarra itzultzen du, eta ez zatidura. Adibidez:

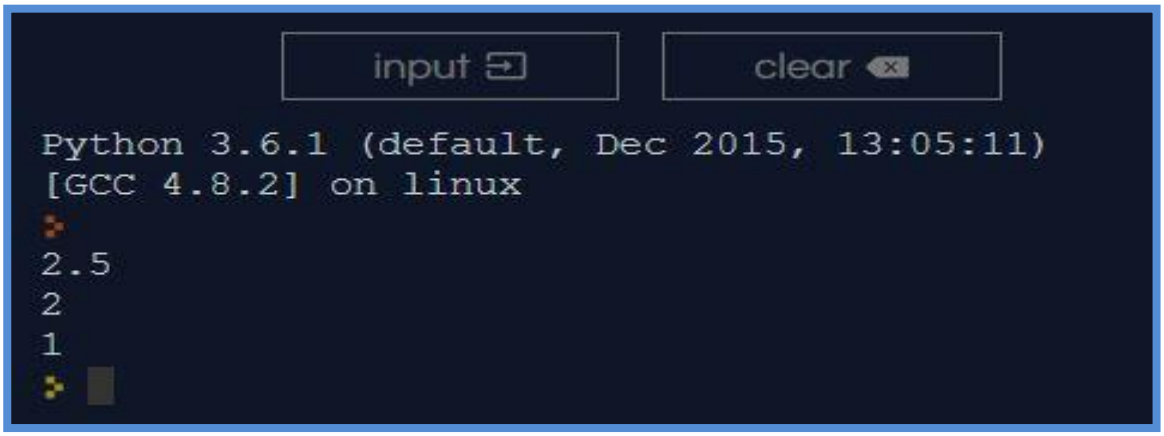
> 5 / 2 eragiketaren zatidura 2.5 da, baina 5 % 2 eragiketak hondarra itzultzen duenez, emaitza 1 izango da.

Horrez gain, garrantzitsua da zatiketaren eta zatiketa osoaren arteko desberdintasuna zein den argi izatea. Zatiketak zenbaki erreal bat itzultzen du zati osoarekin eta dagokion hamartarrarekin; aldiz, zatiketa egiteko // eragile aritmetikoa erabiltzen bada, zatiketaren emaitza zenbaki osoa izango da, hamartarrik gabe. Adibidez, 5//2 eragiketaren emaitza 2 izango da.

Hori guztiori kontuan hartuta, hona hemen Pythonen egindako programa bat eta bere exekuzioaren emaitza:



4.2.1. irudia

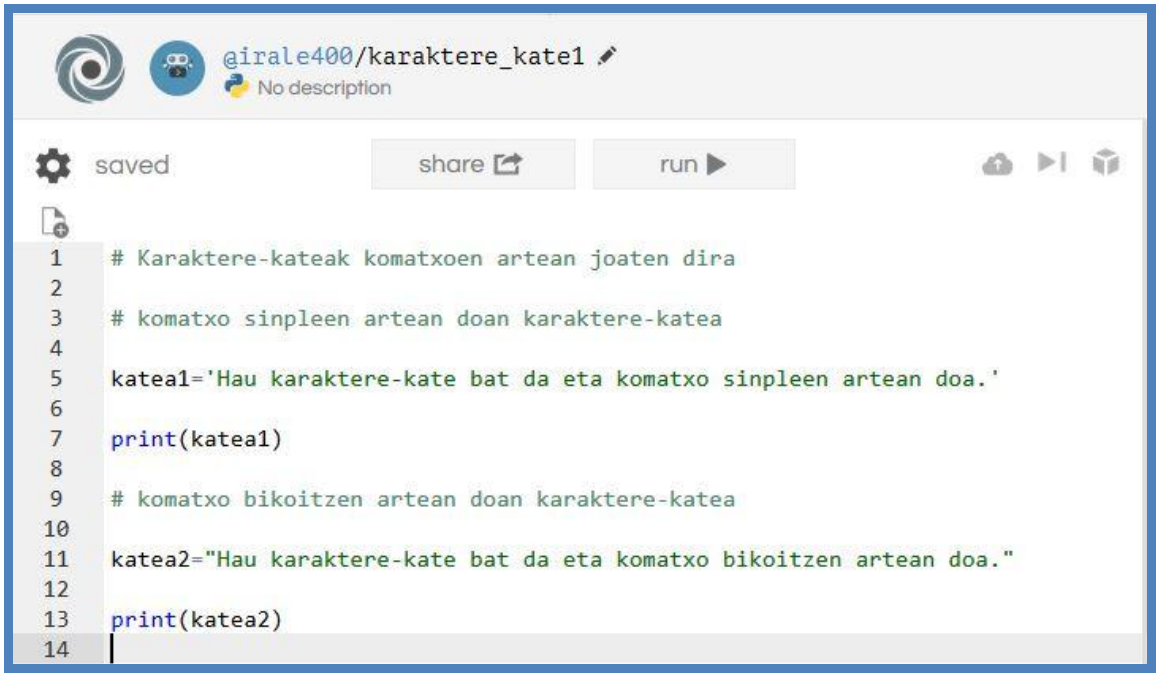


4.2.2 irudia

4.3. KARAKTERE-KATEAK ETA FUNTZIOAK

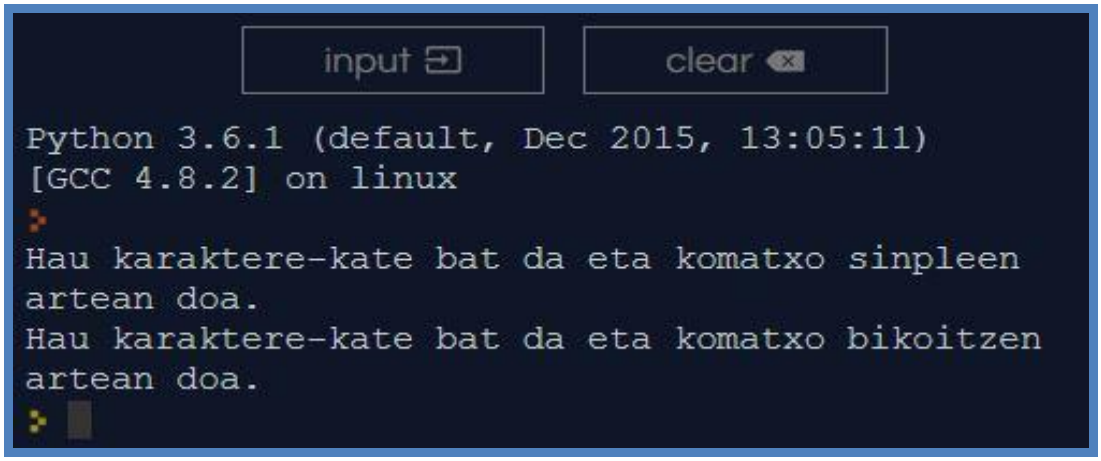
4.3.1. KARAKTERE-KATEAK

Karaktere-kateak komatxo sinple edo komatxo bikoitzen arteko testuak dira. Hona hemen komatxoen erabileraren nondik norakoak ulertzeko programazio-eredu bat :



4.3.1.1. irudia

Hau da bere kodearen exekuzioaren emaitza :

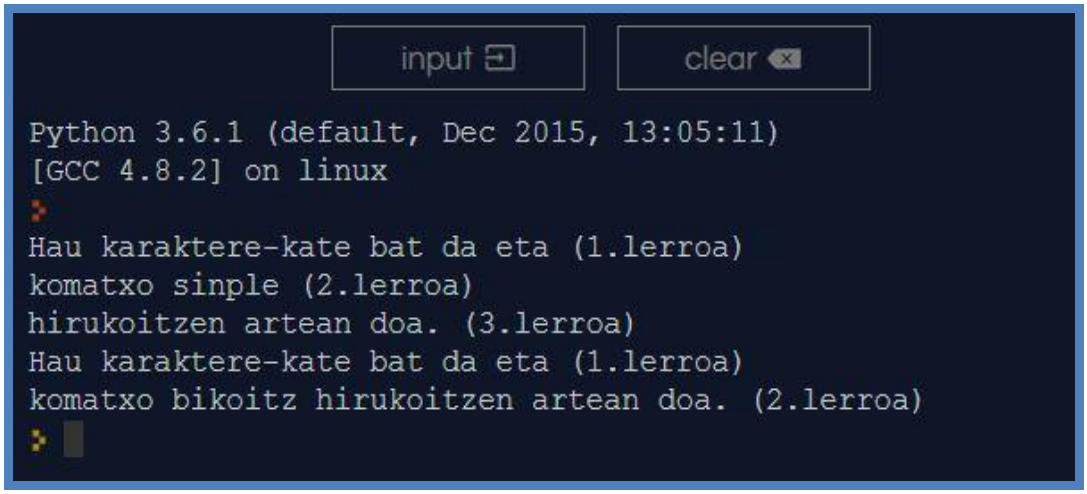


4.3.1.2. irudia

Hala ere, komatxo sinpleen edo komatxo bikoitzen arteko karaktere-kateek ez dituzte errespetatzen lerro-jauziak. Horretarako, ezinbestekoa da karaktere-katea komatxo sinple hirukoitzen –'''*katea*'''– edo komatxo bikoitz hirukoitzen –"""*katea*"""– artean idaztea. Horrela, testua lerro anitzetan idatzi daiteke, eta, beste programazio-lengoaia batzuetan ez bezala, katea inprimatzean errespetatu egiten dira testuan sartutako lerro-jauziak, **/n** karaktere berezia idatzi beharrik gabe. Ikus ondoko adibidea:

4.3.1.3. irudia

Hau da aurreko adibideko kodearen exekuzioaren emaitza:



4.3.1.4. irudia

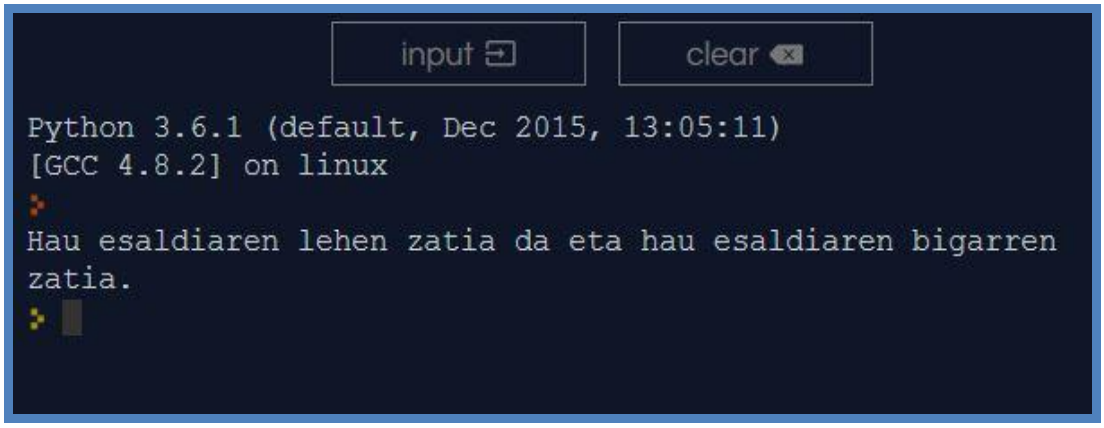
Komatxoez gain, badira Phytonek onartzen dituen beste bi ikur ere: + ikurra eta * ikurra. Hona hemen ikurraren funtzioa zein den eta horien adibide bana:

+ ikurra

Karaktere-kateak kateatzeko baliatzen du Phytonek + ikurra.

4.3.1.5. irudia

Hau da bere kodetzearen exekuzioaren emaitza:



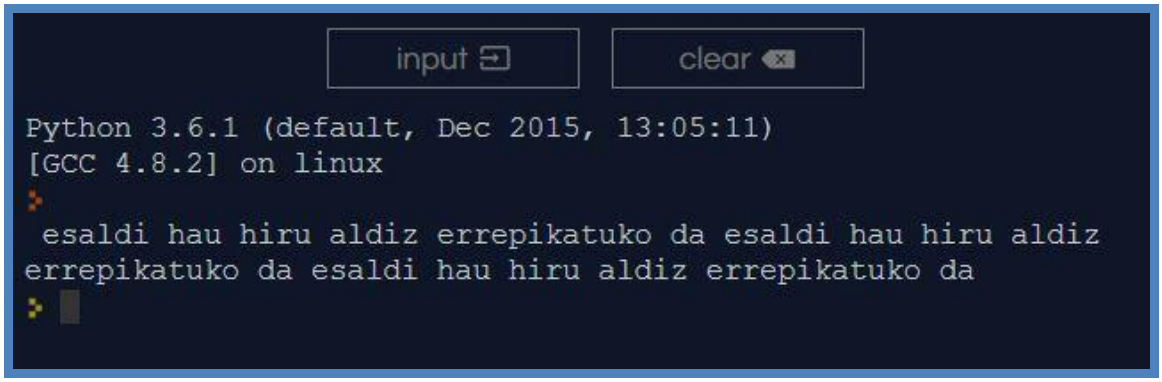
4.3.1.6. irudia

*** ikurra**

Eragile honekin programatzaileak zehazten duen beste aldiz errepikatzen da karaktere-katea.

4.3.1.7. irudia

Hona hemen bere kodearen exekuzioaren emaitza:



4.3.1.8. irudia

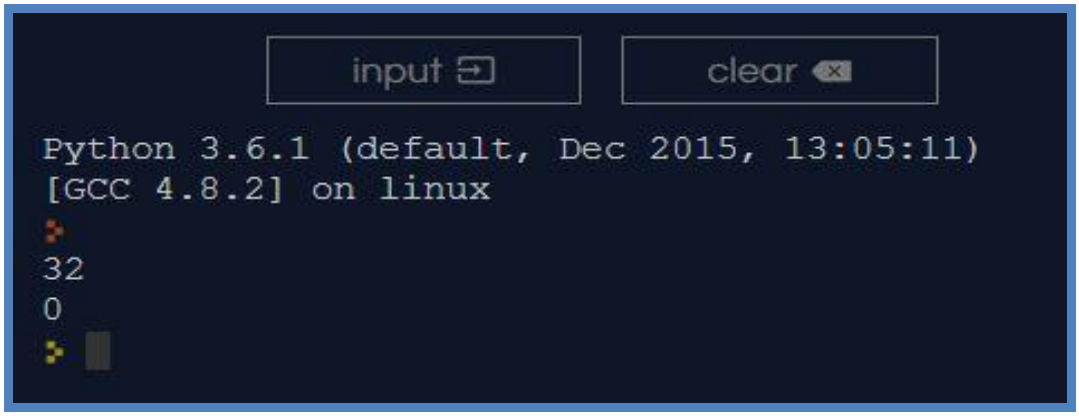
4.3.2. FUNTZIOAK

Karaktere-kateekin lan egiteko, Pythonek askotariko funtzioak eskaintzen ditu. Hona hemen erabilienak eta funtzio bakoitzari dagokion adibidea:

- **`find`**(). Bilatzen ari den karaktere-katearen lehen hizkiaren indizea itzultzen du funtzio honek. Indizeak 0tik hasita zenbatzen ditu Pythonek. Bilatzen ari den karaktere-katea aurkitzen ez badu, -1 balioa itzuliko du, honela:

4.3.2.1. irudia

Hau da `find` funtzioarekin egindako kodearen exekuzioaren emaitza:

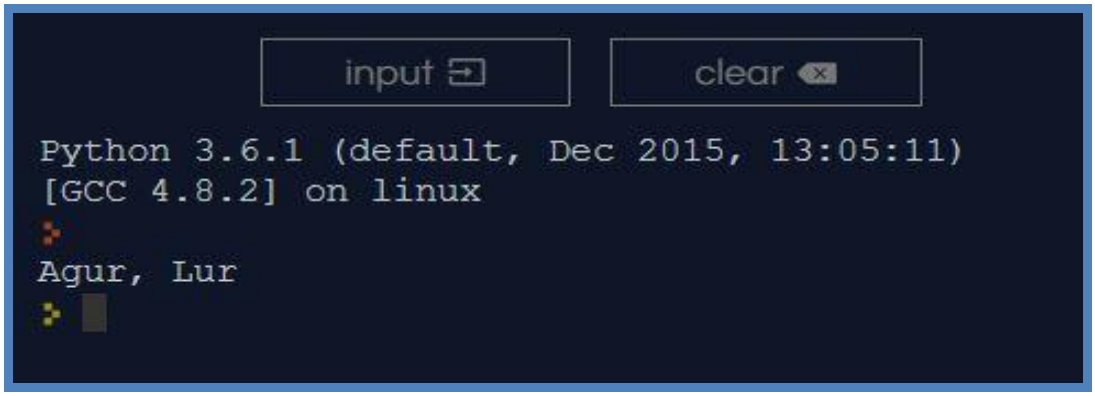


4.3.2.2. irudia

- **`replace`**(). Karaktere batzuk beste batzuengatik ordezkatzeko balio du funtzio honek, eta honela egin daiteke:

4.3.2.3. irudia

Hemen `replace` funtzioarekin egindako kodetzearen exekuzioaren emaitza:

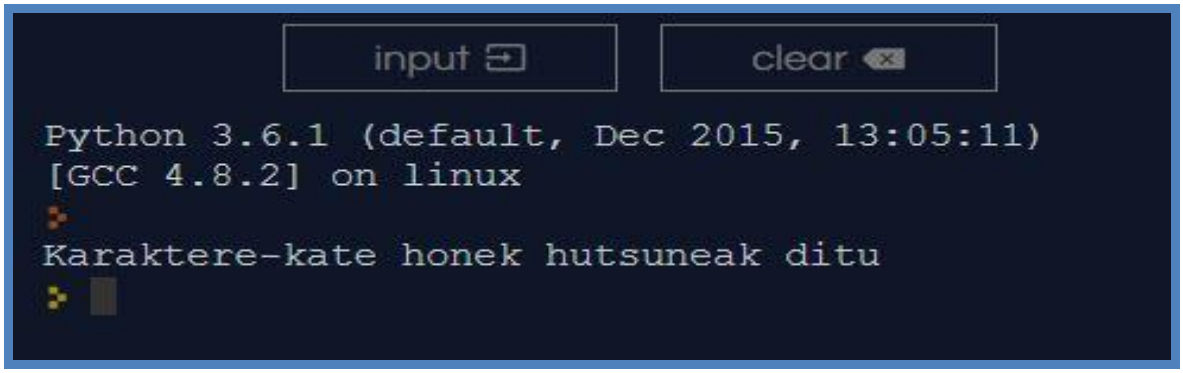


4.3.2.4. irudia

- **`strip`**(). Karaktere-katean dauden ezkerreko eta eskuineko hutsune guztiak ezabatzen ditu funtzio honek.

4.3.2.5. irudia

Hau, berriz, `strip` funtzioarekin egindako kodearen exekuzioaren emaitza:

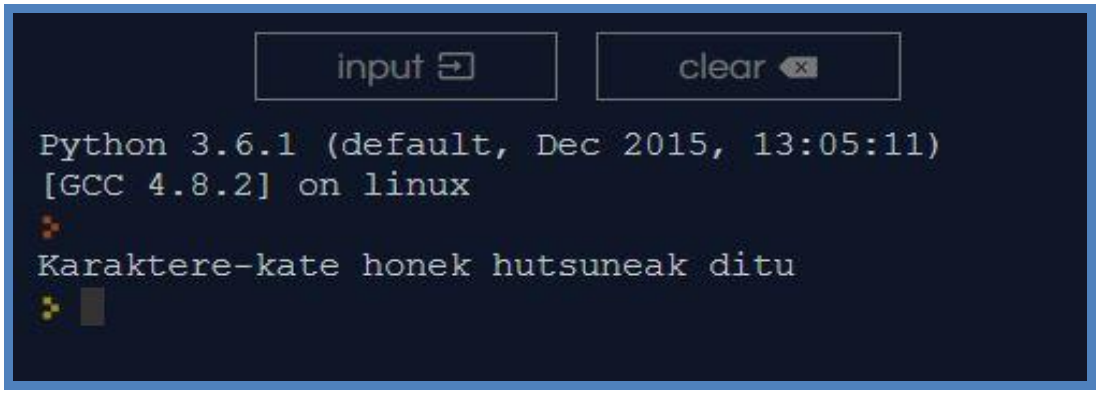


4.3.2.6. irudia

- **`lstrip`**(). Karaktere-katean dauden hutsune guztiak ezabatzen ditu funtzio honek, ezkerretik eskuinera.

4.3.2.7. irudia

Hemen bere kodearen exekuzioaren emaitza:

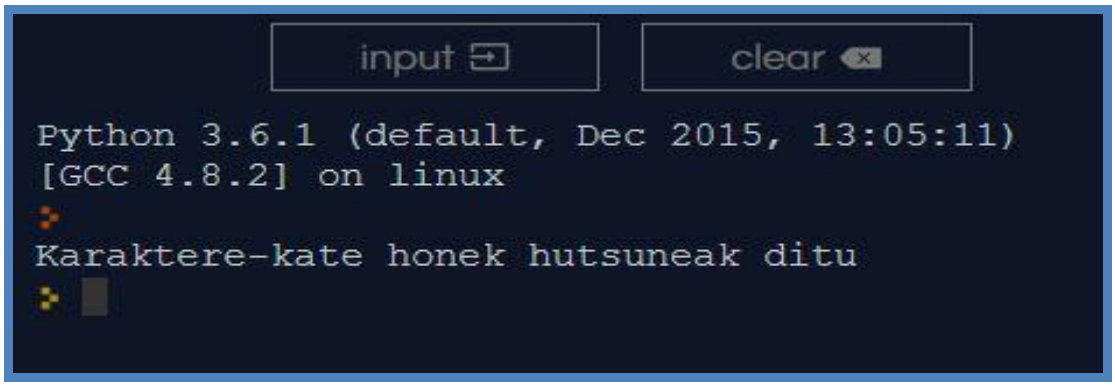


4.3.2.8. irudia

- **`rstrip`**(). Funtzio honek karaktere-katean dauden hutsune guztiak ezabatzen ditu, eskuinetik ezkerrera egin ere.

4.3.2.9. irudia

Hau da dagokion kodearen exekuzioaren emaitza:

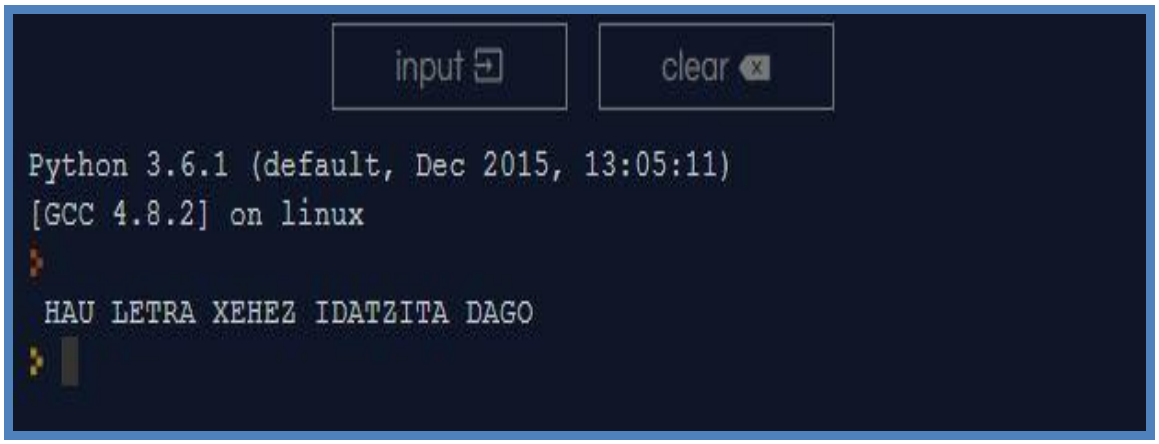


4.3.2.10. irudia

- **upper**(). Karaktere-kateko karaktereak letra larri bihurtzen ditu `upper()` funtzioak.

4.3.2.11. irudia

Hau da goiko irudiko programari dagokion kodearen exekuzioaren emaitza:

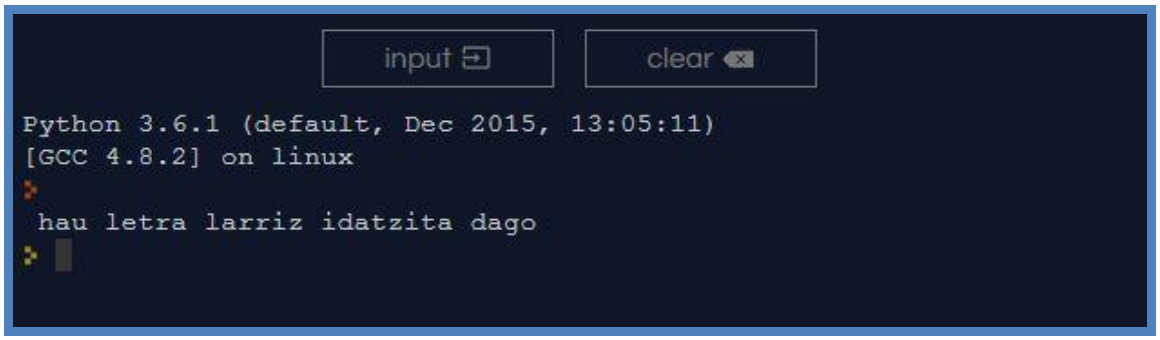


4.3.2.12. irudia

- **`lower()`.** Karaktere-kateko karaktere guztiak letra xehe bihurtzen ditu funtzio honek.

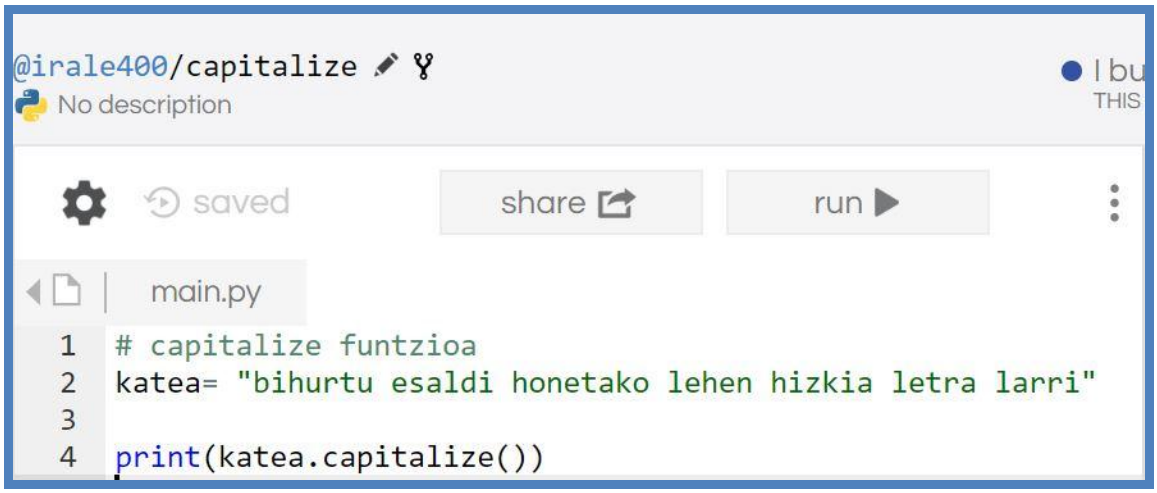
4.3.2.13. irudia

Hau da `lower` funtzioarekin egindako kodearen exekuzioaren emaitza:



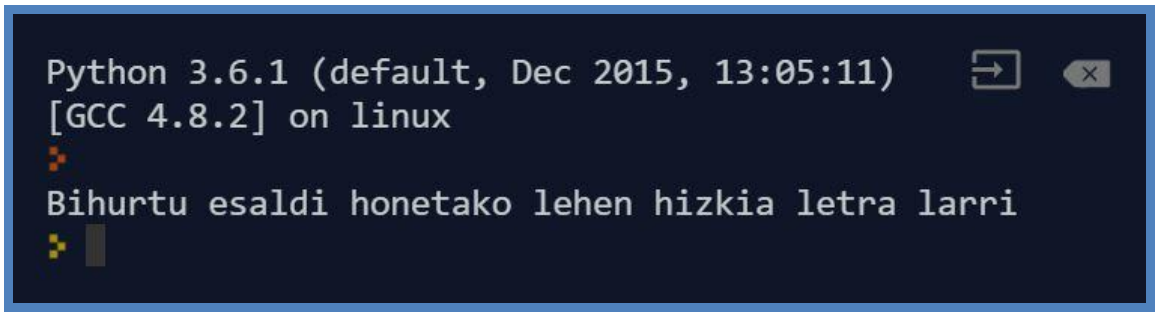
4.3.2.14. irudia

- **`capitalize`**(). Funtzio honek karaktere-kateko lehen hizkia letra larri bihurtzen du.



4.3.2.15. irudia

Hau da `capitalize` funtzioarekin egindako kodearen exekuzioaren emaitza:

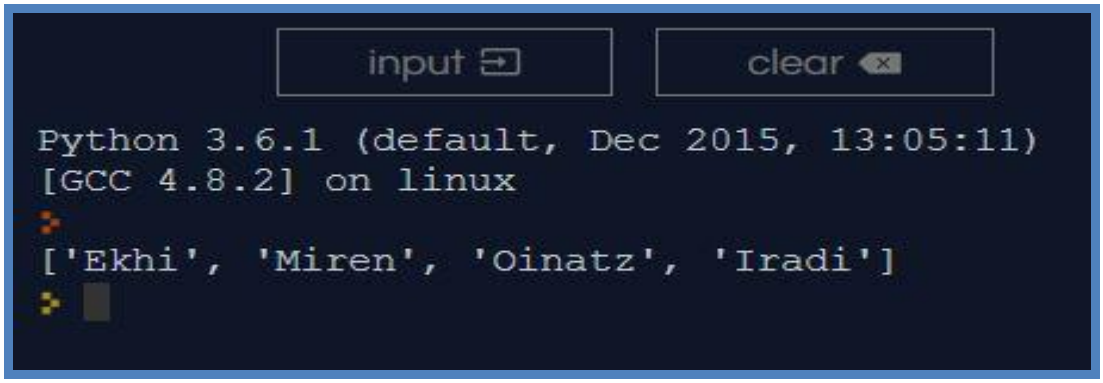


4.3.2.16. irudia

- **`split`**(). Karaktere-kate bat zatitzeko erabiltzen da `split()` funtzioa.

4.3.2.17. irudia

Hau da dagokion kodearen exekuzioaren emaitza:

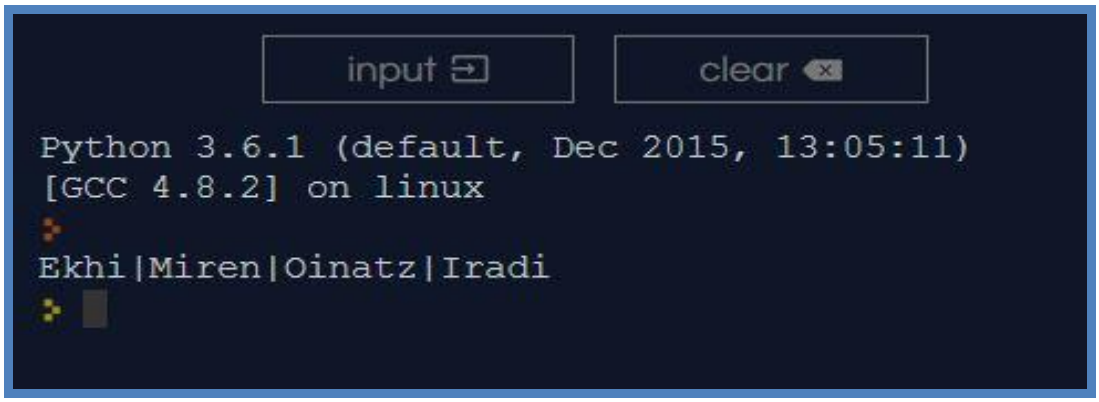


4.3.2.18. irudia

- **`join()`.** Karaktere-kateak elkartzeko erabiltzen da funtzio hau.

4.3.2.19. irudia

`Join` kodearen exekuzioaren emaitza da honako hau:



4.3.2.20. irudia

4.4. BOOLEARRAK

Aldagai boolear batek bi balio soilik har ditzake: *true* (egia) edo *false* (gezurra). Balio horiek oso garrantzitsuak dira bai baldintzapeko egituretarako, bai begizten egituretarako.

Balio boolearren arteko eragiketak egiteko eragile logikoak behar dira. Hona hemen eragile logiko erabilienak:

ERAGILE LOGIKOAK	DESKRIBAPENA	ADIBIDEA
and	a eta b betetzen da?	e = *True and False* # e *False* da.
or	a edo b betetzen da?	e = *True or False* # e *True* da.

not	a-ren ezeztapena	e = *not True* # e *False* da.
		e = *not False* # e *True* da.

Bestalde, erlazioak bi balio boolear konparatzeko erabiltzen dira, eta eragiketa horien emaitzak balio boolearra izaten dute. Hona hemen balio boolearrak konparatzeko erabiltzen diren erlazioak:

ERLAZIOAK	DESKRIBAPENA	ADIBIDEA
==	a eta b berdinak al dira?	e = 4 == 2 # e *False* da.
! =	a eta b desberdinak al dira?	e = 4 != 2 # e *True* da.
<	a b baino txikiagoa al da?	e = 4 < 2 # e *False* da.
>	a b baino handiagoa al da?	e = 4 > 2 # e *True* da.

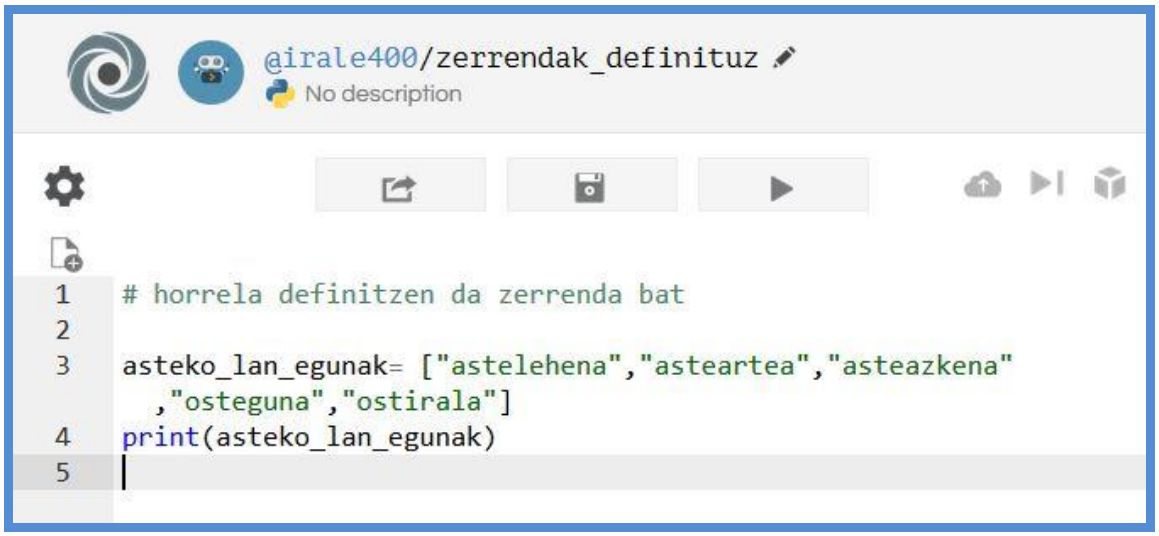
<=	a b baino txikiagoa ala berdina da?	e = 4 <= 2 # e *False* da.
		e = 2 <= 2 # e *True* da.

>=	a b baino handiagoa ala berdina da?	e = 4 >= 2 # e *True* da.
		e = 2 >= 2 # e *True* da.

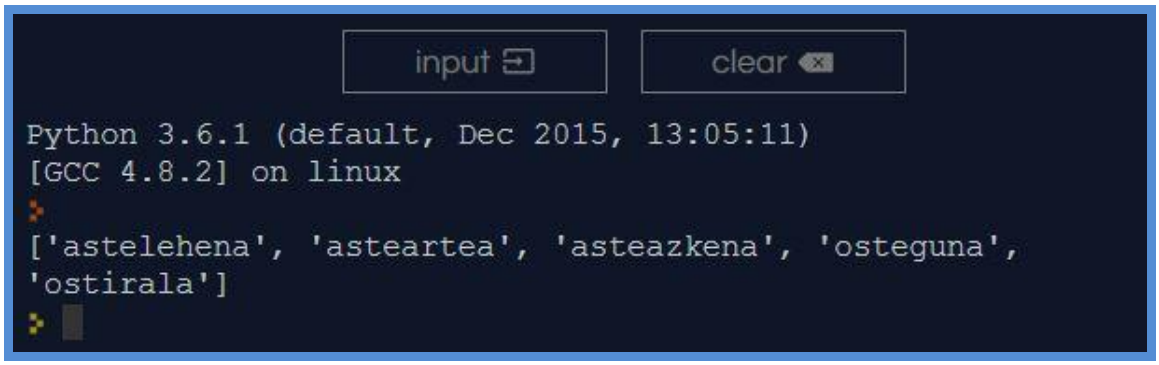
4.5. ZERRENDAK

Zerrendak datu-bilduma ordenatuak dira, eta programazioaren esparruan `array` edo *bektore* ere deitzen zaie. Zerrenda batek askotariko datu motak izan ditzake, hau da, zenbaki, karaktere-kate eta boolear motako balioak izan ditzake, baita zerrenda motako balioak ere.

Zerrenda bat sortzea zerrendan gorde nahi diren balioak kortxeteen artean eta koma bidez banatuta jartzea bezain sinplea da. Hona hemen zerrenda bat nola definitzen den ikusteko adibidea (4.5.1. irudia) eta exekuzioaren emaitza (4.5.2. irudia):

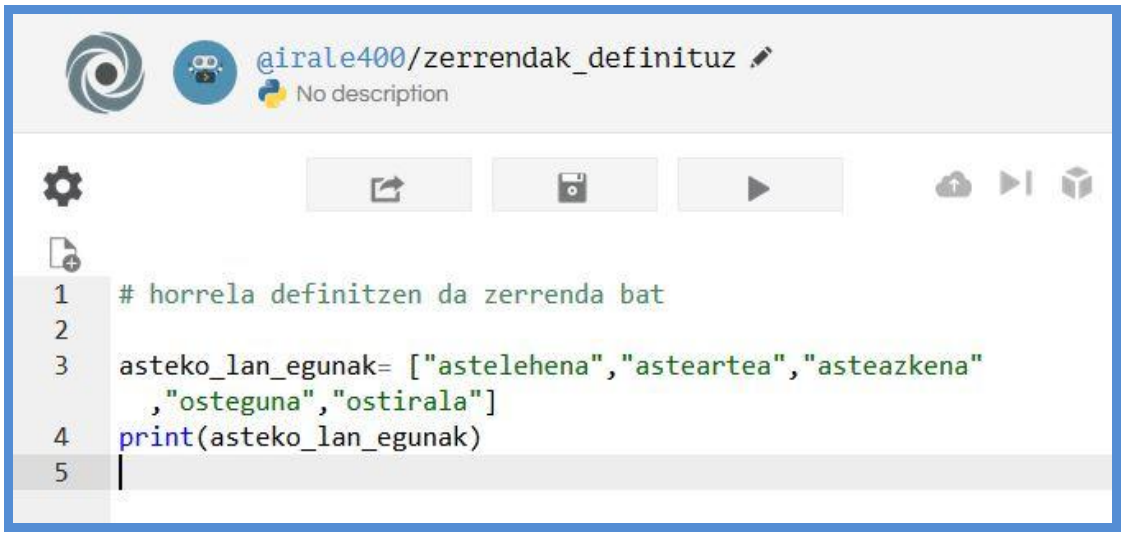


4.5.1. irudia

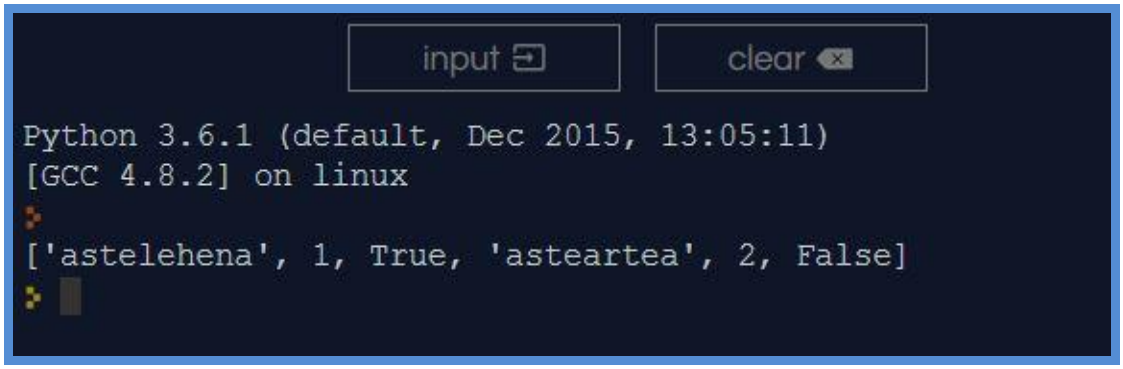


4.5.2. irudia

Zerrenda batean askotariko datu moten balioak ere gorde daitezke. Hona hemen horren adibide bat eta exekuzioaren emaitza:



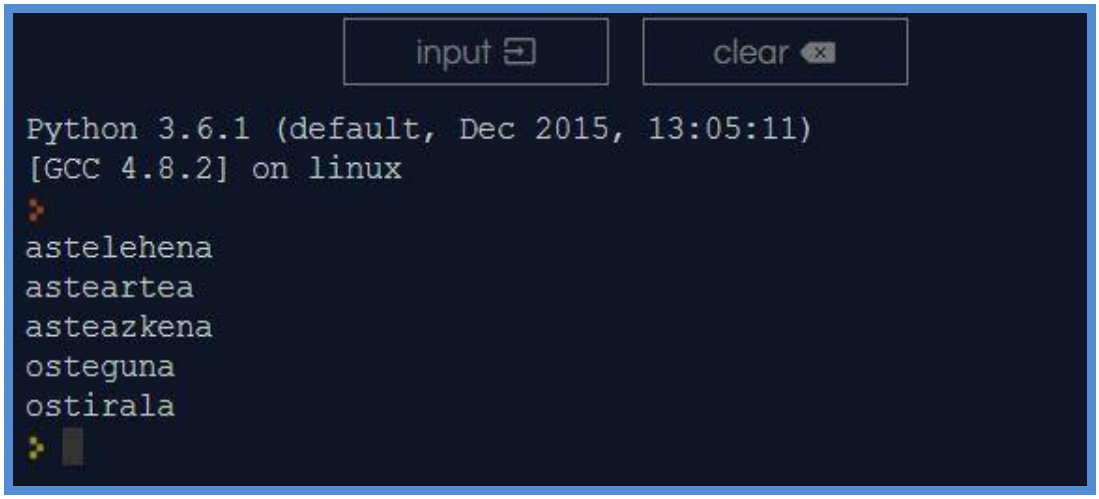
4.5.3. irudia



4.5.4. irudia

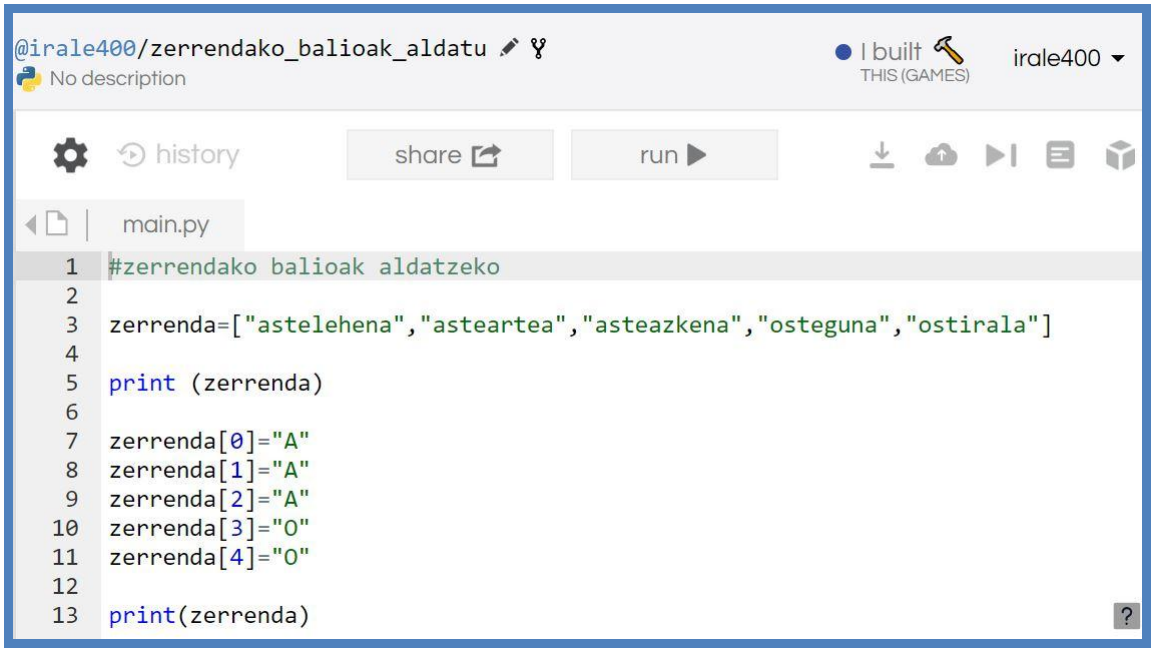
Zerrendako elementu bakoitza atzitzeko, zerrendaren izena eta elementuaren indizea kortxete artean zehaztu behar dira. Kontuan hartu behar da lehenengo elementuaren indizea 0 dela beti, eta ez 1. Ikus honako adibide hau eta programari dagokion exekuzioaren emaitza:

4.5.5. irudia



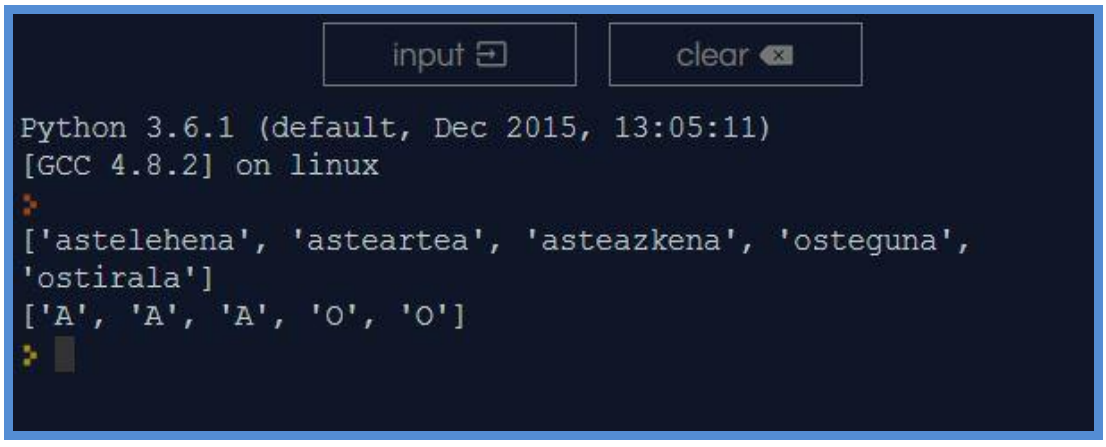
4.5.6. irudia

Zerrendako elementu baten balioa aldatzeko, zerrendaren izena eta kortxete artean aldatu nahi den balioaren indizea zehaztu behar dira. Honela:



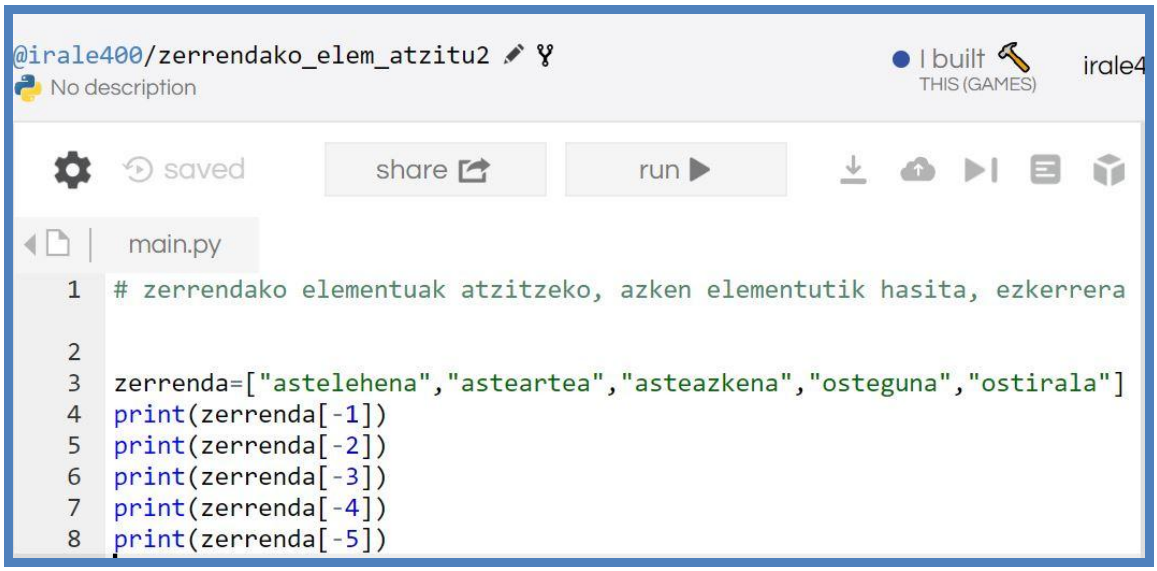
4.5.7. irudia

Hona hemen 4.5.7. irudiko programaren exekuzioaren emaitza:

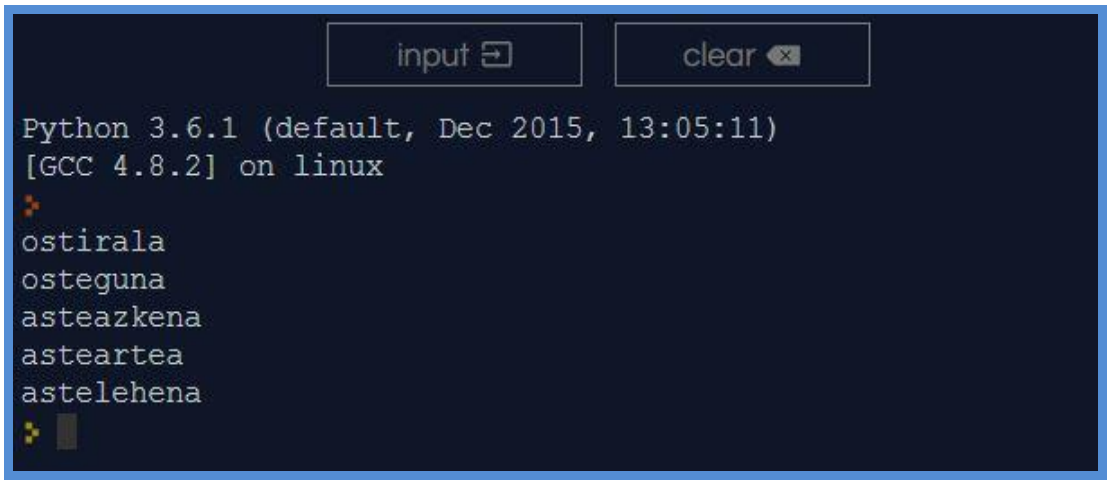


4.5.8. irudia

Aipatutako funtzioaz gainera, kortxeteek Pythonen indizeetarako zenbaki negatiboak erabiltzeko ere balio dute. Hau da, indize modura zenbaki negatibo bat erabili nahi denean, zerrendako azken elementutik hasiko da kontatzen Python, eskuinetik ezkerrera. Hona hemen indize negatiboak erabiltzen dituen programa baten adibidea eta bere exekuzioaren emaitza:

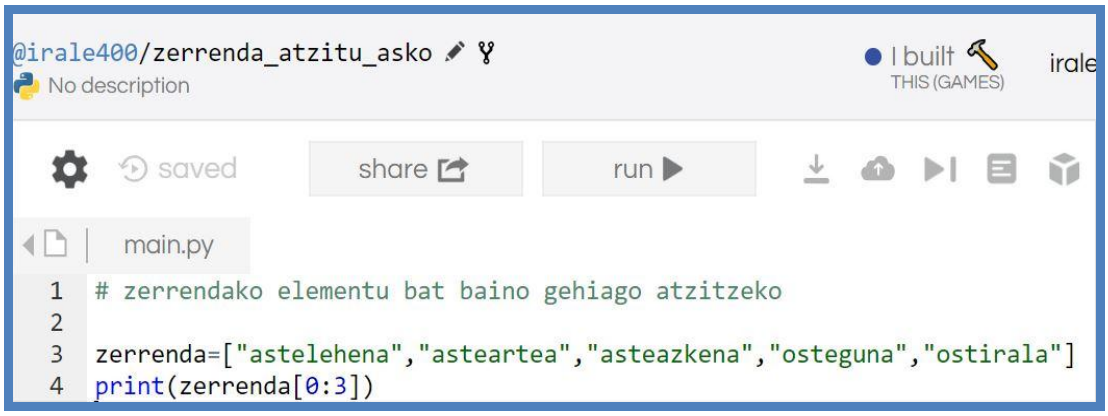


4.5.9. irudia

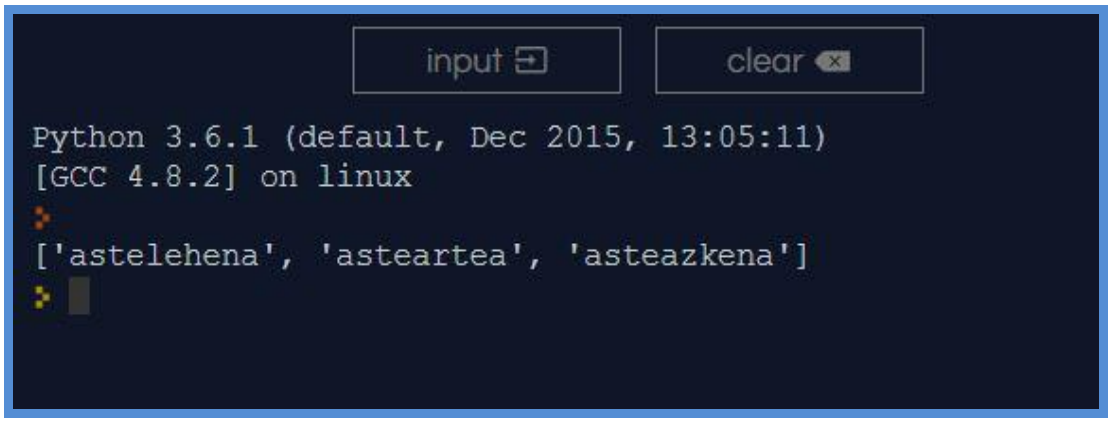


4.5.10. irudia

Pythonek zerrenda bateko elementu bat baino gehiago aldi berean atzitzea nahi bada, *slicing*-ak edo partizioak egin daitezke. Hau da, zenbaki baten ordez, hasiera eta bukaerako zenbakiak zehazten badira (hasiera:bukaera), definitutako lehen posiziotik bigarren posiziora arteko balioak dituen zerrenda nahi dela ulertuko du Pythonek, bigarren posizioko balioa hartu gabe. Ikus 4.5.11. irudiko adibidea, eta 4.5.12. irudian, berriz, dagokion exekuzioaren emaitza:

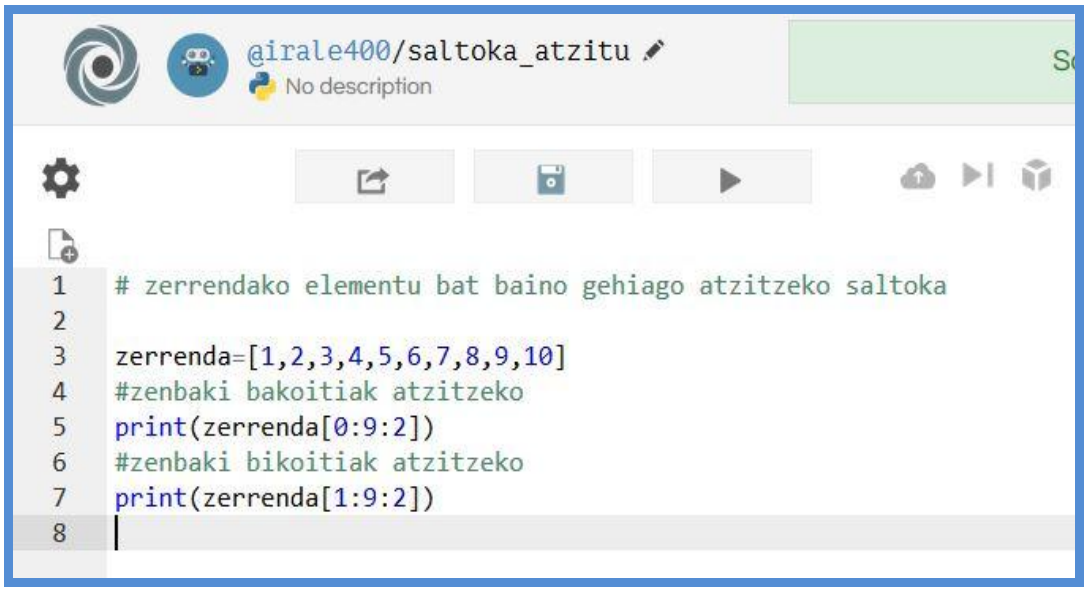


4.5.11. irudia



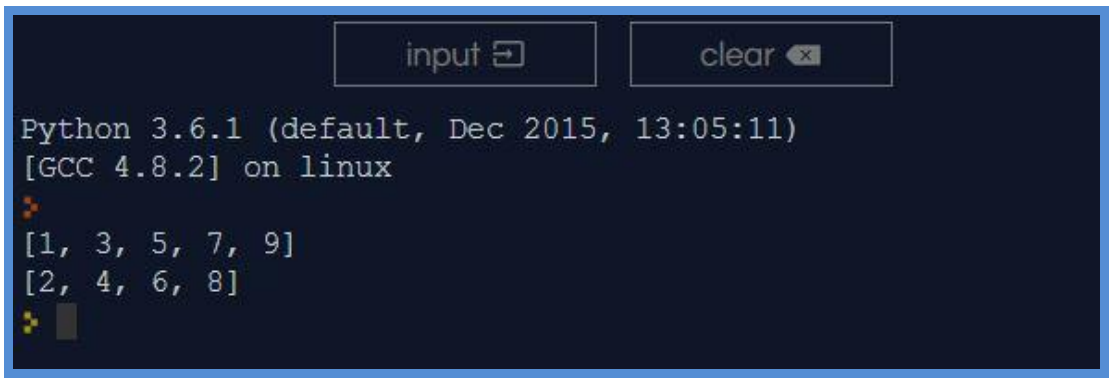
4.5.12. irudia

Aldiz, bi zenbakiren ordez hiru zenbaki idatziz gero (hasiera: bukaera: jauzia), atzitzen den elementu batetik atzituko den hurrengo elementura zenbat posizio egongo diren zehazten du azkeneko zenbakiak. Honela:



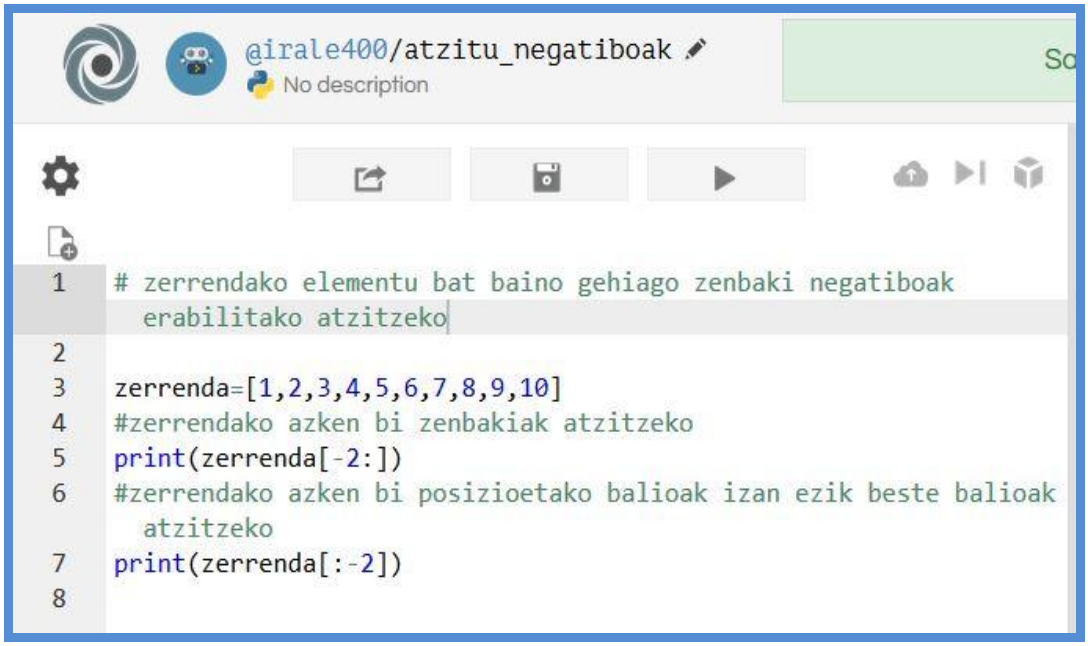
4.5.13. irudia

Ikus hemen 4.5.13. irudiko kodearen exekuzioaren emaitza:



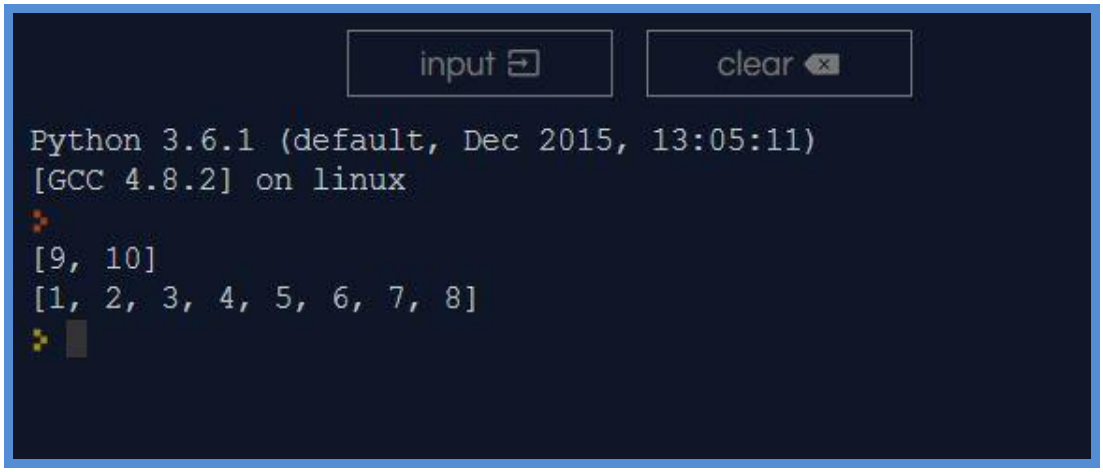
4.5.14. irudia

Partizioa edo *slicing*-a egiteko, zenbaki negatiboak ere erabili daitezke, honela:



4.5.15. irudia

Hemen 4.5.15. irudiko kodearen exekuzioaren emaitza:

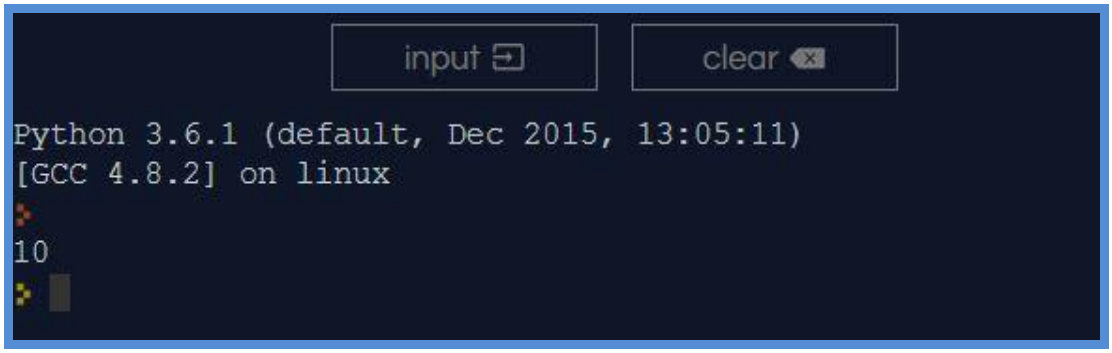


4.5.16. irudia

Programatzean, interesgarria izaten da zerrenda baten elementu-kopurua zein den jakitea; horretarako, Pythonek `len` funtzioa erabiltzen du, adibidean egiten den moduan:

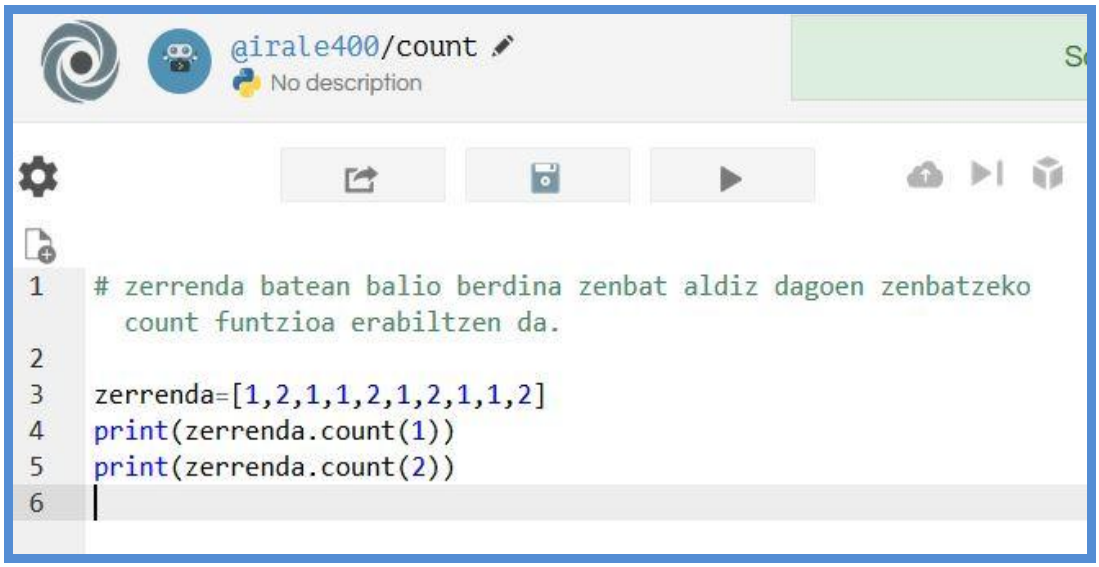
4.5.17. irudia

Ikus programaren kodeari dagokion exekuzioaren emaitza:

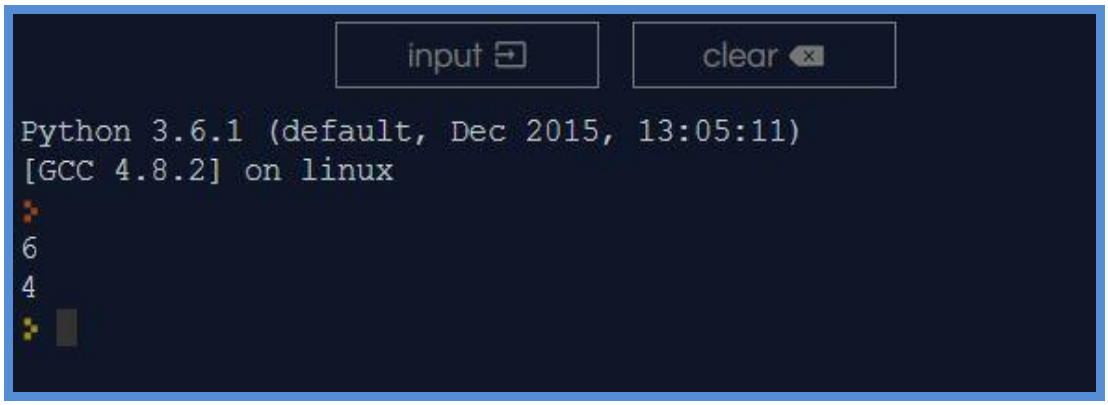


4.5.18. irudia

Zerrenda baten elementu-kopuruaz gain, komeni da jakitea zerrenda batean balio berdina zenbat aldiz agertzen den. Horretarako, Pythonek `count` funtzioa erabiltzen du. Ikus adibide honetan nola erabil daitekeen `count` funtzioa, eta, zein den kodearen exekuzioaren emaitza:



4.5.19. irudia



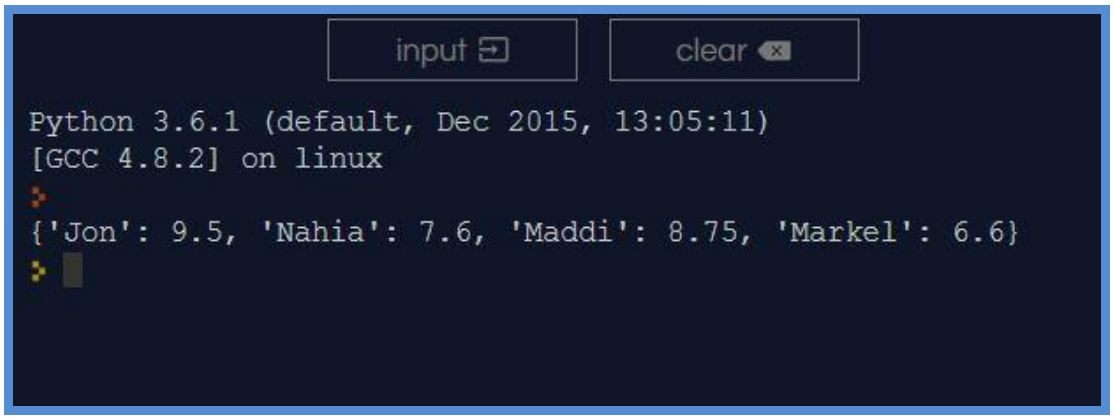
4.5.20. irudia

4.6. HIZTEGIAK

Klabe bat eta balio bat erlazionatzen dituzten zerrendak dira hiztegiak –eta *matrize*-ak ere deitzen zaie–. Hona hemen ikasleen izenak eta batez besteko nota gordetzen duen adibide bat:

4.6.1. irudia

Hau da 4.6.1. irudiko kodearen exekuzioaren emaitza:

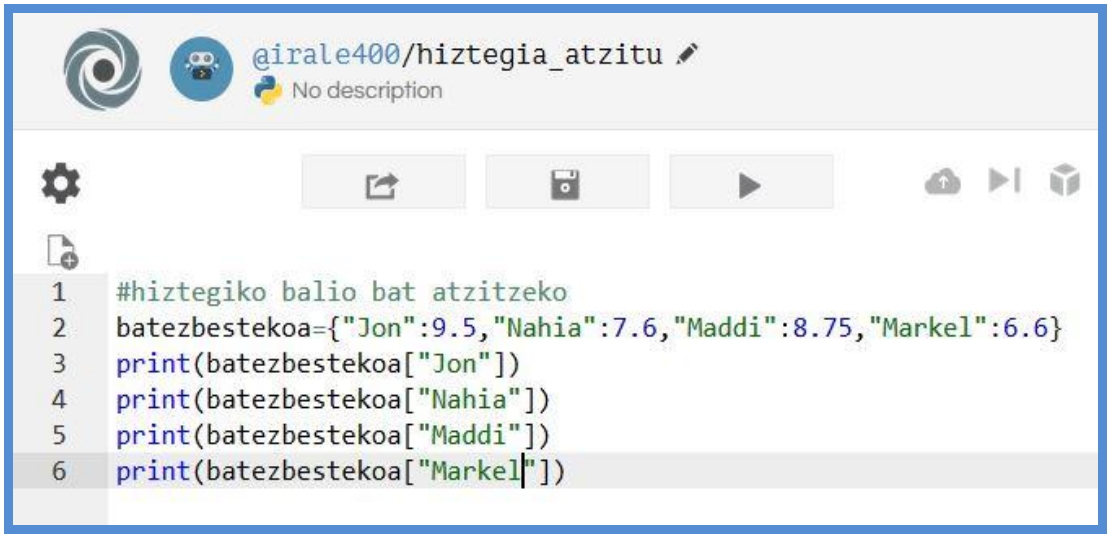


4.6.2. irudia

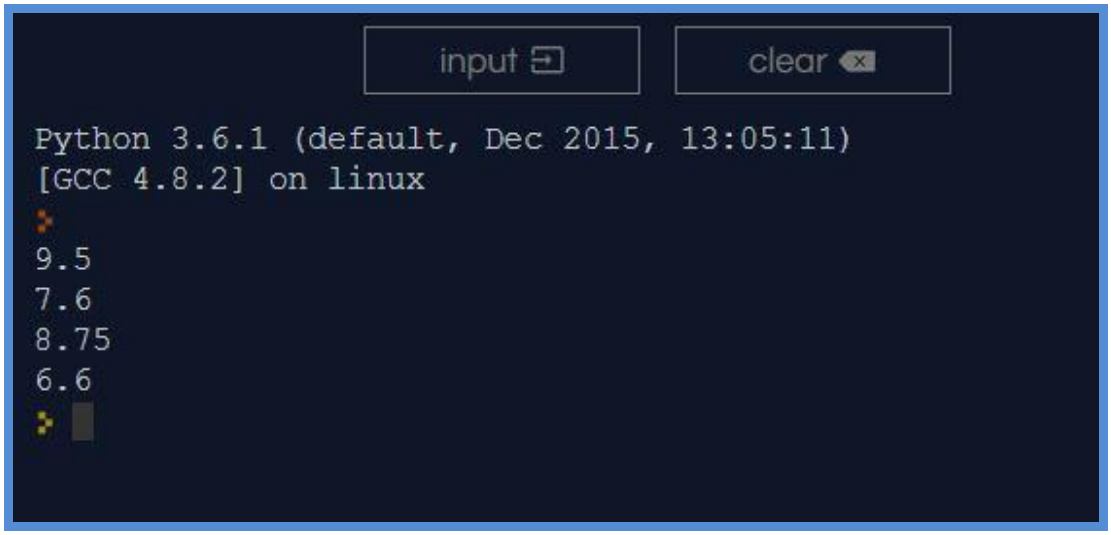
Adibidean ikusten da lehenengo balioa klabeari dagokiola eta bigarrena, berriz, klabeari lotutako balioa dela. Zenbakiak, karaktere-kateak eta balio boolearrak erabili daitezke klabe gisa.

Hiztegien eta zerrenden arteko desberdintasun handiena da zerrendetan balio bat atzitzeko bere indizea erabiltzen dela. Aldiz, hiztegietan programatzaileak definitzen du klabea, eta, horretarako, [] eragilea erabiltzen da. Desberdintasun horretaz gain, bada beste bat ere: zerrendetan partizioa egin daiteke, baina hiztegietan ez.

Hona hemen hiztegiko balio bat atzitzen duen programa baten adibidea eta bere kodearen exkeuzioaren emaitza:



4.6.3. irudia



4.6.4. irudia

5. FLUXU-KONTROLERAKO KONTROL-EGITURAK

5.1 BALDINTZAPEKO EGITURAK

Programa bat exekutatu beharreko agindu multzoa baino zerbait gehiago da. Baldintzapeko aginduek baldintzen balioak aztertzeko aukera ematen dute, eta baldintzaren emaitzaren arabera programak era batera edo beste batera jokatuko du; hau da, baldintzaren emaitza baiezkoa bada, kode zati zehatz bat exekutatuko du, eta, aldiz, baldintzaren emaitza ezezkoa bada, beste kode zati bat. Balio hauekin lan egiteko ezinbestekoak dira eragile logiko eta erlazioak (ikus 4.6 *Boolearrak*).

5.1.1. `if` kontrol-egitura

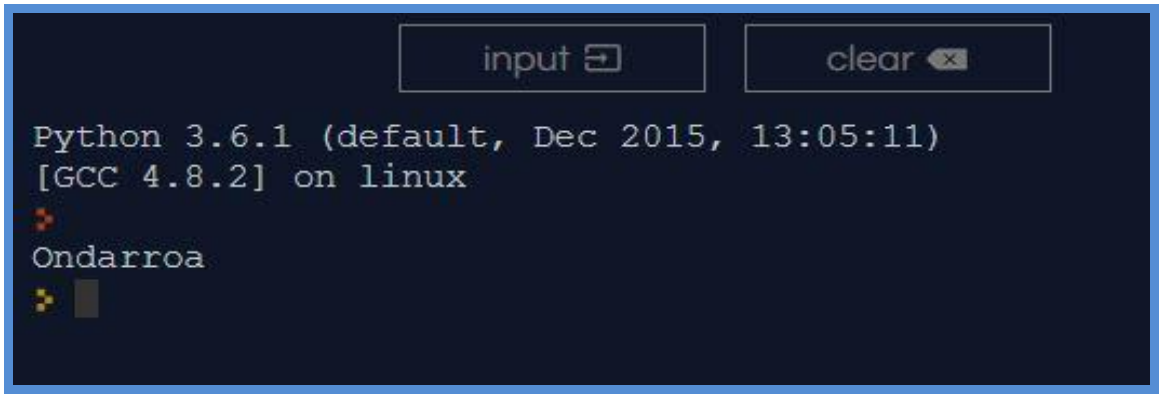
Baldintzapeko egiturarik sinpleenetako bat da `if` kontrol-egitura. Baldintza betetzen denean, egitura honek hurrengo lerroetan dituen sententziak edo aginduak exekutatuko ditu. Aldiz, baldintza betetzen ez denean, ez ditu beteko. Horretarako, `if` eta jarraian aztertu beharreko baldintza edo baldintzak zehaztuko dira lehenengo, eta ondoren `:` ikurra. Halaber, baldintza betez gero, exekutatu beharreko agindua edo agindu multzoa koskatuta (*indentation*) idatziko da; alegia, tabulagailua erabiliz. Pythonen, derrigorrezkoa da agindu multzo hori koskatuta idaztea, bestela programak konpilazio-arazoak ematen ditu. Hona hemen nolakoa den`if` kontrol-egituraren sintaxia:

```
if baldintza :
      1.agindua
      2.agindua
              …
```

5.1.1.1 irudia

`if` kontrol-egituraren adibide bat da honako irudi honetan agertzen dena, eta hurrengoan, berriz, exekuzioaren emaitza:

5.1.1.2 irudia



5.1.1.3 irudia

5.1.2. if … else kontrol-egitura

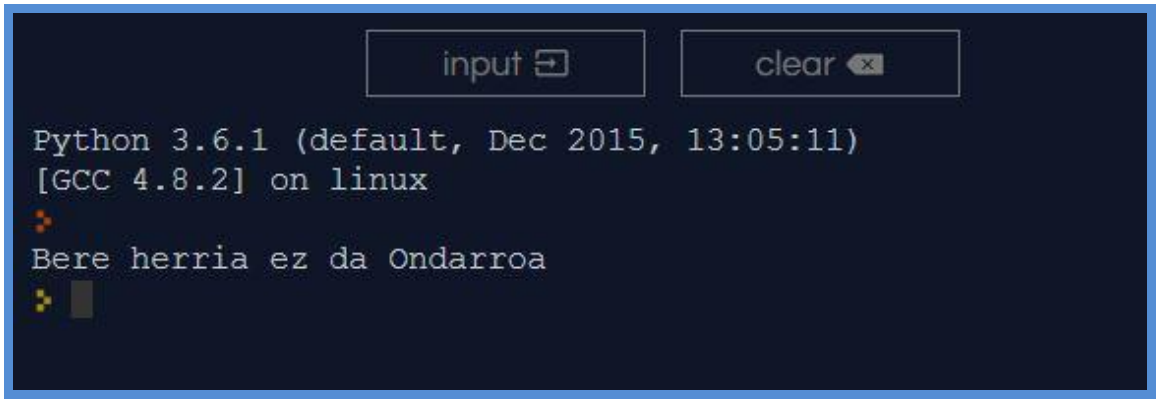
Zer egin beharko luke programatzaileak baldintza betetzean, agindu multzo bat exekutatu nahi badu? Eta baldintza betetzen ez bada, beste agindu multzo bat exekutatu nahi badu? Horretarako, aukera bat `if` motako bi kontrol-egitura erabiltzea izan daiteke, baina bada beste aukera eraginkorrago bat ere: `if … else` kontrol-egitura. Hona hemen `if … else` kontrol-egituraren sintaxia:

```
if baldintza :
      1.agindua
      2.agindua
                …
else:
     1.agindua
      2.agindua
                …
```

5.1.2.1 irudia

Ikus, ondoren, `if … else` egitura nola erabil daitekeen eta bere exekuzioaren emaitza:

5.1.2.2 irudia



5.1.2.3 irudia

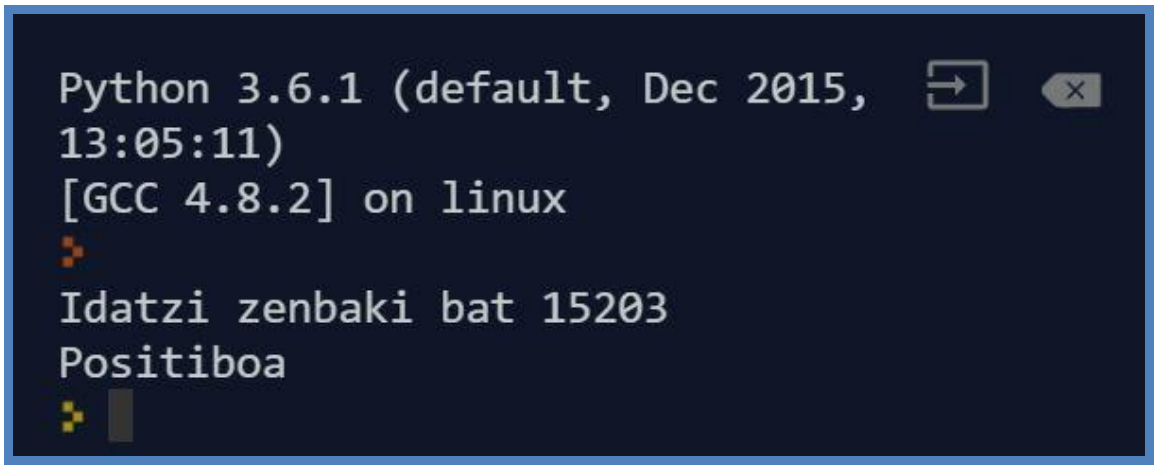
5.1.3. `if … elif … else` kontrol-egitura

Baldintzapeko egituretarako, bada `if` eta `else` aginduez gain, beste agindu bat ere: `elif` agindua, hain zuzen. `elif, else if`-ren laburdura da, eta ˈzenbakia zero baino handiagoa badaˈ irakurtzen da: `elif` zenbakia > ˋ0. Hau da, lehenengo `if`-aren baldintza ebaluatzen da, eta baldintza betetzen bada, dagokion agindu multzoa exekutatuko da. Ondoren, `elif`-ean dagoen baldintza aztertzen da, eta betetzen bada, dagokion agindu multzoa exekutatzen da. Aldiz, `elif`-ko baldintzarik betetzen ez bada, `else`-ri dagokion agindu multzoa exekutatzen da.

Hona hemen `elif` agindua erabiltzen duen programa bat, zeinak aztertzen duen erabiltzaileak teklatu bidez idazten duen zenbakia positiboa, negatiboa edo zero den:

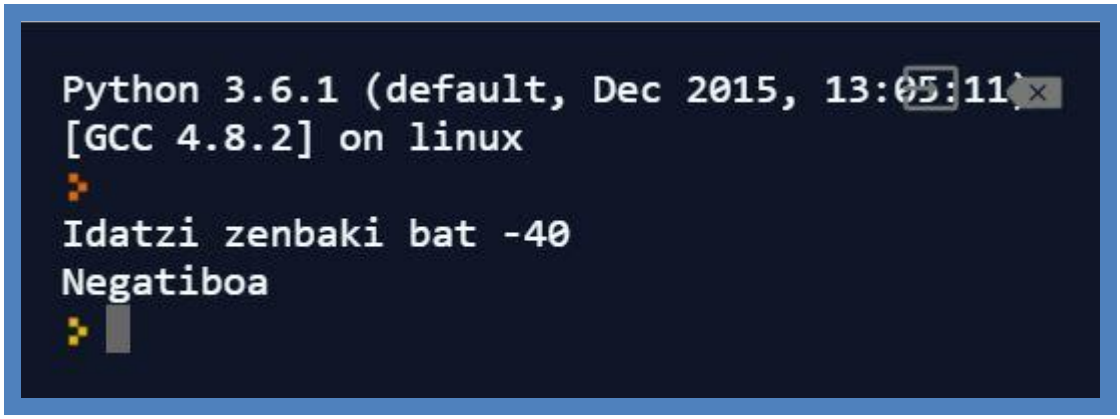
5.1.3.1 irudia

Parametro gisa zenbaki positiboa idazten denean, hau da programaren exekuzioaren emaitza:



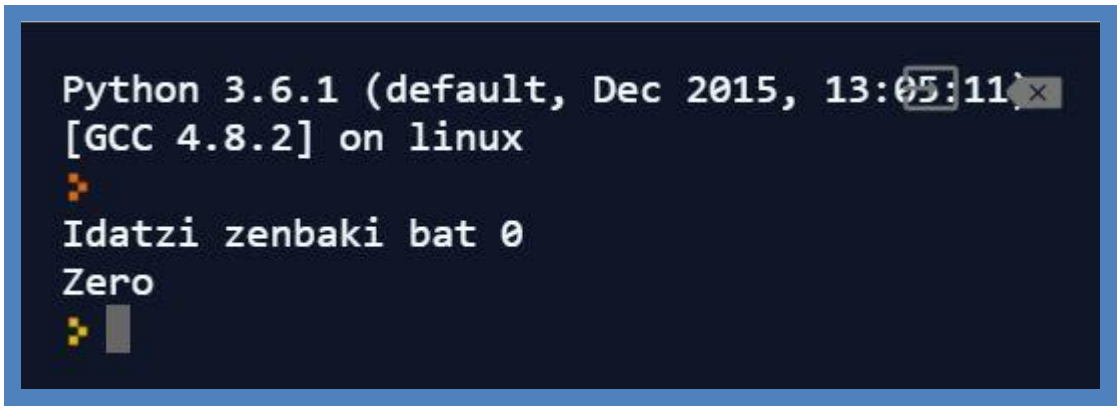
5.1.3.2 irudia

Parametro gisa zenbaki negatiboa idazten denean, hemen programaren exekuzioaren emaitza:



5.1.3.3 irudia

Aldiz, parametro gisa zero zenbakia idazten denean, exekuzioaren beste emaitza hau emango du:

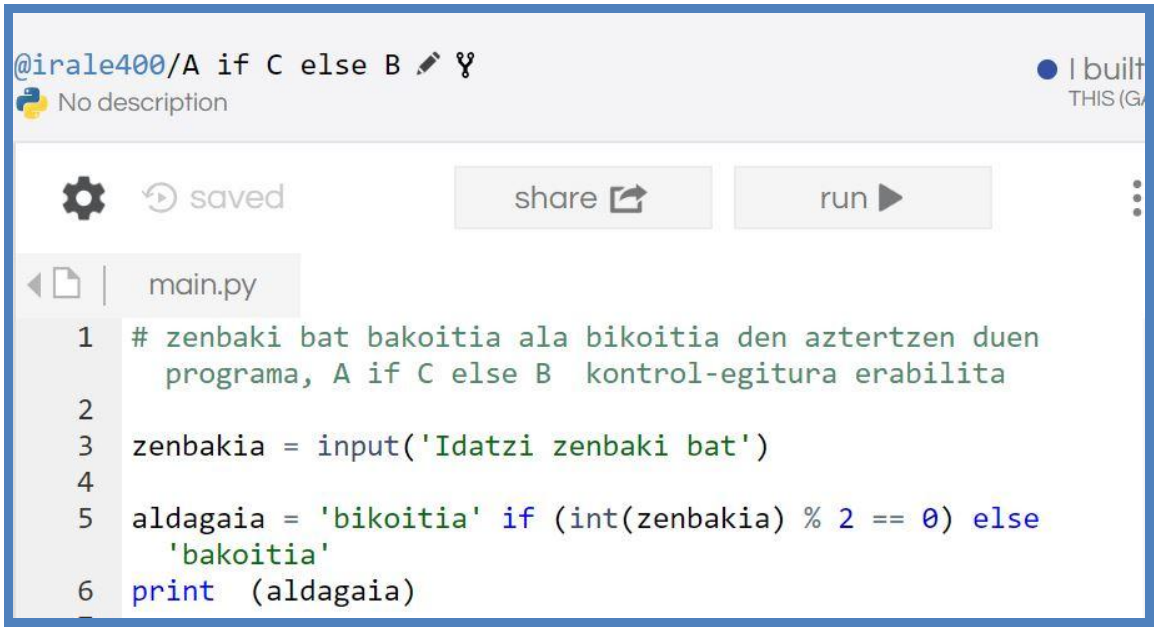


5.1.3.4 irudia

5.1.4. A `if` C `else` B kontrol-egitura

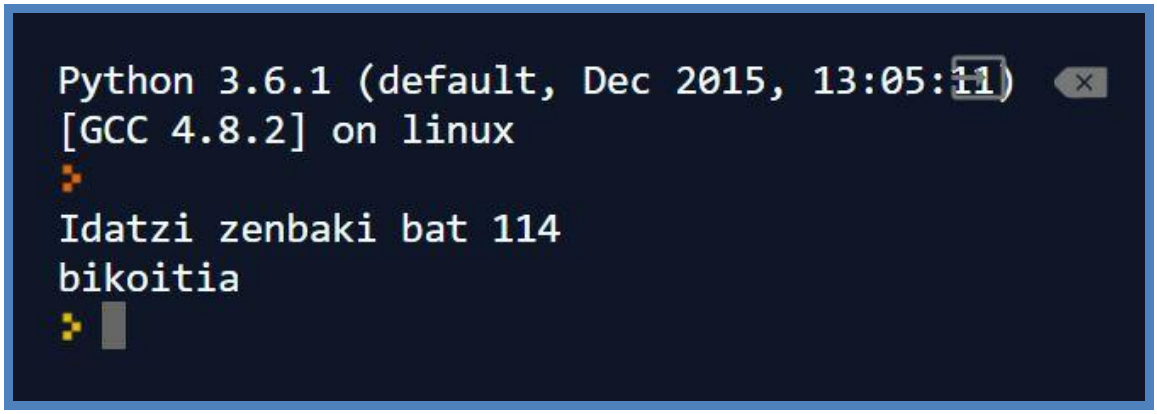
Aipatutako baldintzapeko kontrol-egiturez gain, bada beste programazio-lengoaietan ? eragilearen moduko funtzionamendua duen kontrol-egitura; hau da, `if … else` definitzeko modu laburragoa. Kontrol-egitura honetan, C baldintza exekutatzen da, eta betetzen bada, A itzultzen du,aldiz, betetzen ez bada, B itzultzen du: A `if` C `else` B

Hona hemen beste adibide bat:



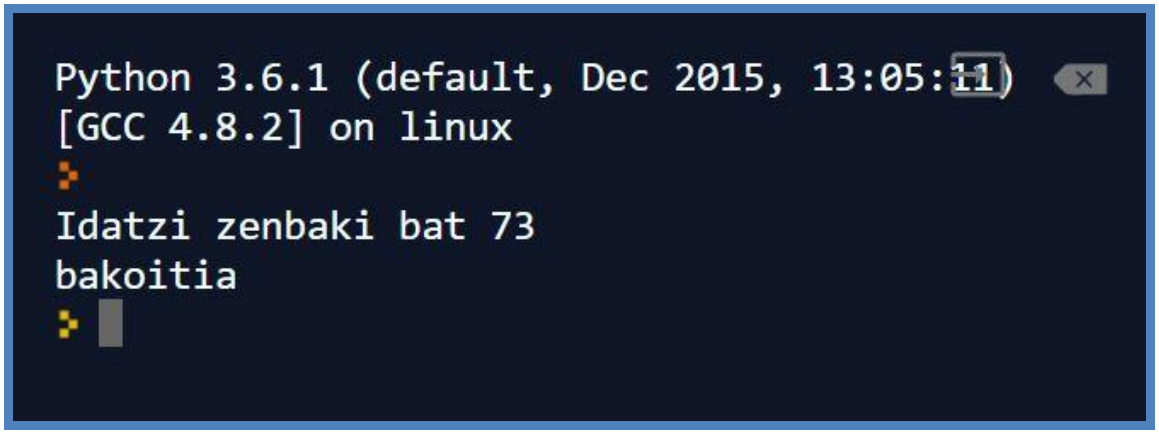
5.1.4.1 irudia

Parametro gisa zenbaki bikoitia idazten denean, hona hemen programaren exekuzioaren emaitza:



5.1.4.2 irudia

Aldiz, parametro gisa zenbaki bakoitia idazten bada, hau izango da programaren exekuzioaren emaitza:



5.1.4.3 irudia

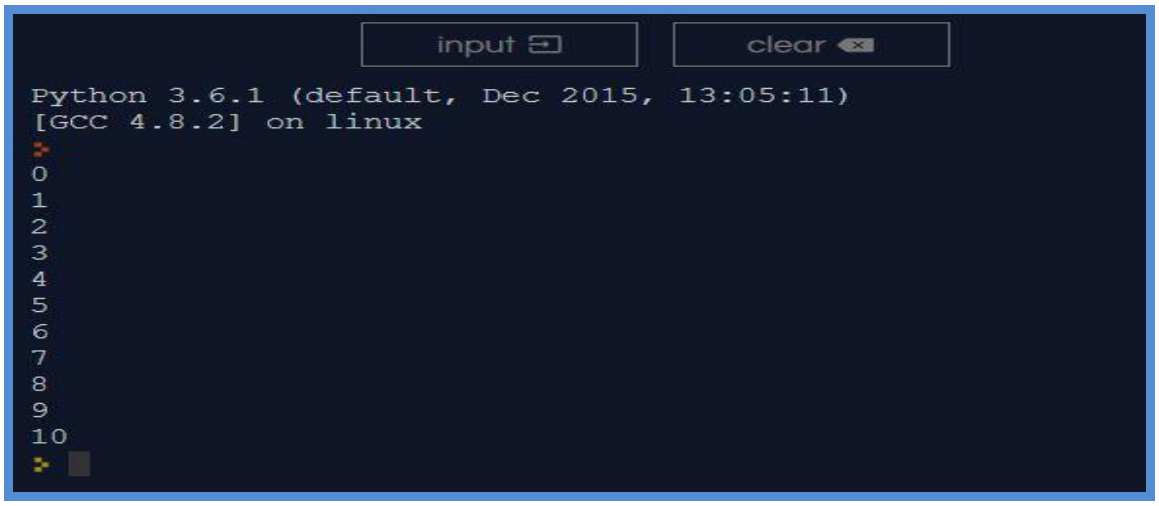
5.2. BEGIZTAK

Aurretik aipatu bezala, baldintzapeko kontrol-egiturek baldintzen arabera agindu multzo bat edo beste bat exekutatzeko aukera eskaintzen dute. Aldiz, begiztek agindu multzoa nahi beste aldiz errepikatzeko balio dute,betiere baldintza betetzen den bitartean. Begiztei dagokienez, `while` eta `for … in` kontrol-egiturak aztertuko dira ondoren:

5.2.1. `while` kontrol-egitura

Baldintza betetzen den bitartean, `while` (bitartean) begiztak kode zati bat exekutatzen du. Hona hemen *zenbakia* izeneko aldagaia 10 baino txikiagoa den bitartean, aldagaiaren balioa pantailan bistaratuko duen programaren adibide bat eta dagokion emaitza:

5.2.1.1 irudia



5.2.1.2 irudia

Programa horren funtzionamendua honakoa da:

Hasteko *zenbakia* izeneko aldagaia 0 baliora hasieratzen da. Ondoren, `while` kontrol-egituran *zenbakia* izeneko aldagaiaren balioa 10 edo 10 baino txikiagoa den aztertzen da. Lehenengo iterazioan, *zenbakia* aldagaiak 0 balioa du, eta, ondorioz, `while` barneko aginduak exekutatuko dira; hau da, `print` (zenbakia) aginduak 0 bistaratuko du pantailan, eta, ondoren, *zenbakia* aldagaiak zuen balioari bat balioa gehituko zaio; beraz, 0 + 1 = 1 balio izango du. Hona hemen programaren exekuzioaren balio guztiak azaltzen dituen taula:

Iterazioa	*Zenbakia* aldagaiaren balioa	Pantailan bistaratuko dena	Zenbakia = zenbakia +1
1	0	0	0 + 1 = 1
2	1	1	1 + 1 = 2
3	2	2	2 + 1 = 3
4	3	3	3 + 1 = 4
5	4	4	4 + 1 = 5
6	5	5	5 + 1 = 6
7	6	6	6 + 1 = 7
8	7	7	7 + 1 = 8
9	8	8	8 + 1 = 9
10	9	9	9 + 1 = 10
11	10	10	10 + 1 = 11

5.2.2. `for … in` kontrol-egitura

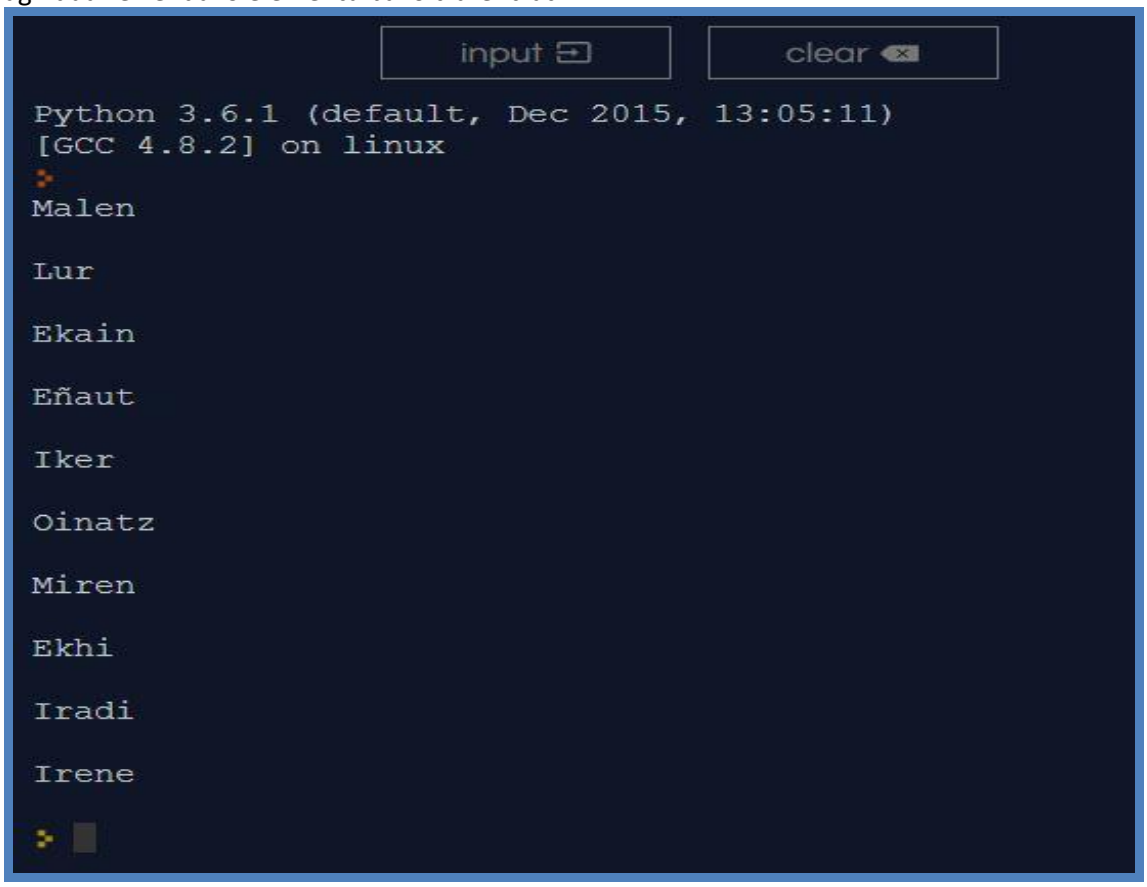
Pythonen, `for … in` kontrol-egitura zerrenda batean lan egiteko erabiltzen da, eta bere sintaxia hau da:

```
for elementua in zerrenda:
      1.agindua
      2.agindua
         …
```

`for … in` kontrol-egiturarekin egindako programa baten adibidea da honakoa, zeinak zerrenda bateko elementuak bistaratzen dituen. Ondoren, programa horren exekuzioaren emaitza:

5.2.2.1 irudia

Programaren funtzionamendua ulertzeko, ikus 5.2.2.2. irudian nola exekutatzen den `print` agindua zerrendako elementu bakoitzarentzat:

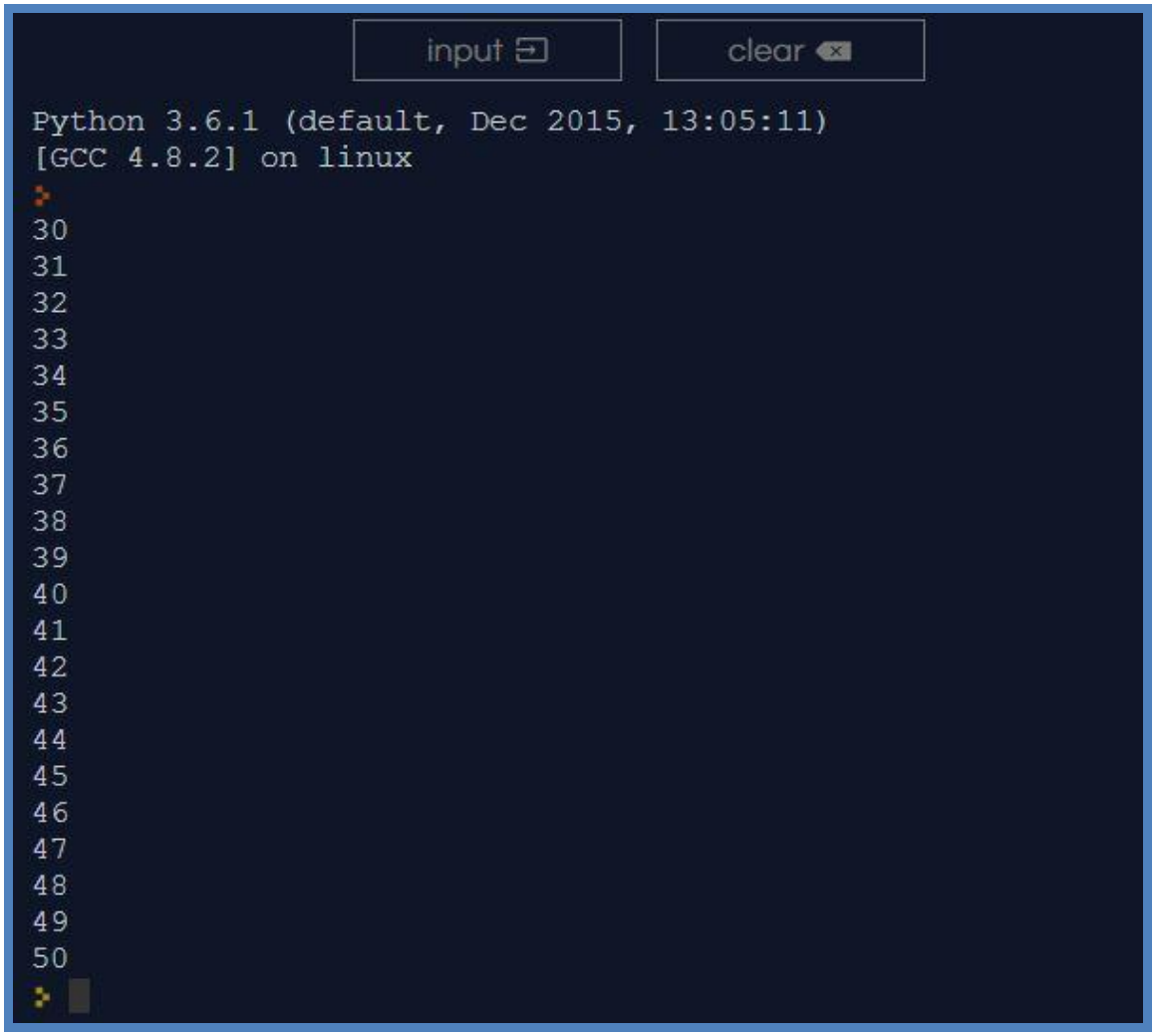


5.2.2.2 irudia

Pythonen ez bezala, C edo Java programazio-lengoaietan `for` kontrol-egitura tarte jakin bateko balioak bistaratzeko ere erabiltzen da. Beraz, zer egin Pythonen 30 eta 50 arteko zenbakiak inprimatu nahi badira? Horretarako, `range` (tartea) metodoa erabilita zerrenda bat sortu behar da, eta, ondoren, zerrenda horretako elementuak bistaratu.

Hona hemen 30 eta 50 bitartean dauden zenbakiak –biak barne– bistaratzen dituen programa baten adibidea, eta bere exekuzioaren emaitza:

5.2.2.3 irudia

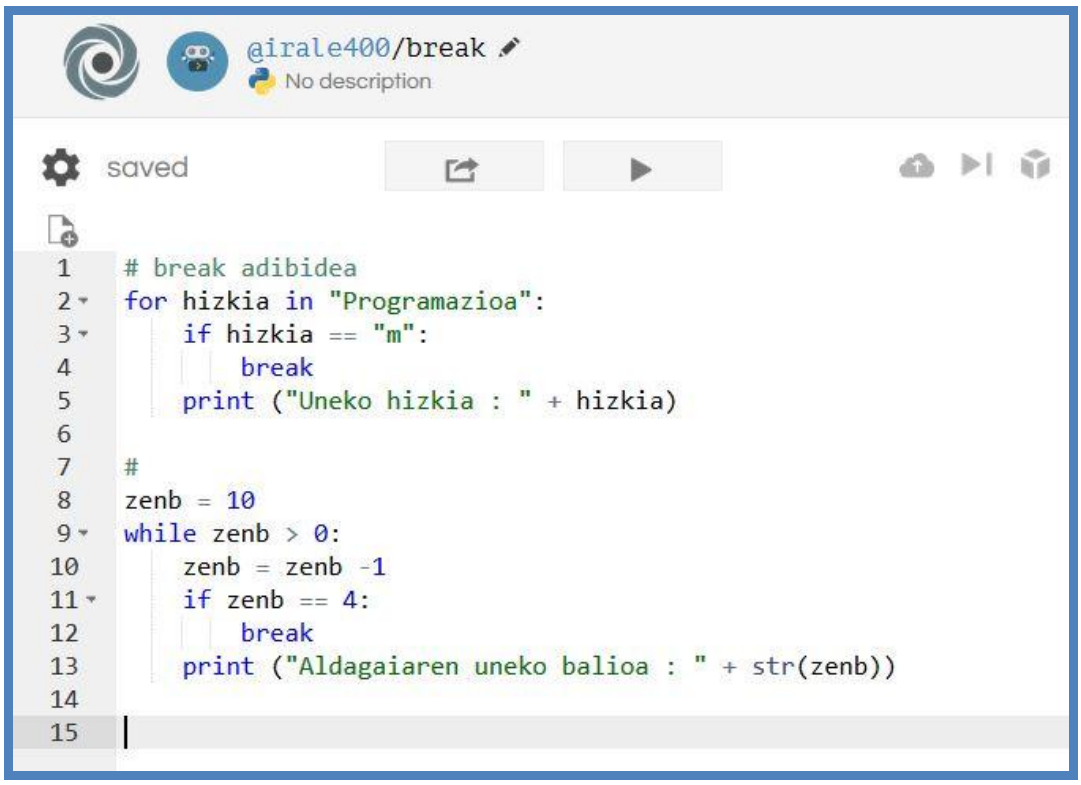


5.2.2.3 irudia

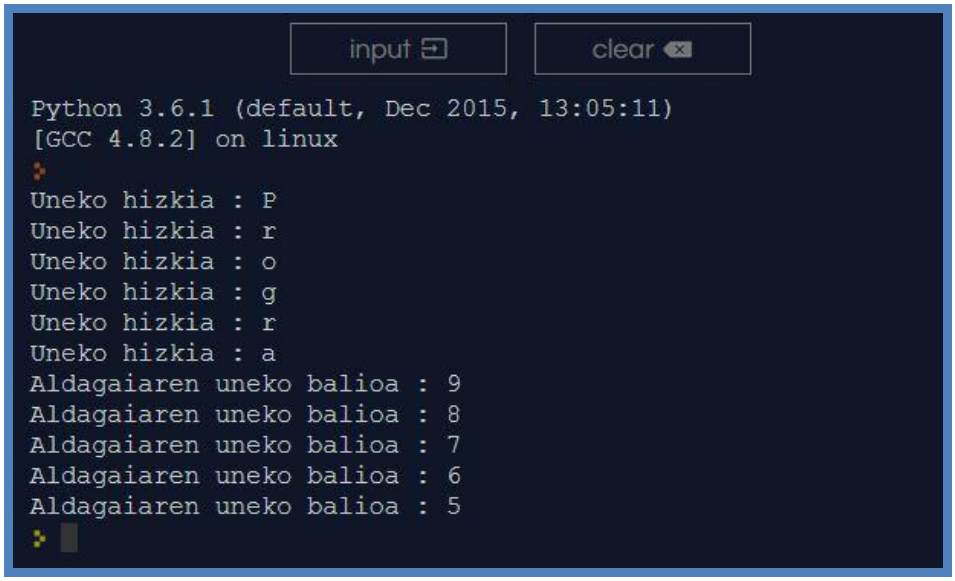
5.3. `break` ETA `continue` AGINDUAK

Pythonen beste programazio-lengoaietan bezala, `break` eta `continue` aginduak `for` eta `while` bezalako kontrol-egituretan erabiltzen dira, hau da, begiztetan.

Pythonen, **`break`** agindua exekutatzean, begizta moztu egiten da, eta hurrenez hurren datorren agindu multzoa exekutatzen jarraitzen du. Hona hemen adibide bat eta bere exekuzioaren emaitza:

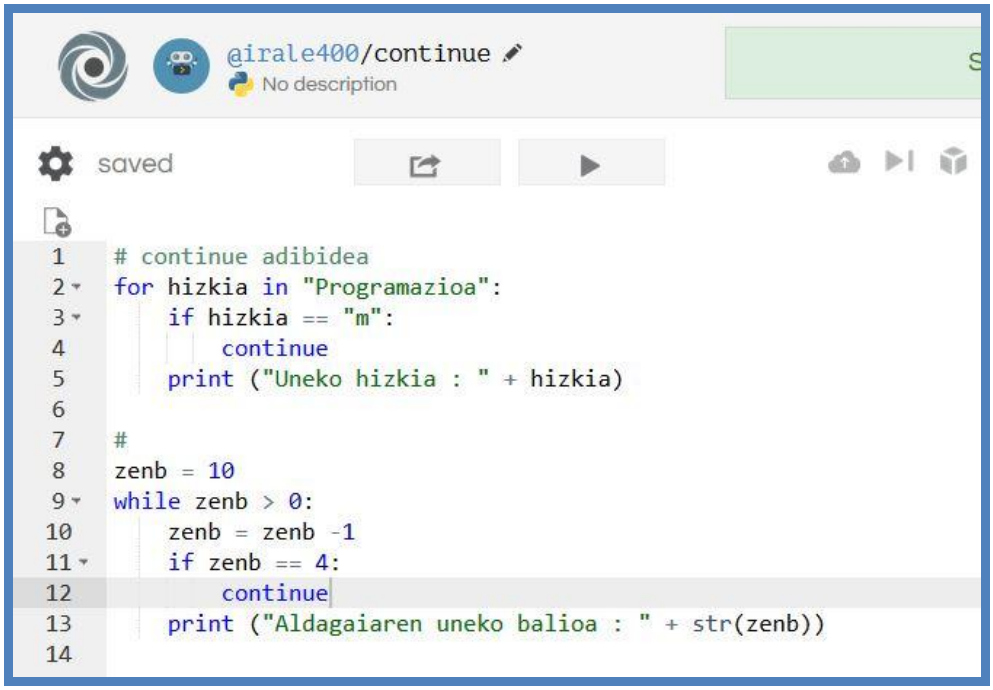


5.3.1 irudia

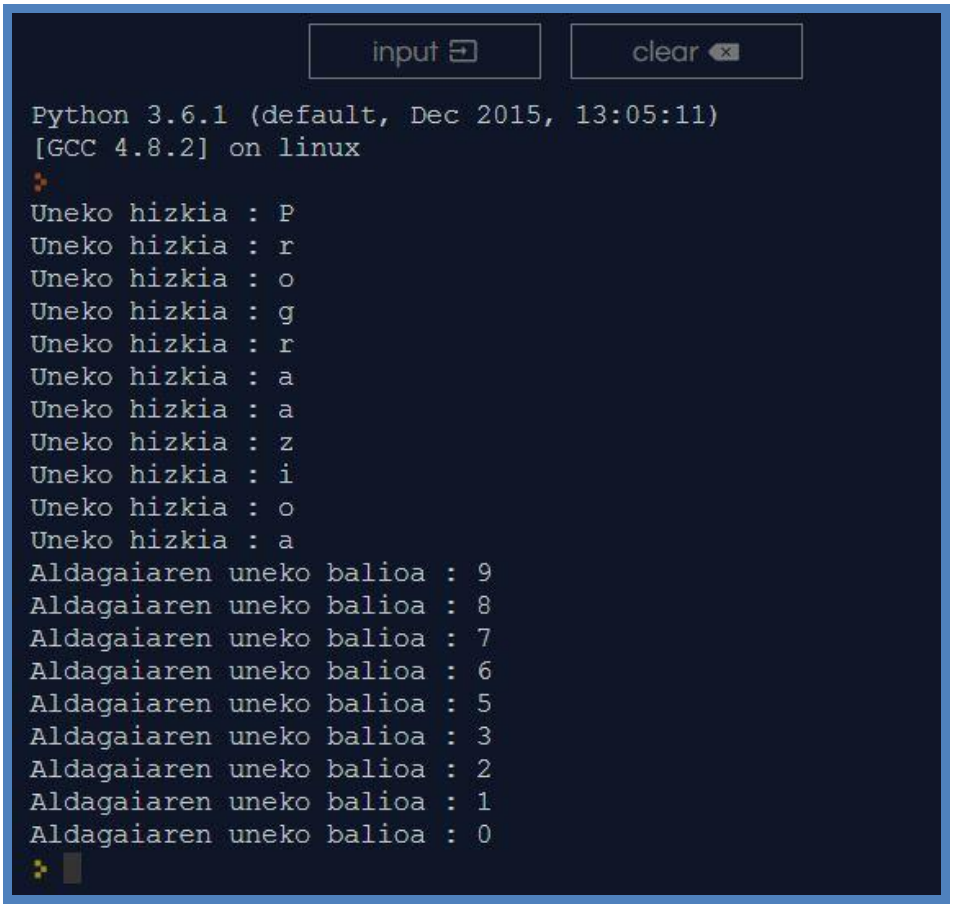


5.3.2 irudia

Aldiz, **`continue`** agindua erabiltzen bada, begiztaren hasierara itzultzen da programa, ez baititu jarraian datozen aginduak kontuan hartuko; beraz, hurrengo iterazioa exekutatzen da. Honela:



5.3.3 irudia


input
clear
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Uneko hizkia : P
Uneko hizkia : r
Uneko hizkia : o
Uneko hizkia : g
Uneko hizkia : r
Uneko hizkia : a
Uneko hizkia : a
Uneko hizkia : z
Uneko hizkia : i
Uneko hizkia : o
Uneko hizkia : a
Aldagaiaren uneko balioa : 9
Aldagaiaren uneko balioa : 8
Aldagaiaren uneko balioa : 7
Aldagaiaren uneko balioa : 6
Aldagaiaren uneko balioa : 5
Aldagaiaren uneko balioa : 3
Aldagaiaren uneko balioa : 2
Aldagaiaren uneko balioa : 1
Aldagaiaren uneko balioa : 0

5.3.4 irudia

6. FUNTZIOAK

Programazioan, **funtzioa** esaten zaio ataza batzuk egin eta balio bat itzultzen duen izendatutako kode-zatiari. Aldiz, **prozedura** esaten zaio ataza batzuk egin eta baliorik itzultzen ez duen izendatutako kode-zatiari.

Pythonen izendatutako kode-zati guztiek itzultzen dute balioren bat, eta izan kontuan `None` ere (hutsa edo nulua) balio bat dela; beraz, ez da existitzen prozedurarik Python programazio-lengoaian.

Programazioan, batzuetan, kode zati bera behin eta berriz errepikatzen da; hori dela eta, komeni da programa batean sarri erabili behar diren kode zatiak funtzio bihurtzea, kode-zati horiek behin baino gehiagotan berrerabiltzeko. Halaber, funtzioak erabiltzeak kode argiagoa eta irakurterrazagoa sortzen laguntzen du.

Pythonen funtzioak honela deklaratzen dira:

```
def funtzioaren_izena (1parametroa, 2parametroa):

Print(1parametroa)

Print(2 parametroa)
```

6.1. irudia

Beraz, honako hurrenkera hau behar du funtzioaren definizioak:

- Lehenengo, `def` hitz erreserbatua, eta, jarraian, programatzaileak funtzioari jarritako izena.
- Ondoren, parentesi artean eta koma bidez , funtzioak hartuko dituen argumentuak.
- Bukatzeko, bi puntuak eta aginduekin osatutako kode-zatia, zeinak hurrengo lerroetan funtzioak exekutatuko dituen.

Bestalde, funtzioen deklarazioaren inguruan argi geratu behar da honako hau ere:

Funtzioa deklaratzean, izena bakarrik jartzen zaio agindu multzoari edo kode-zatiari, ez da besterik egiten. Beraz, funtzioa definitzean, funtzioak duen kodea ez da exekutatzen, eta kode hori exekutatu nahi bada, funtzioari deitu behar zaio.

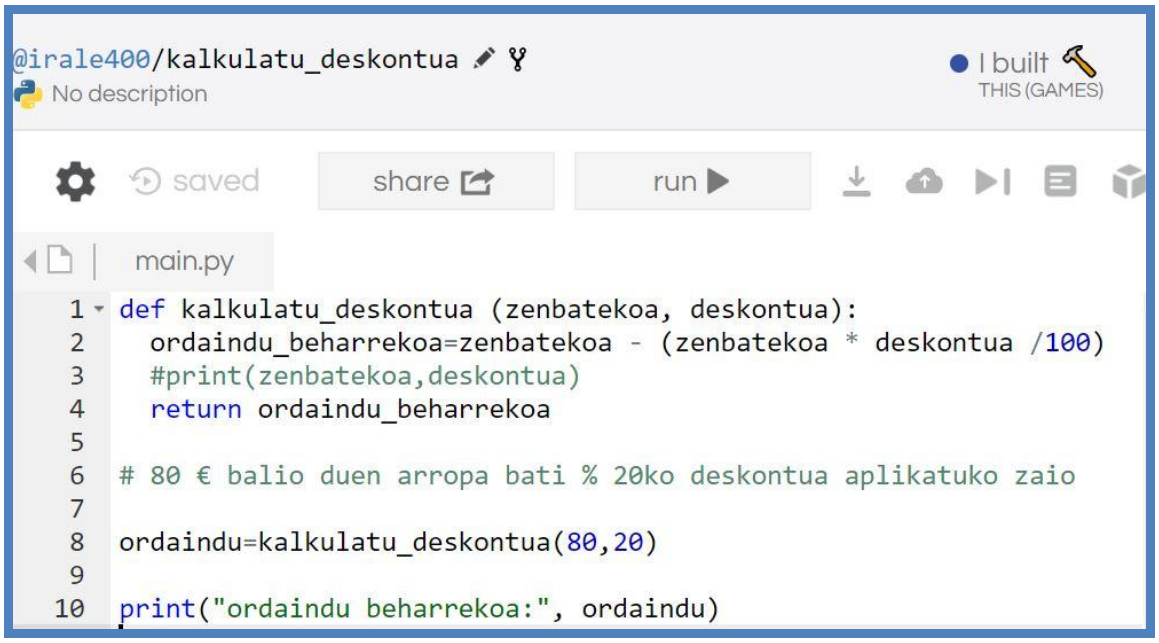
Funtzioari deitzeko, lehenengo exekutatu nahi den funtzioaren izena idatzi behar da, eta, ondoren, parentesi artean eta koma bidez banatuta, funtzioari parametro moduan pasatuko zaizkion balioak idatzi behar dira.

```
funtzioaren_izena (1parametroa, 2parametroa)
```

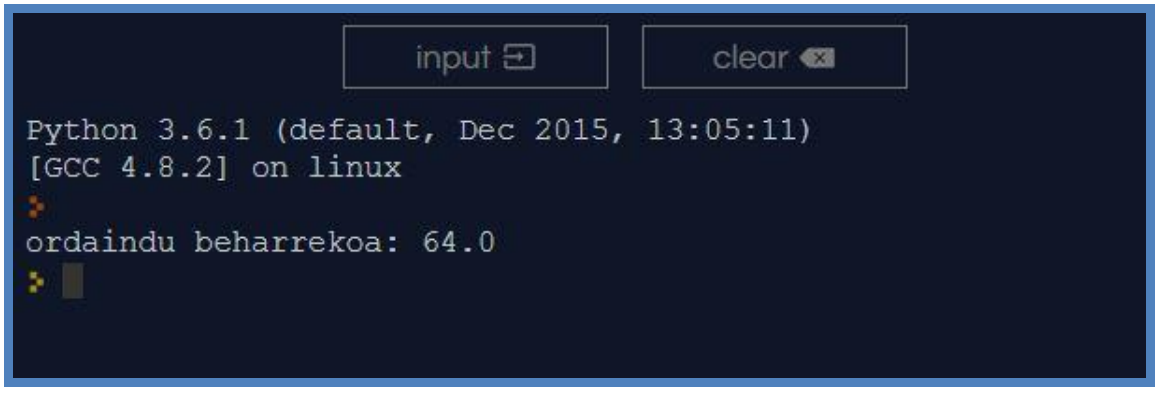
6.2. irudia

Kodeak programatzaileak agindutako egiteko, funtzioari deitzen zaionean, ezinbestekoa da parametroak funtzioan definitutako hurrenkera berean pasatzea.

Emandako azalpena argitzeko, hona hemen funtzio baten definizioaren erabilera-kasu bat eta dagokion exekuzioaren emaitza:



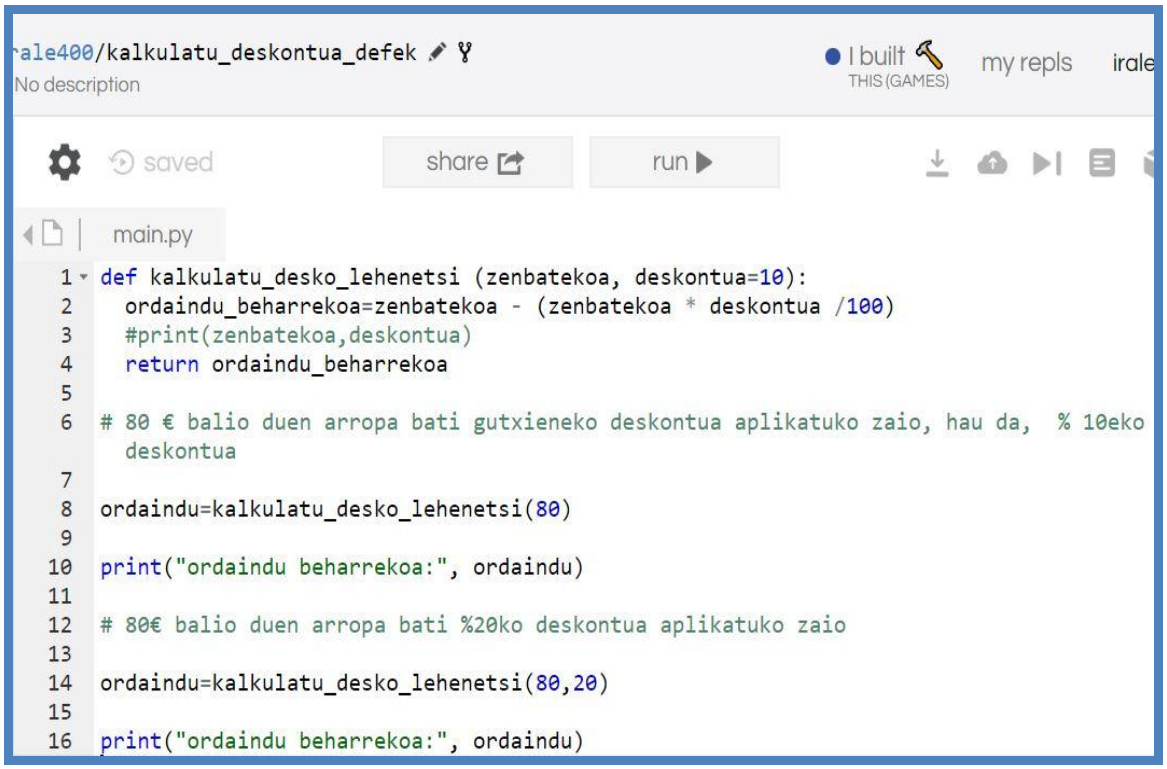
6.3. irudia



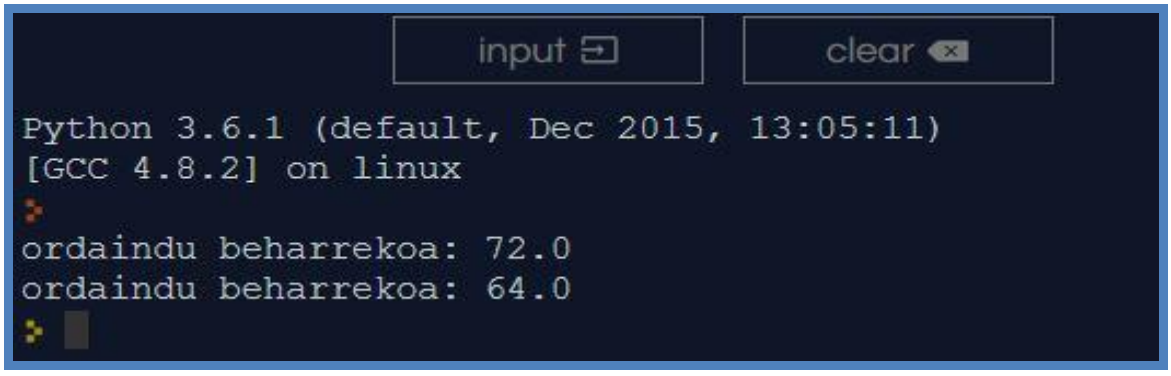
6.4. irudia

Bada funtzioak definitzean kontuan izan beharreko azken irizpide bat ere: funtzioaren definizioan zehaztutako parametro-kopuru berarekin deitu behar zaio funtzioari. Horrela egitenez bada, Pythonek errorea emango du.

Hala ere, esan beharra dago Pythonek parametroei lehenetsitako balioak ere ezartzeko aukera ematen duela. Horrelakoetan, funtzioari baliorik pasatu ezean, funtzioak lehenetsitako balioak hartzen ditu kontuan. Hona hemen azaldutakoa ulertzeko balio duen programa bat eta bere exekuzioaren emaitza:



6.5. irudia



6.6. irudia

ARIKETAK

7. ARIKETAK

Programak egiteko garaia da. Atalka antolatutako ariketak dira ondoren aurkituko dituzunak. Egin ariketak, zehaztutako baldintzak betez.

1. ATALA: ALGORITMOAK

1.1. ariketa

Idatzi arrautza bat frijitzeko eman beharreko pausoak azaltzen dituen algoritmoa.

Ariketa ebazteko modu bat:

1. Isuri olioa zartaginera.
2. Piztu sua.
3. Jarri zartagina sutan.
4. Berotu olioa.
5. Apurtu arrautza.
6. Bota arrautza zartaginera.
7. Bota olioa arrautzaren gainean bitsaderarekin.
8. Kolore zuria hartutakoan, atera arrautza platerera.

1.2. ariketa

Idatzi algoritmo bat zeinak azaltzen dituen zinema-aretoan gustuko pelikula ikusteko eman beharreko urratsak.

Ariketa ebazteko modu bat:

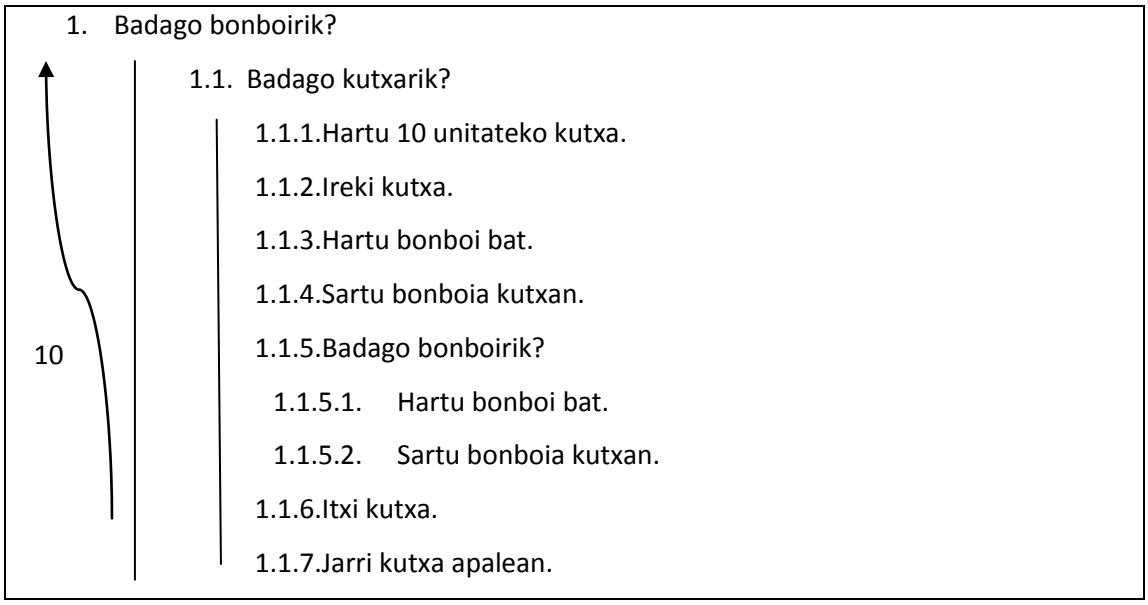
1. Aukeratu pelikula.
    1.1. Bilatu egunkaria.
    1.2. Bilatu ikuskizunen saila egunkarian.
        1.2.1. Ikuskizunen saila aurkitu bitartean:
            1.2.1.1. Pasatu orrialdea.
    1.3. Aukeratu gustuko filma.
        1.3.1. Filmak dauden bitartean:
            1.3.1.1. Aztertu filma.
            1.3.1.2. Gustukoa bada:
                1.3.1.2.1. Gogoratu.
2. Joan zinema-aretora.
3. Erosi sarrera.
    3.1. Badago sarrerarik?
        3.1.1. Sarrerak erosteko, ilararik badago?
            3.1.1.1. Jarri ilaran.
            3.1.1.2. Leihatilara heldu bitartean:
                3.1.1.2.1. Itxaron aurrekoa mugitu arte.
                3.1.1.2.2. Mugitu aurrera.
        3.1.2. Erosi sarrera.
    3.2. Bestela, joan etxera.
4. Ikusi filma.
    4.1. Sartu aretora.
    4.2. Eman sarrera ate-zaintzaileari.
    4.3. Begiratu eserlekuaren zenbakia.
    4.4. Bilatu eserlekua.
        4.4.1. Bilatutako eserlekua ez bada txartelean izendatutako:
            4.4.1.1. Joan hurrengo eserlekura.
    4.5. Eseri eta ikusi filma.

1.3. ariketa

Idatzi algoritmo bat bonboi-makina batek egiten duen ekintza hauetarako.

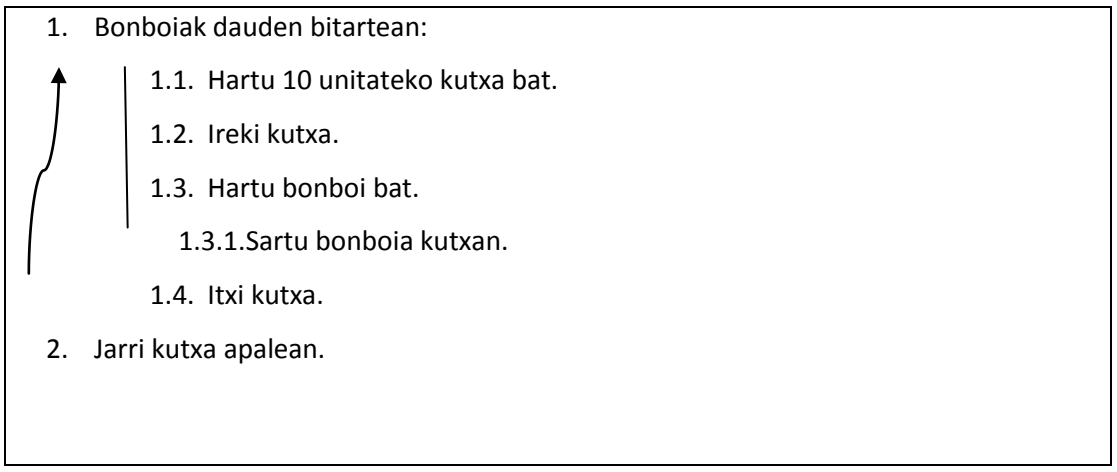
1. Bete bonboiekin 10 unitateko kutxa bat.

Ariketa ebazteko modu bat:



2. Pentsatu kutxa-kopuru infinitua duzuoa, eta bonboi-kopurua, berriz, 10 zenbakiaren multiploa. Hori jakinda, idatzi algoritmo bat zeinak bonboi guztiak kutxetan sartuko dituen.

Ariketa ebazteko modu bat:



3. Bonboi-kopuru mugagabea duzula jakinda, idatzi algoritmo bat zeinak kutxa guztiak bonboiz beteko dituen.

Ariketa ebazteko modu bat:

1. Kutxak dauden bitartean:
    1.1. Hartu 10 unitateko kutxa bat.
    1.2. Ireki kutxa.
    1.3. Hartu bonboi bat.
    1.4. Sartu bonboia kutxan.
    1.5. Itxi kutxa.
2. Jarri kutxa apalean.

1.4. ariketa

Zurezko blokeak moldekatuz zurezko dadoak egiten dituen makina batek honako ekintza hauek egiten daki:

- Hartu zurezko blokea.
- Moldekatu blokea.
- Margotu kuboa.
- Idatzi zenbakia.
- Biratu kuboa.
- Sartu kuboa kutxan.
- Ziurtatu blokerik soberan dagoen.

1. Hori guztiori kontuan izanda, idatzi dado bat eraikitzeko algoritmoa.

Ariketa ebazteko modu bat:

1. Badago blokerik?
   - 1.1. Hartu zurezko blokea.
   - 1.2. Moldekatu blokea.
   - 1.3. Margotu kuboa.
   - 1.4. Idatzi zenbakia.
   - 1.5. Biratu kuboa.
   - 1.6. Idatzi zenbakia
   - 1.7. Sartu kuboa kutxan.

5

2. Idatzi algoritmo bat blokeak bukatu arte dadoak eraikitzeko.

Ariketa ebazteko modu bat:

1. Blokeak dauden bitartean:
   - 1.1. Hartu zurezko blokea.
   - 1.2. Moldekatu blokea.
   - 1.3. Margotu kuboa.
   - 1.4. Idatzi zenbakia.
   - 1.5. Biratu kuboa.
   - 1.6. Idatzi zenbakia.
   - 1.7. Sartu kuboa kutxan.

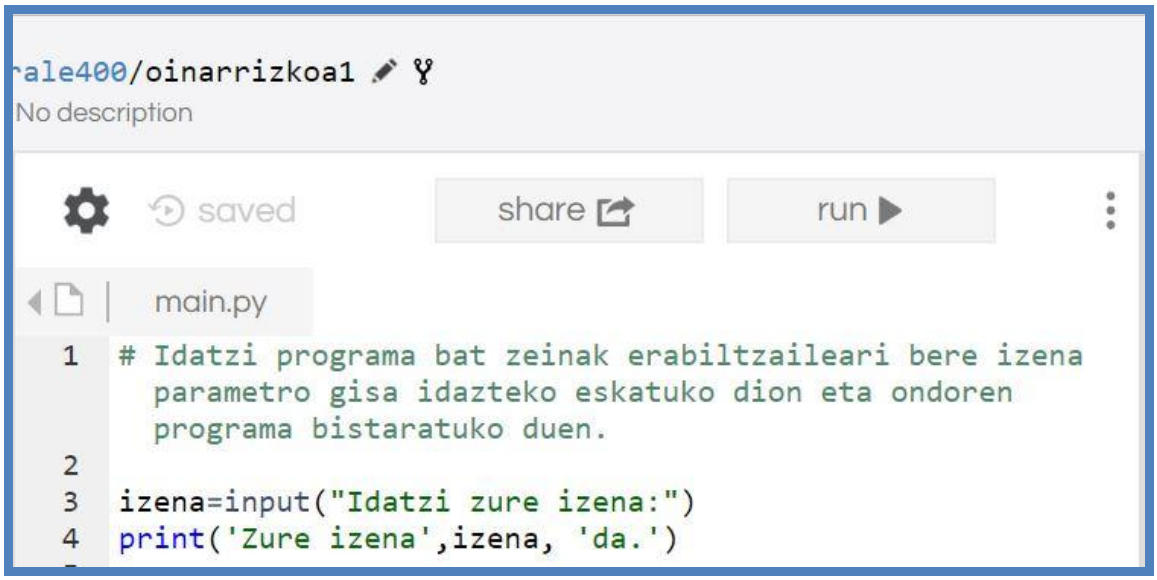
5

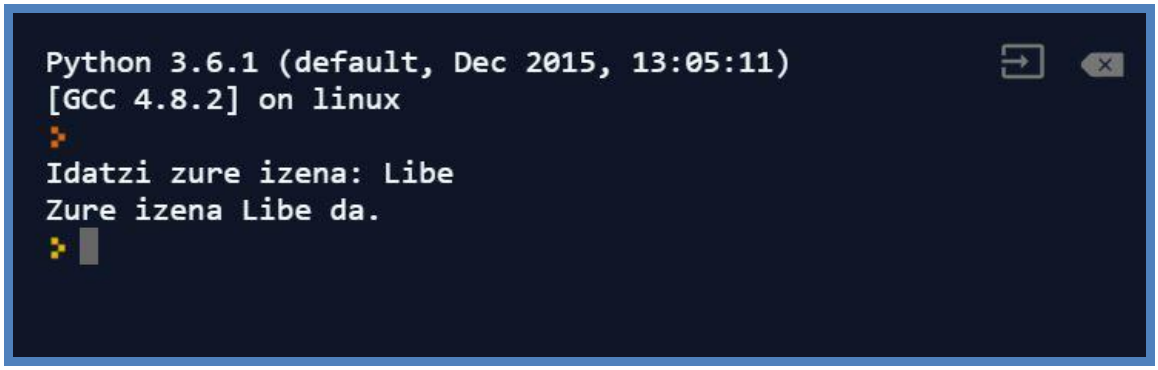
2. ATALA: OINARRIZKO AGINDUAK

2.1. ariketa

Idatzi programa bat zeinak erabiltzaileari bere izena parametro gisa idazteko eskatuko dion eta ondoren programa bistaratuko duen.

Ariketa ebazteko modu bat:

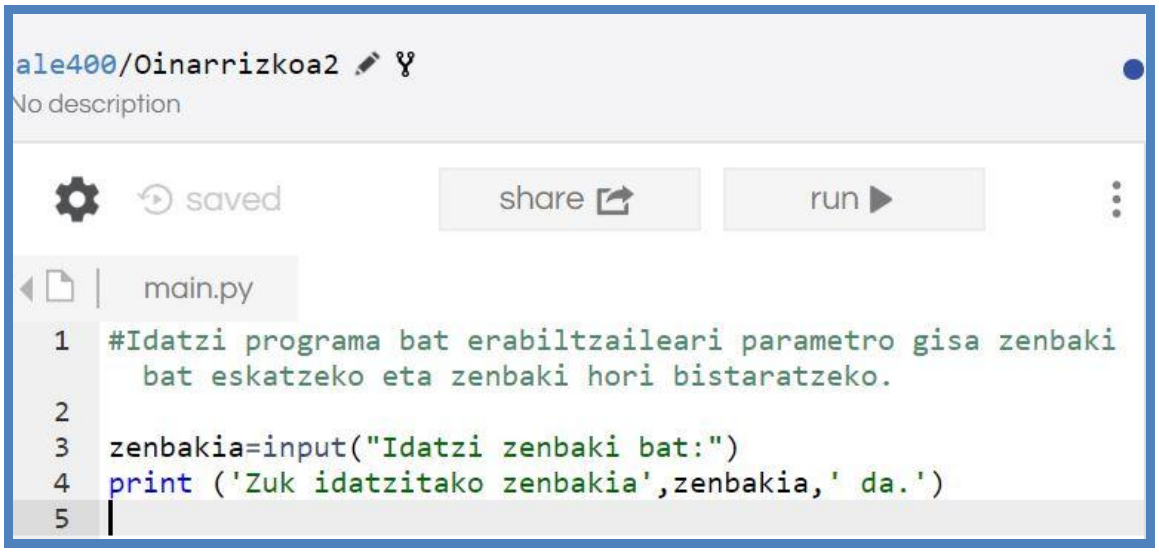


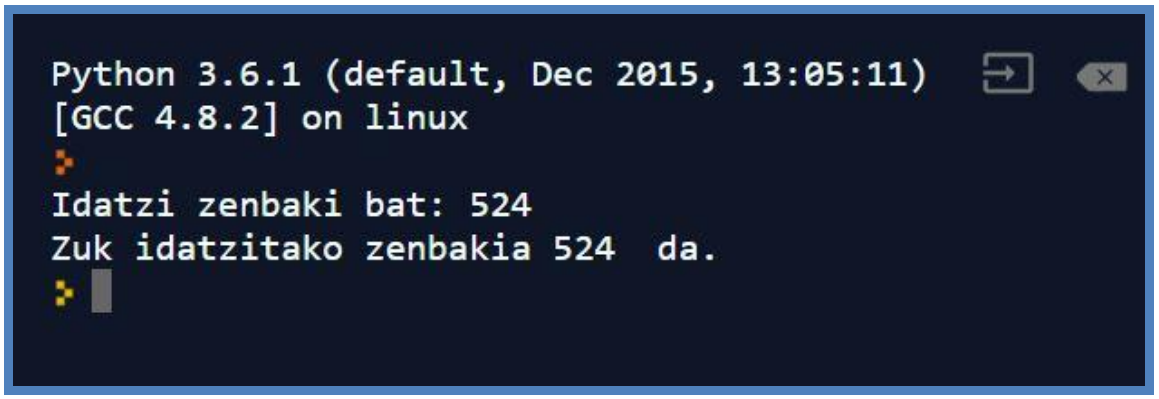


2.2. ariketa

Idatzi programa bat balioko duena erabiltzaileari parametro gisa zenbaki bat eskatzeko eta zenbaki hori bistaratzeko.

Ariketa ebazteko modu bat:

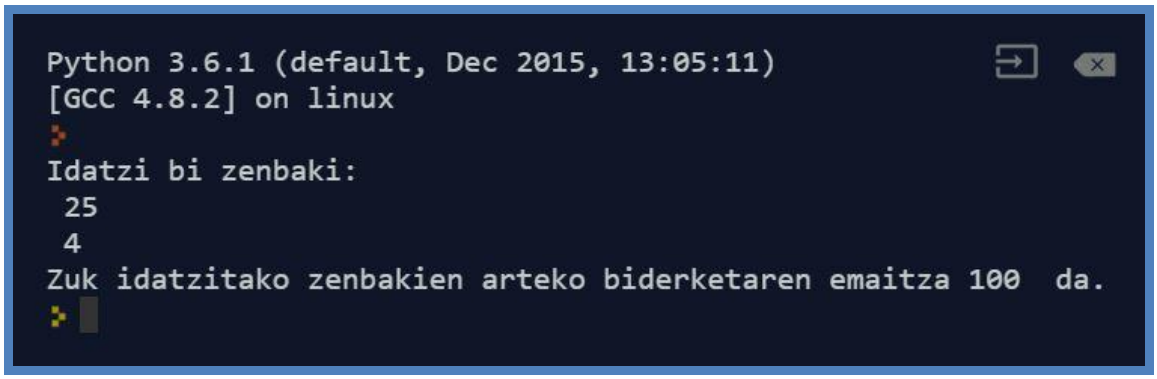




2.3. ariketa

Idatzi programa bat erabiltzaileari parametro gisa bi zenbaki eskatzeko eta haien arteko biderketa bistaratzeko.

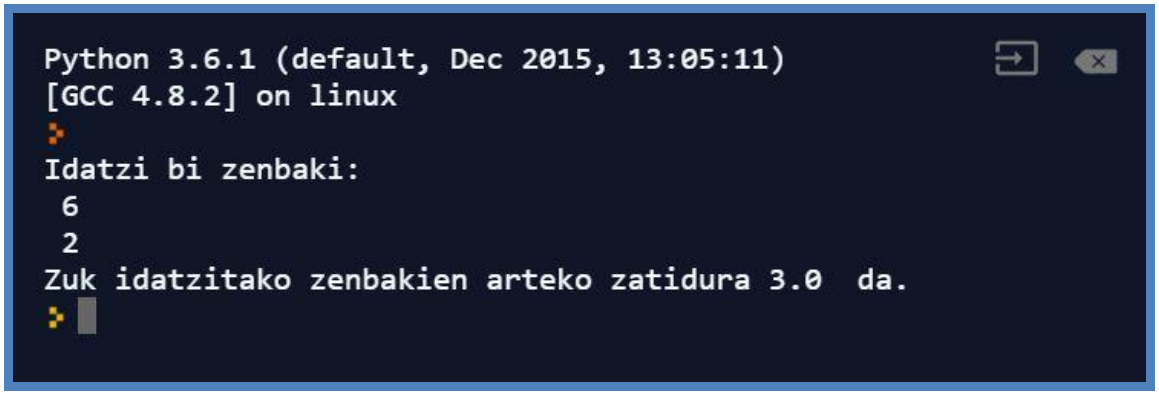
Ariketa ebazteko modu bat:



2.4. ariketa

Idatzi programa bat erabiltzaileari parametro gisa bi zenbaki eskatzeko eta haien arteko zatidura bistaratzeko.

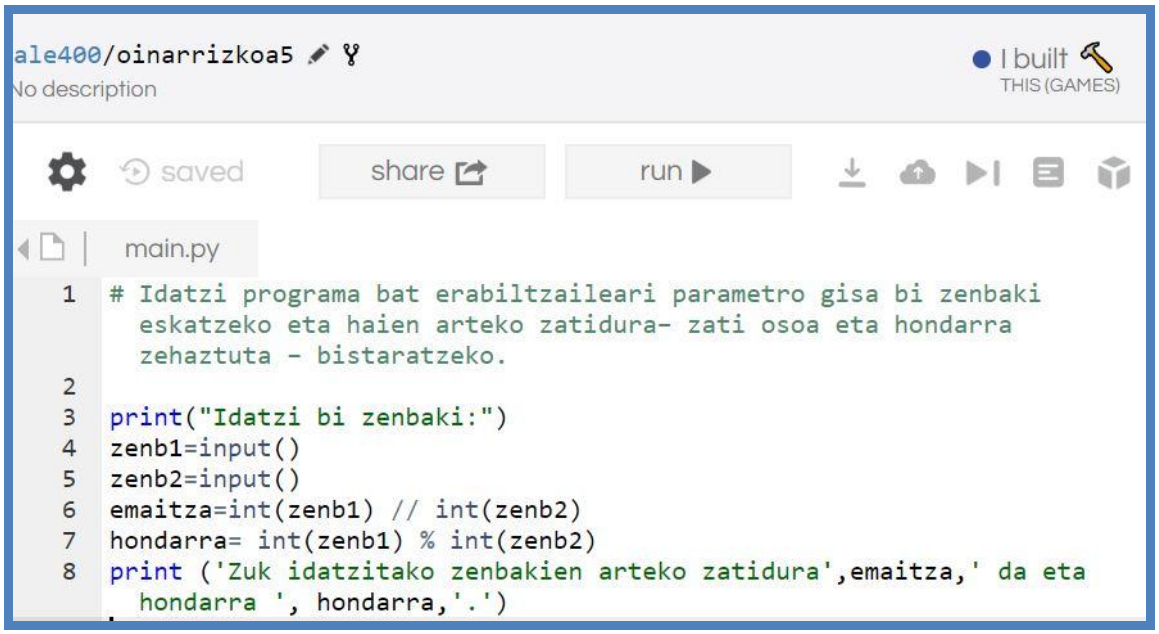
Ariketa ebazteko modu bat:

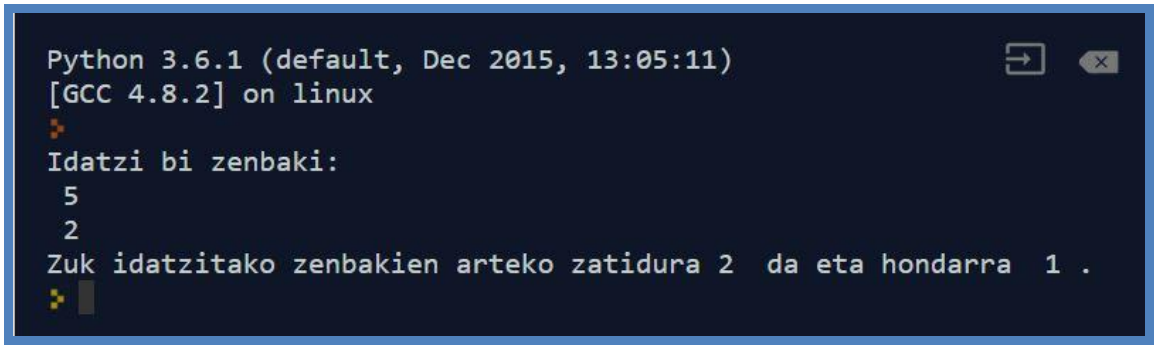


2.5. ariketa

Idatzi programa bat erabiltzaileari parametro gisa bi zenbaki eskatzeko eta haien arteko zatidura– zati osoa eta hondarra zehaztuta – bistaratzeko.

Ariketa ebazteko modu bat:

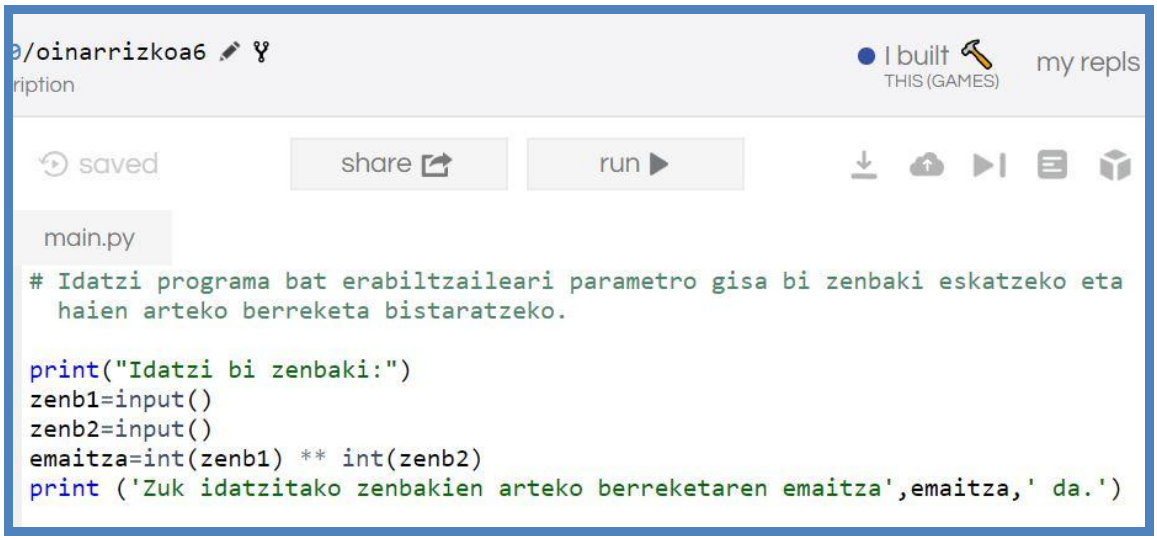


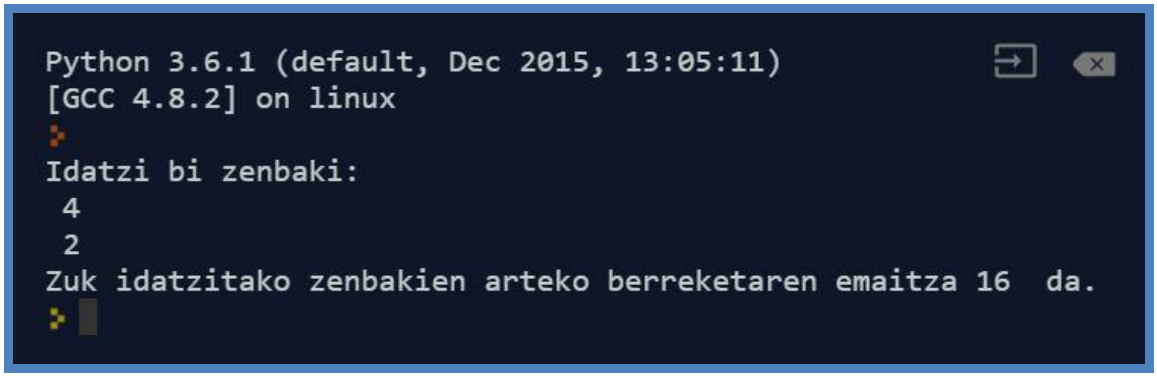


2.6. ariketa

Idatzi programa bat erabiltzaileari parametro gisa bi zenbaki eskatzeko eta haien arteko berreketa bistaratzeko.

Ariketa ebazteko modu bat:



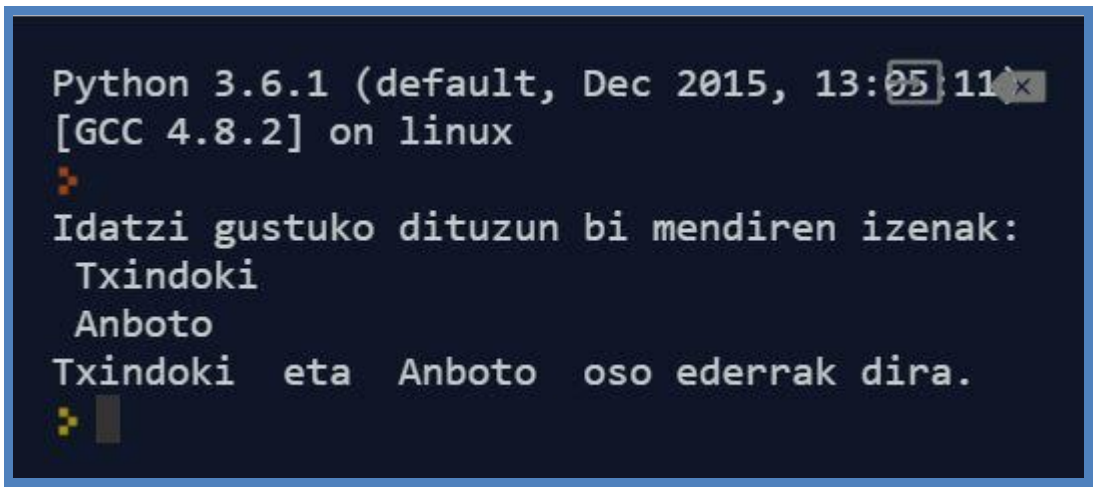


2.7. ariketa

Idatzi programa bat zeinak erabiltzaileari bi mendiren izenak parametro gisa idazteko eskatuko dion, eta, ondoren, bi mendien izenekin honako esaldi hau bistaratuko duen:

'1mendia eta 2mendia oso ederrak dira'.

Ariketa ebazteko modu bat:



2.8. ariketa

`p`, `k` eta `r` aldagai boolearrak dira. Hauek dira aldagaiei dagozkien balioak: `p=False`, `k=True` eta `r=False`. Balio horiek kontuan izanda, espresio hauetako zeinek itzuliko du `True`?

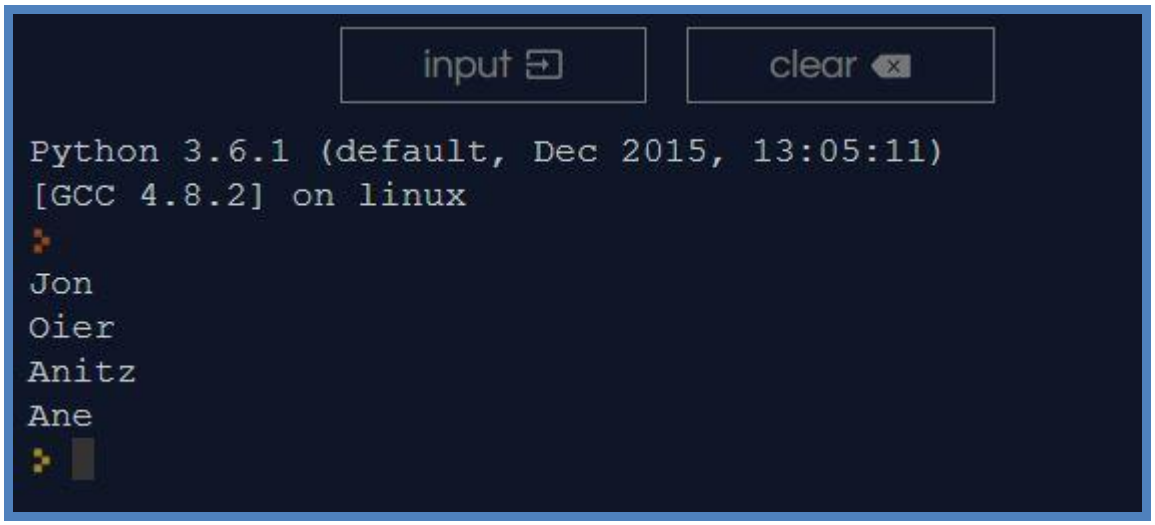
1. `p and not (k or not r)`
2. `p or not k and not r`
3. `k and not r!=p and not (not k)`
4. Aurrekoetatik batek ere ez du itzuliko `True` balioa.

3. ATALA: ZERRENDAK

3.1. ariketa

Sortu zerrenda bat honako izen hauekin: Jon, Oier eta Anitz. Ondoren, gehitu zerrenda horri Ane izena, `append` funtzioa erabiliz. Bukatzeko, bistaratu izenen zerrenda berria.

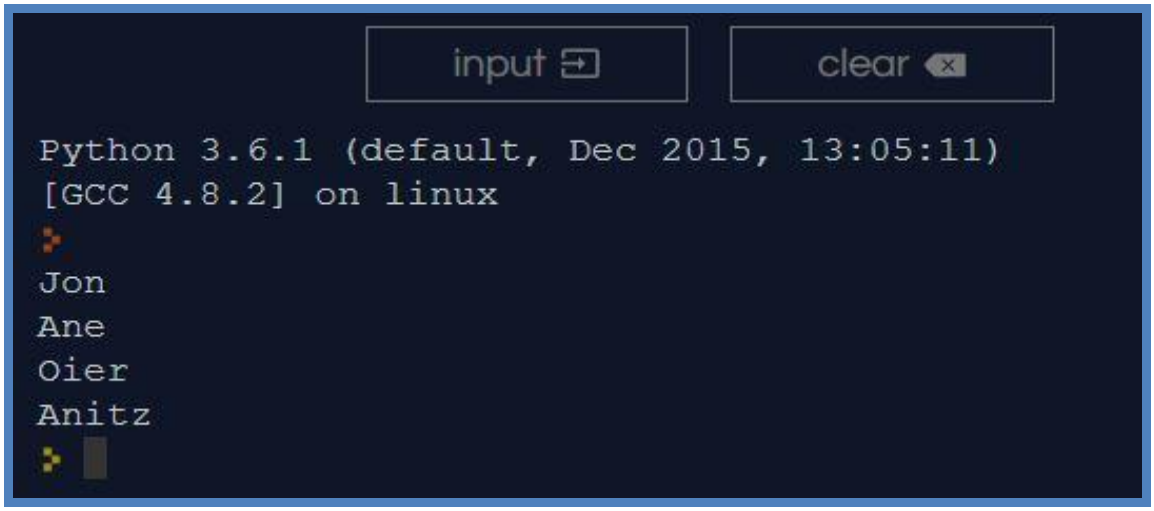
Ariketa ebazteko modu bat:



3.2. ariketa

Gehitu zerrendari beste izen bat 2. posizioan, `insert` funtzioa erabiliz. Ondoren, bistaratu izenen zerrenda berria. Gogoratu indizeak 0tik hasten direla kontatzen.

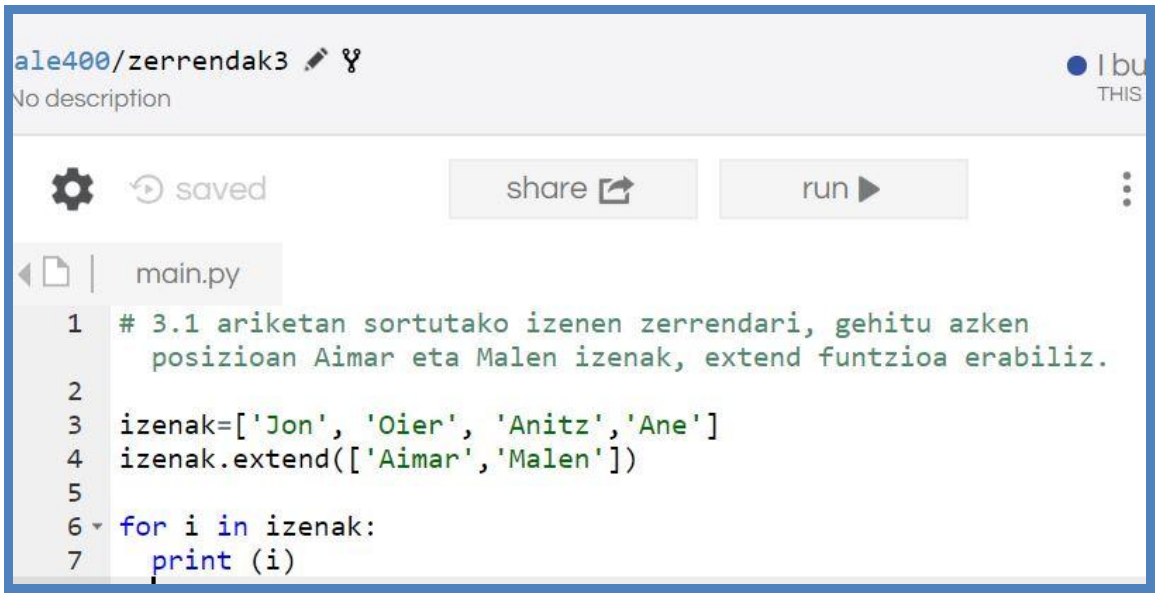
Ariketa ebazteko modu bat:

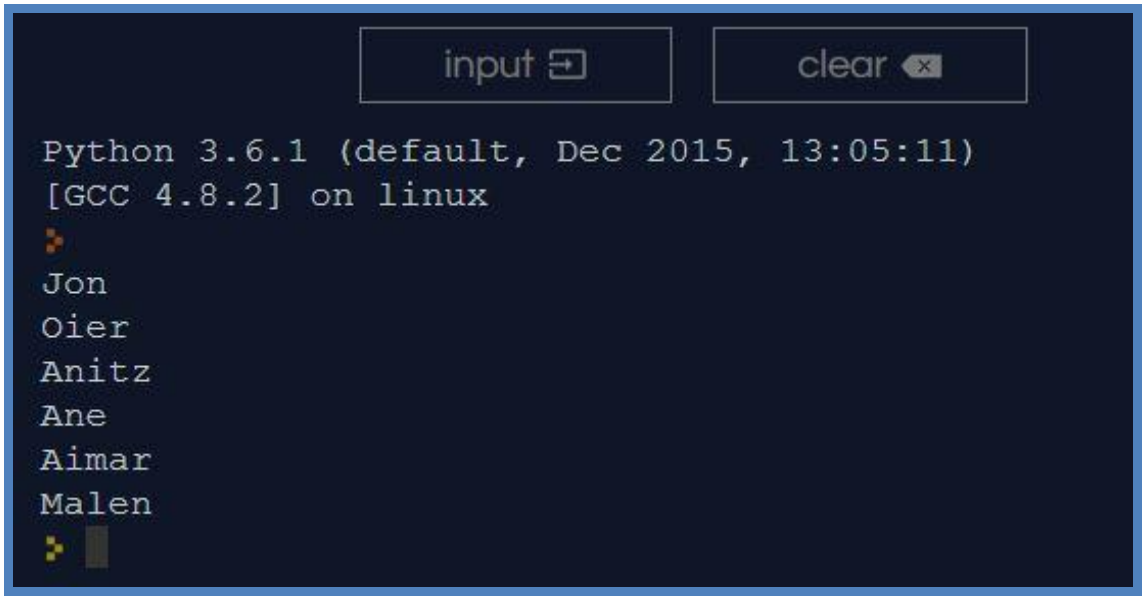


3.3. ariketa

3.1 ariketan sortutako izenen zerrendari, gehitu azken posizioan Aimar eta Malen izenak, `extend` funtzioa erabiliz.

Ariketa ebazteko modu bat:

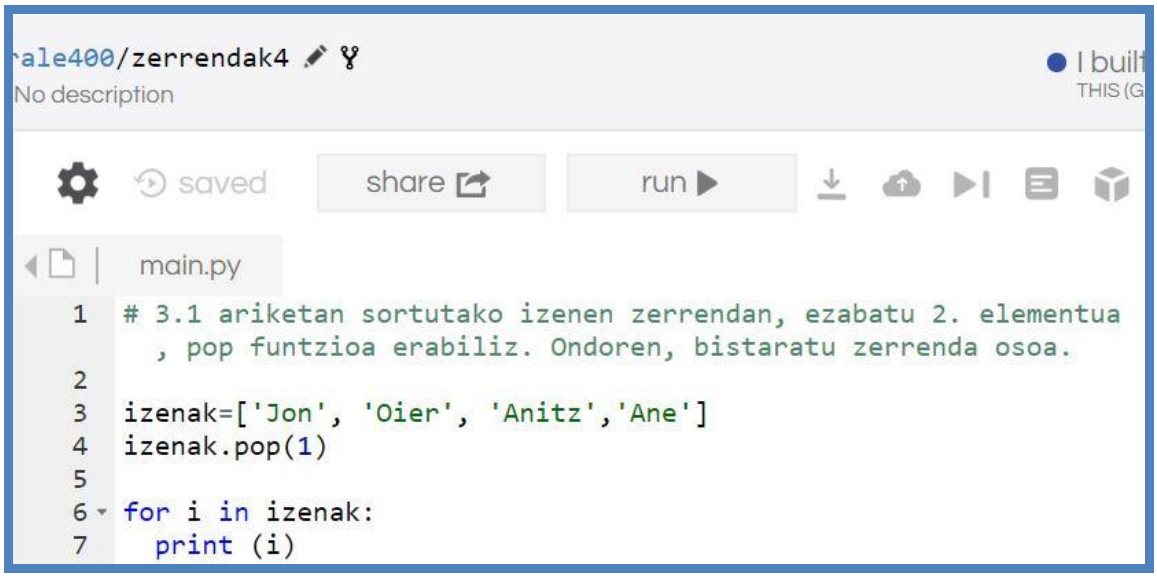


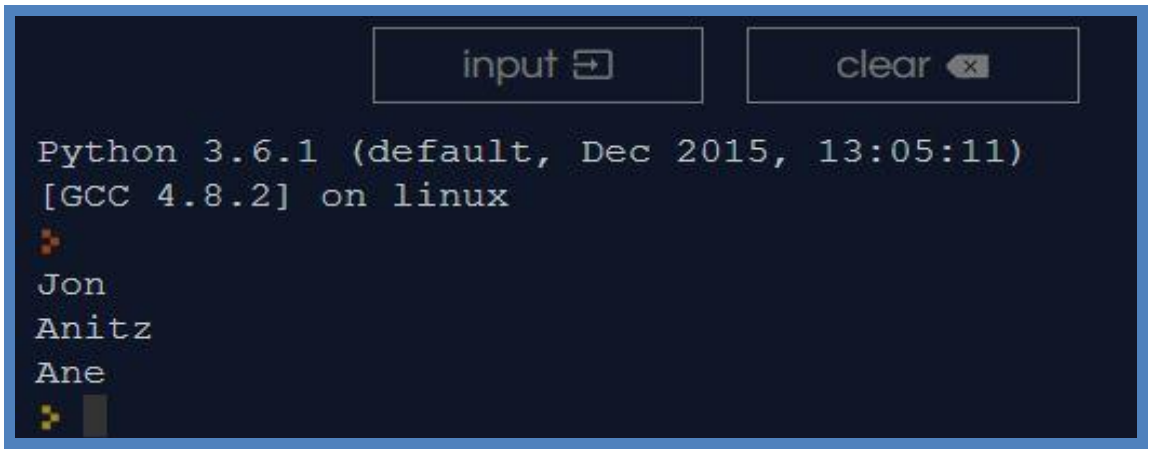


3.4. ariketa

3.1 ariketan sortutako izenen zerrendan, ezabatu 2. elementua, `pop` funtzioa erabiliz. Ondoren, bistaratu zerrenda osoa.

Ariketa ebazteko modu bat:

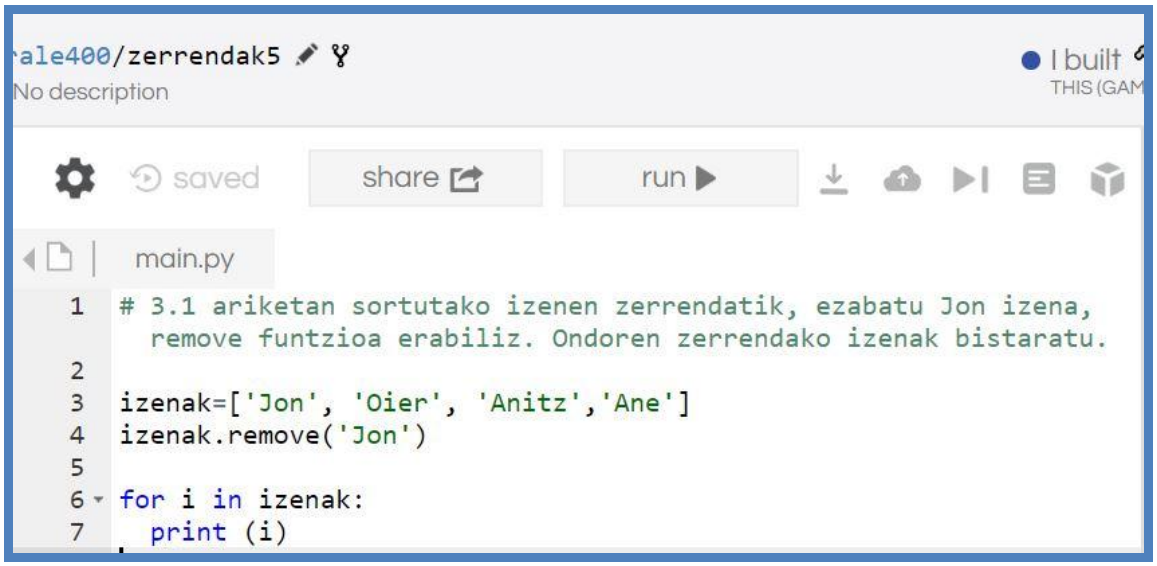


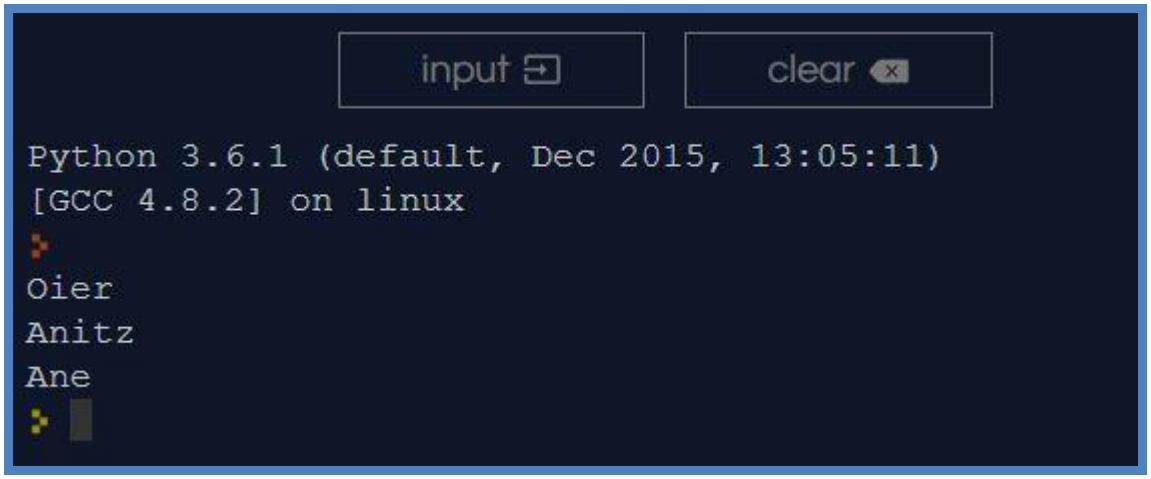


3.5. ariketa

3.1 ariketan sortutako izenen zerrendatik, ezabatu Jon izena, `remove` funtzioa erabiliz. Ondoren zerrendako izenak bistaratu.

Ariketa ebazteko modu bat:

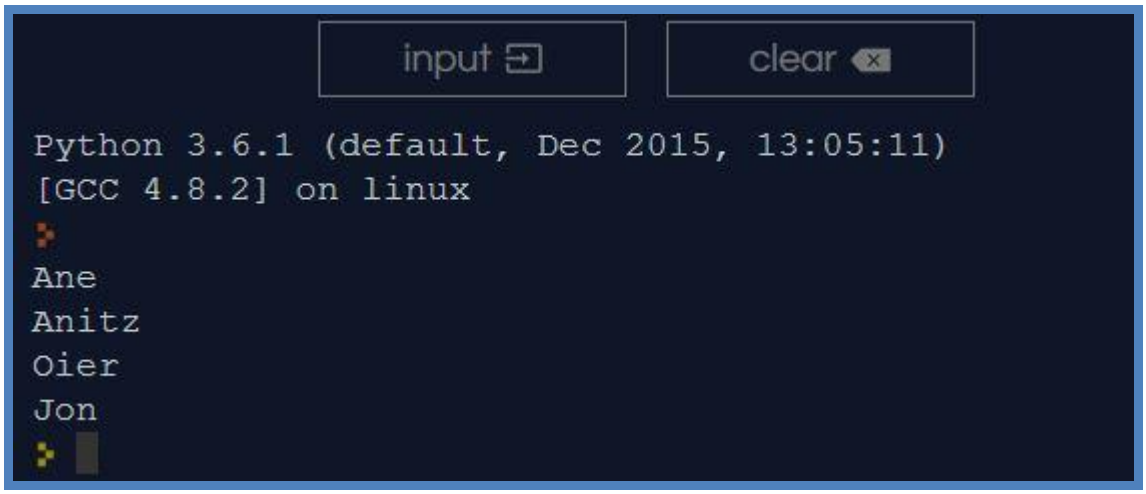




3.6. ariketa

Alderantzikatu 3.1 zerrendako elementuak , `reverse` funtzioa erabiliz. Ondoren, bistaratu zerrendako izenak.

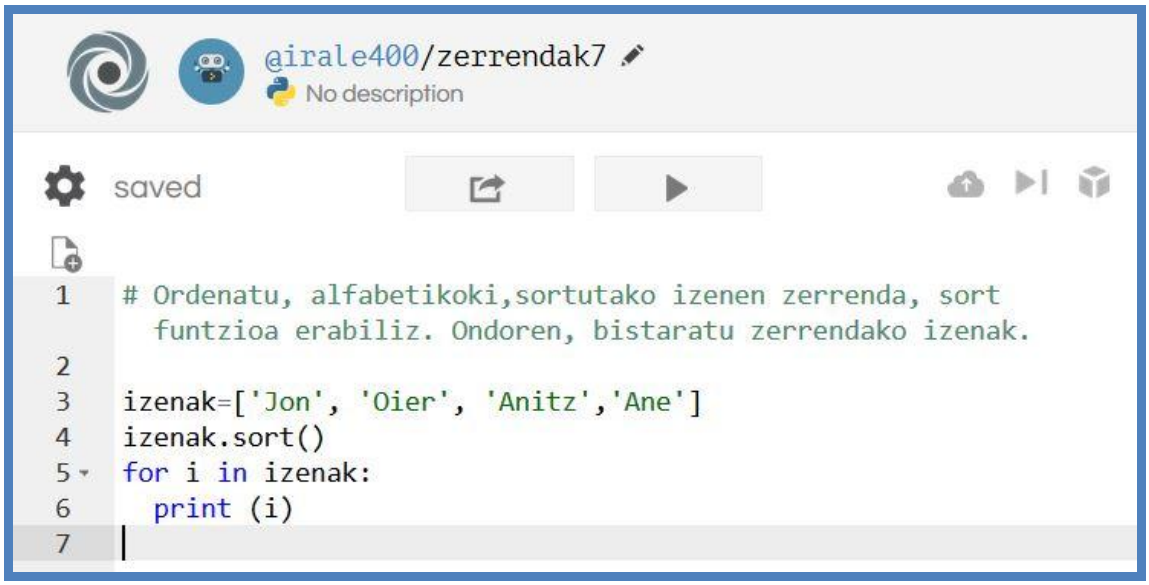
Ariketa ebazteko modu bat:

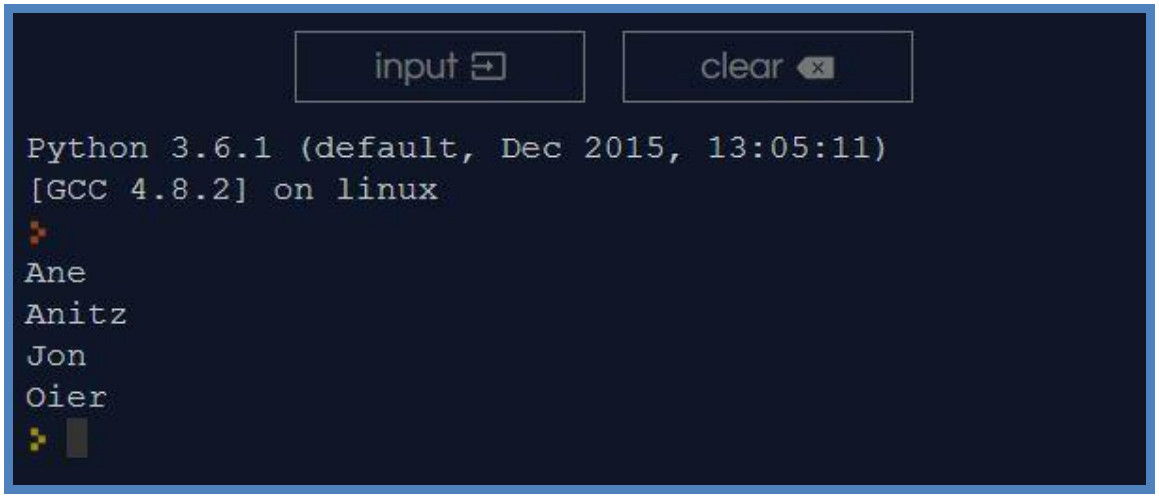


3.7. ariketa

Ordenatu, alfabetikoki, sortutako izenen zerrenda, `sort` funtzioa erabiliz. Ondoren, bistaratu zerrendako izenak.

Ariketa ebazteko modu bat:

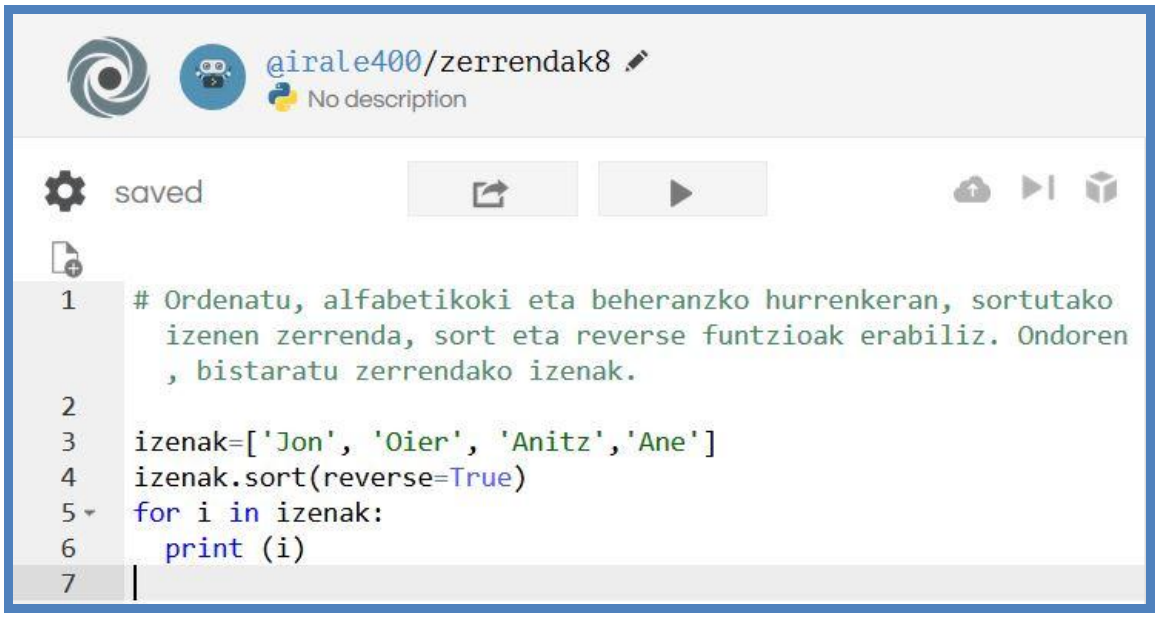


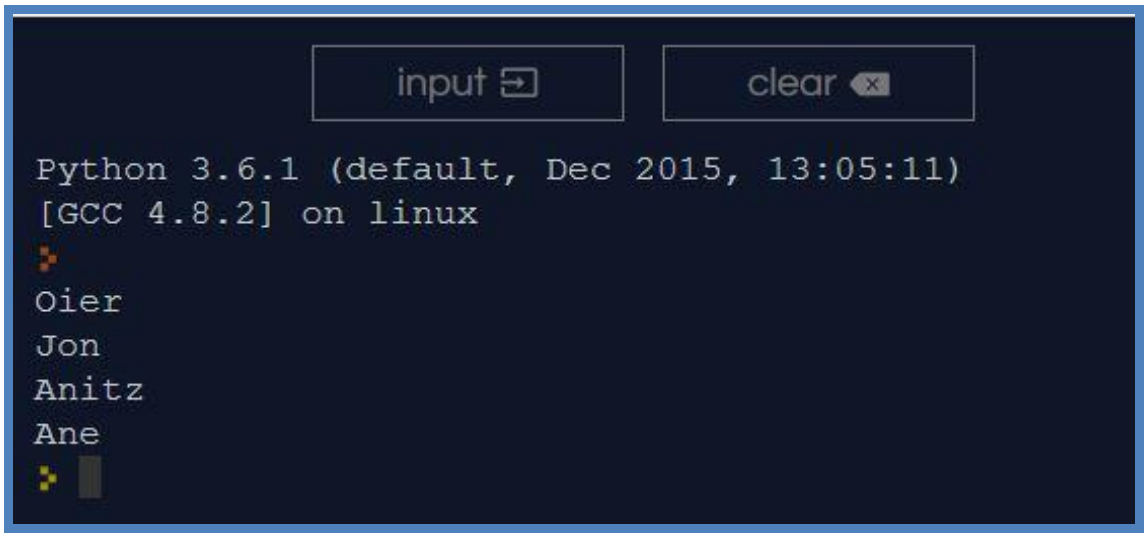


3.8. ariketa

Ordenatu, alfabetikoki eta beheranzko hurrenkeran, sortutako izenen zerrenda, `sort` eta `reverse` funtzioak erabiliz. Ondoren, bistaratu zerrendako izenak.

Ariketa ebazteko modu bat:

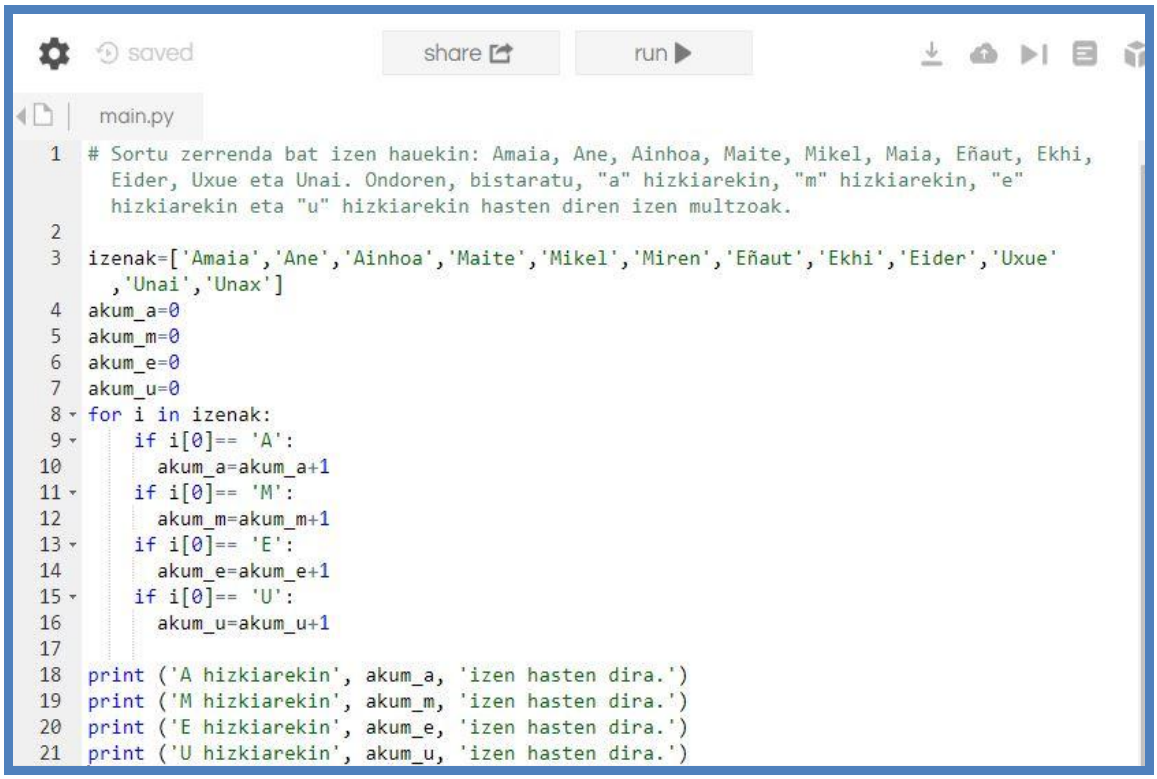


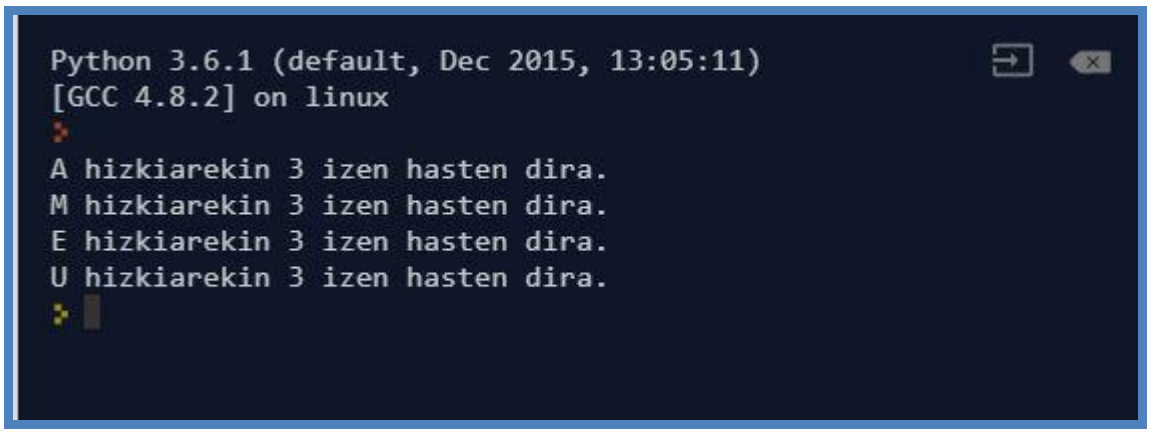


3.9. ariketa

Sortu zerrenda bat izen hauekin: Amaia, Ane, Ainhoa, Maite, Mikel, Maia, Eñaut, Ekhi, Eider, Uxue eta Unai. Ondoren, bistaratu, *a* hizkiarekin, *m* hizkiarekin, *e* hizkiarekin eta *u* hizkiarekin hasten diren izen multzoak.

Ariketa ebazteko modu bat:

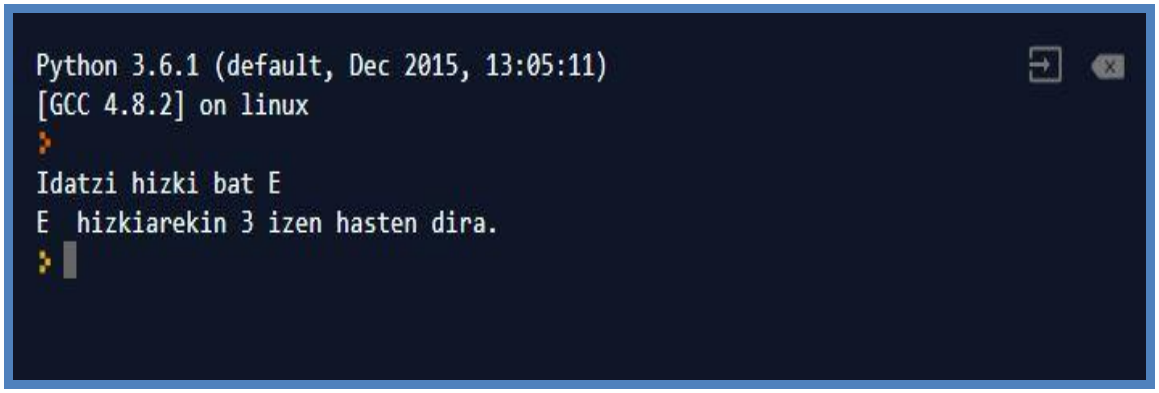




3.10. ariketa

Programatu 3.9. ariketan eskatzen den programa, baina, kasu honetan, erabiltzaileak idatzi beharko du parametro gisa programak zenbatu beharreko izenak zer hizkirekin hasten diren.

Ariketa ebazteko modu bat:

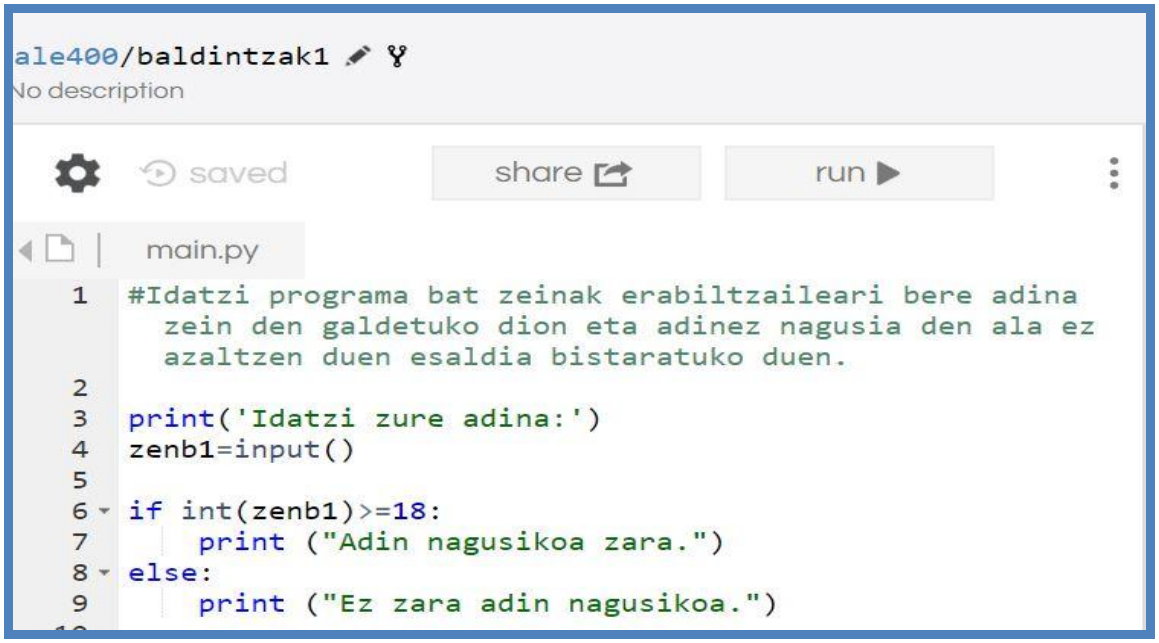


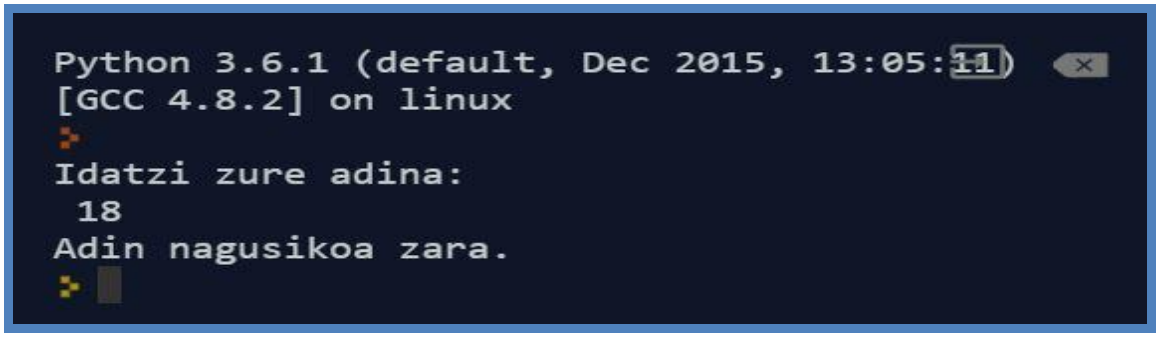
4. ATALA: BALDINTZAK

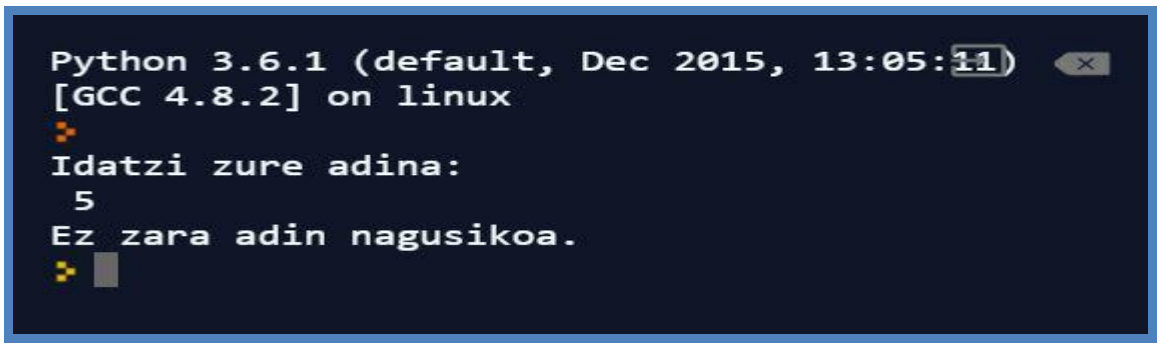
4.1. ariketa

Idatzi programa bat zeinak erabiltzaileari bere adina zein den galdetuko dion eta adinez nagusia den ala ez azaltzen duen esaldia bistaratuko duen.

Ariketa ebazteko modu bat:



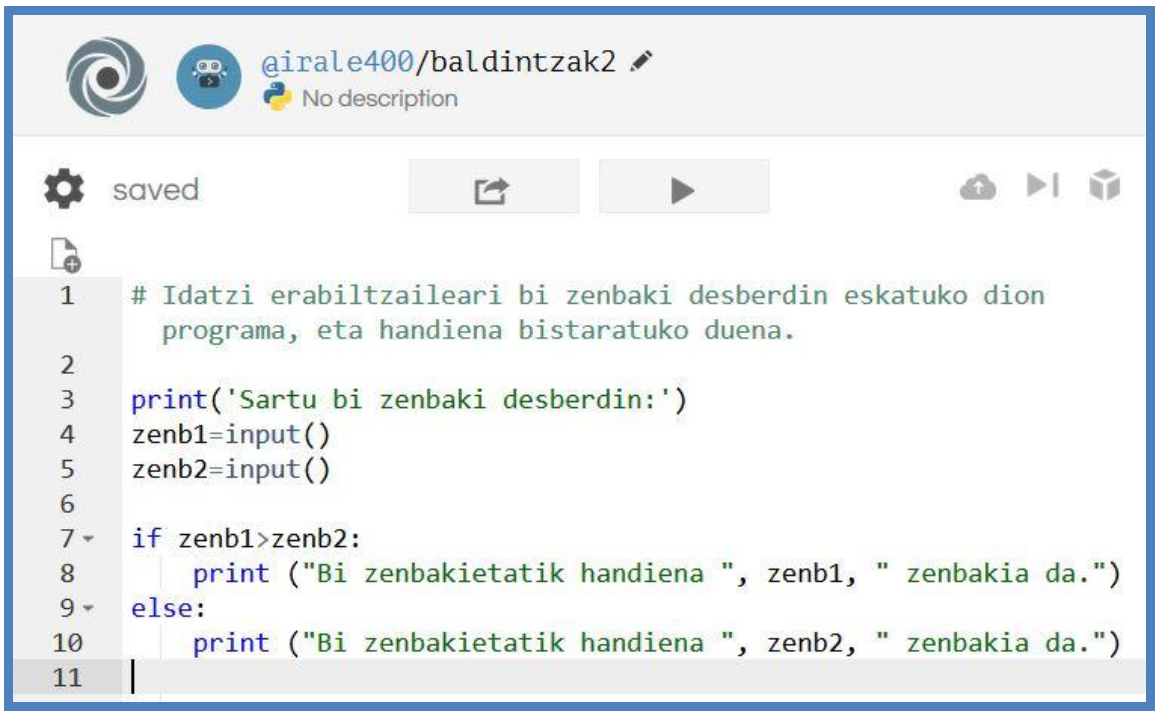


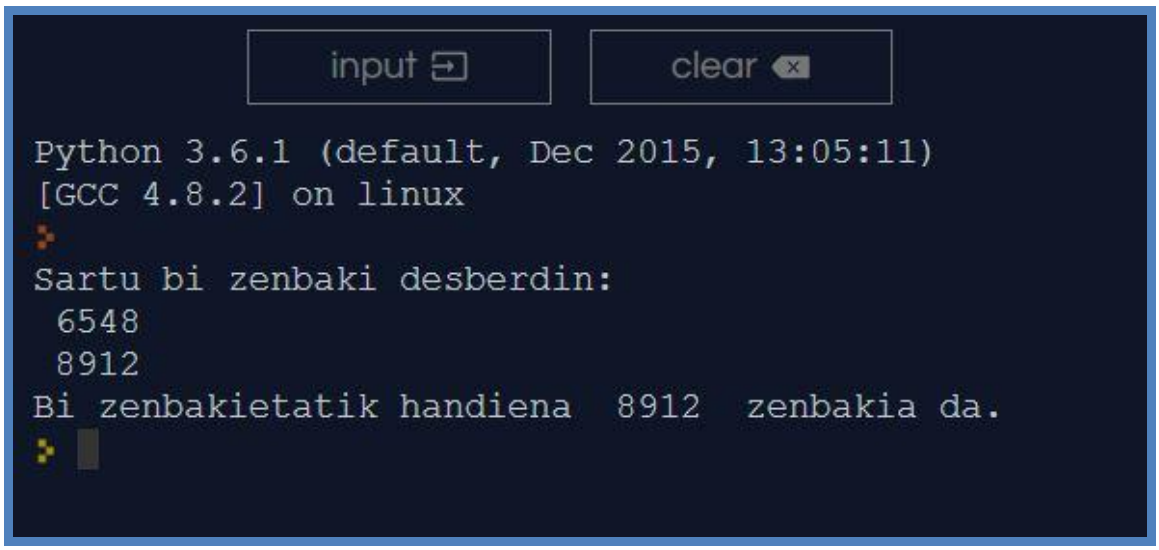


4.2. ariketa

Idatzi programa bat erabiltzaileari bi zenbaki desberdin eskatzeko eta handiena bistaratzeko.

Ariketa ebazteko modu bat:

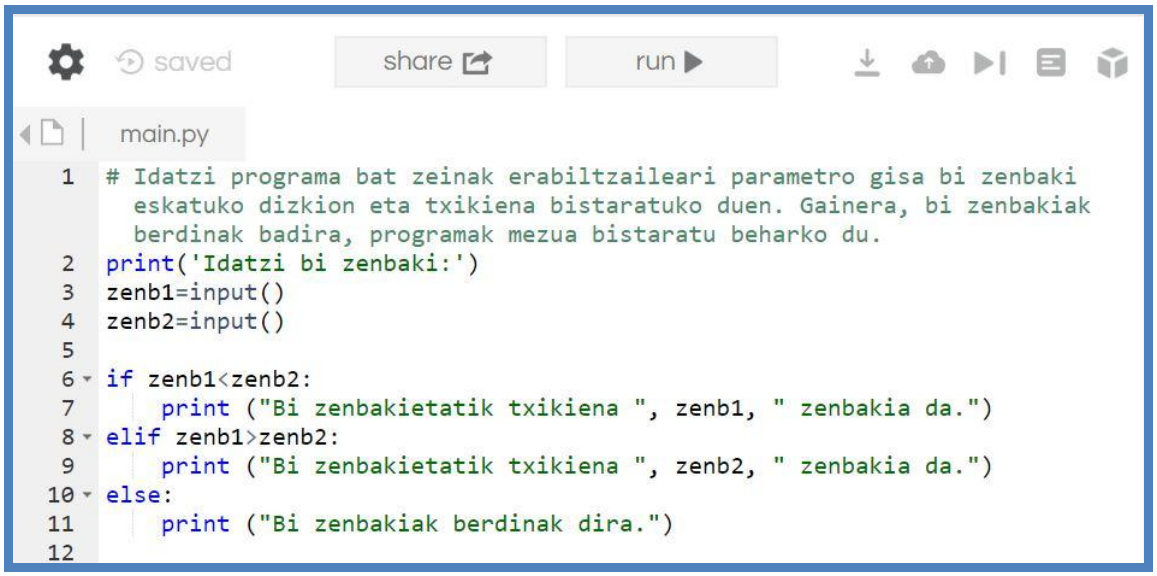


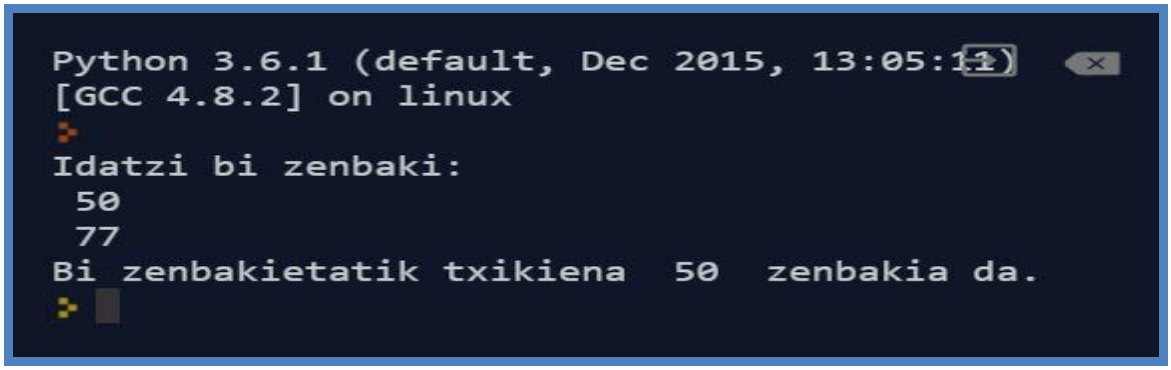


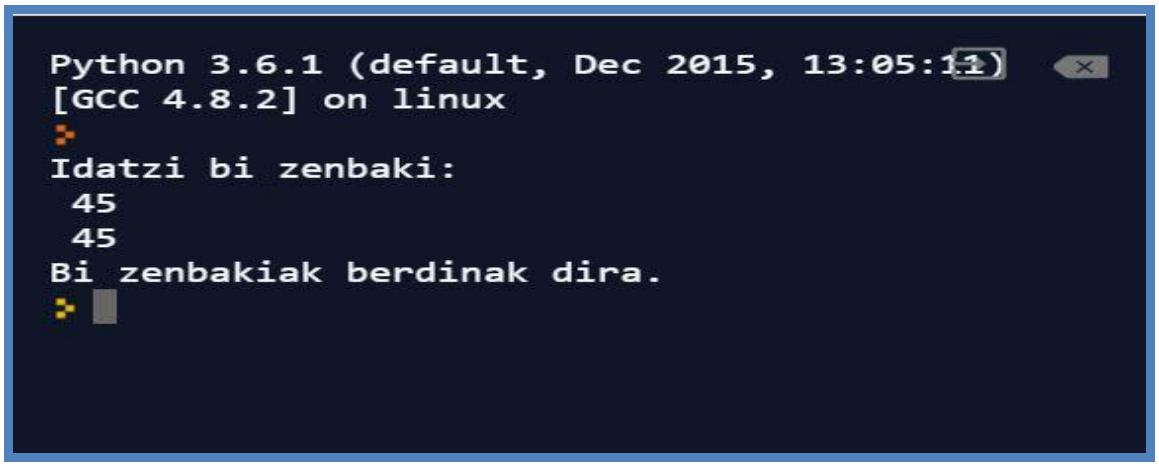
4.3. ariketa

Idatzi programa bat zeinak erabiltzaileari parametro gisa bi zenbaki eskatuko dizkion eta txikiena bistaratuko duen. Gainera, bi zenbakiak berdinak badira, programak mezua bistaratu beharko du.

Ariketa ebazteko modu bat:



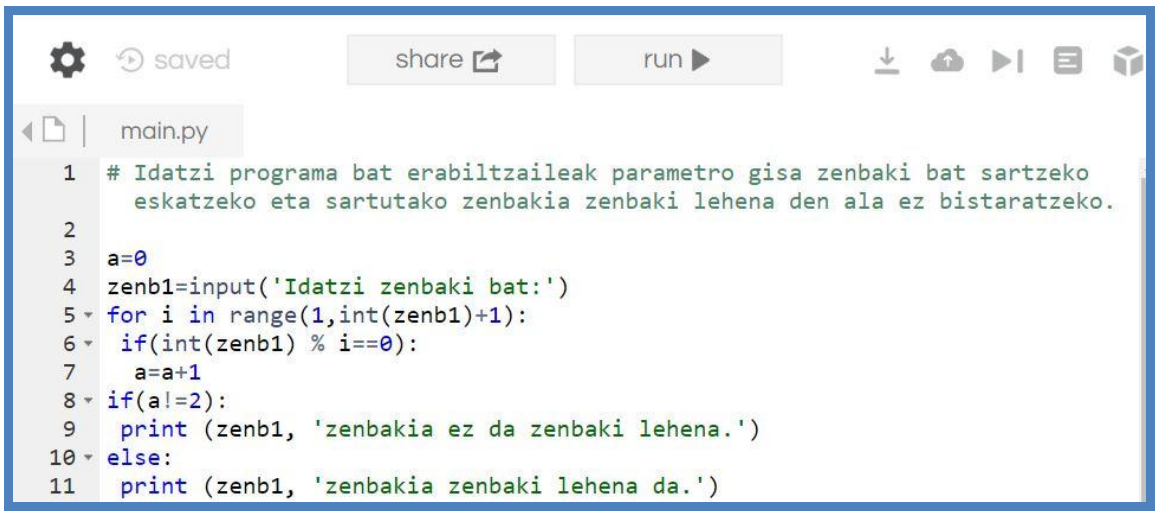


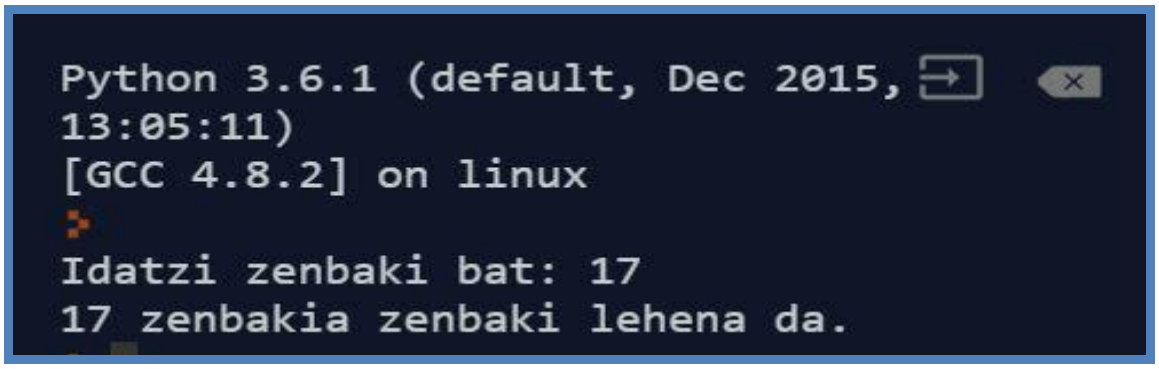


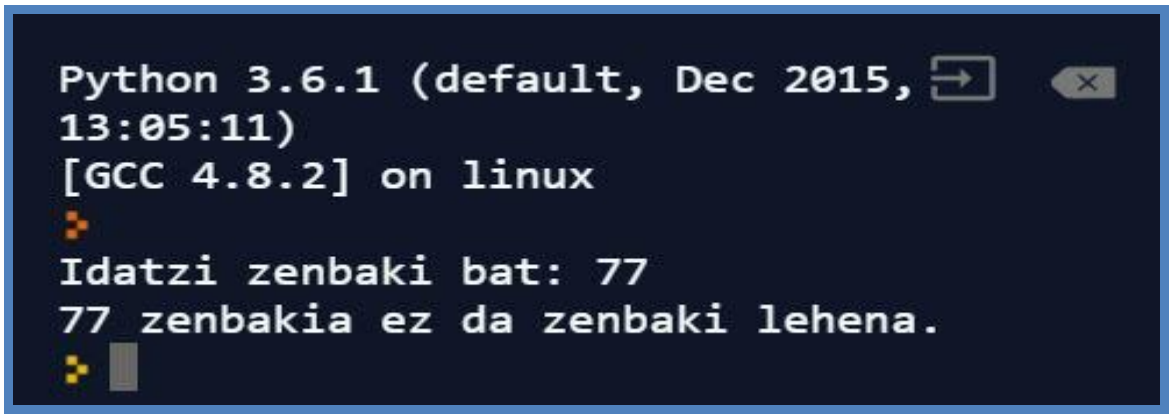
4.4. ariketa

Idatzi programa bat erabiltzaileak parametro gisa zenbaki bat sartzeko eskatzeko eta sartutako zenbakia zenbaki lehena den ala ez bistaratzeko.

Ariketa ebazteko modu bat:





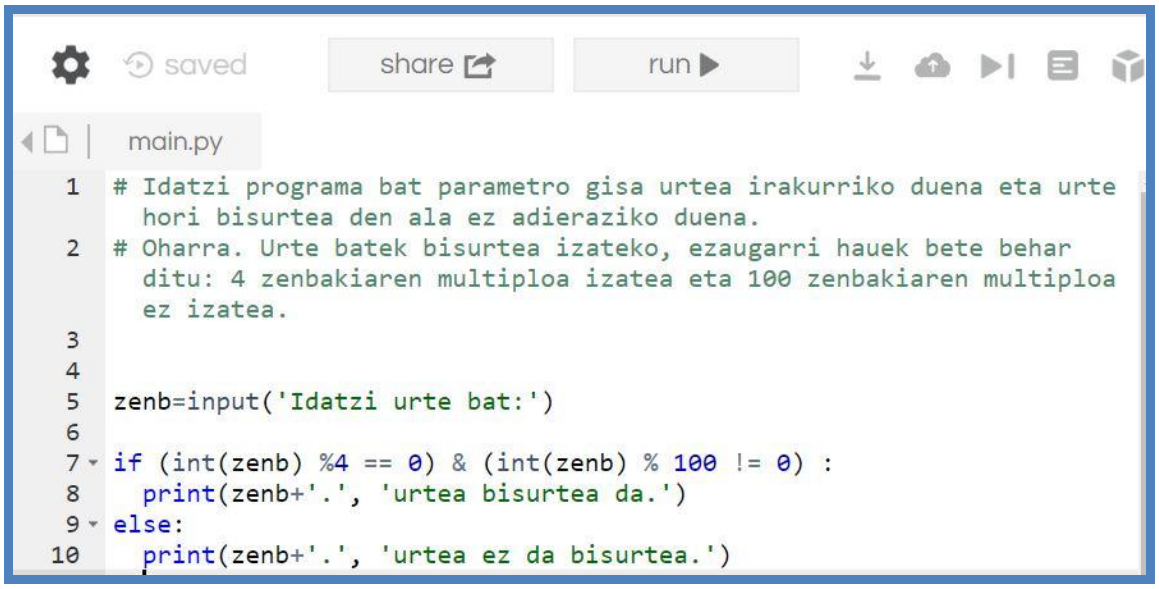


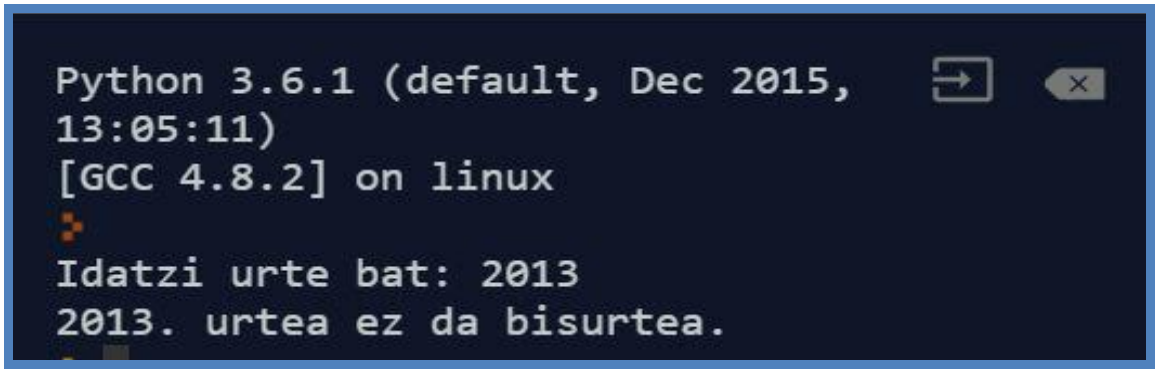
4.5. ariketa

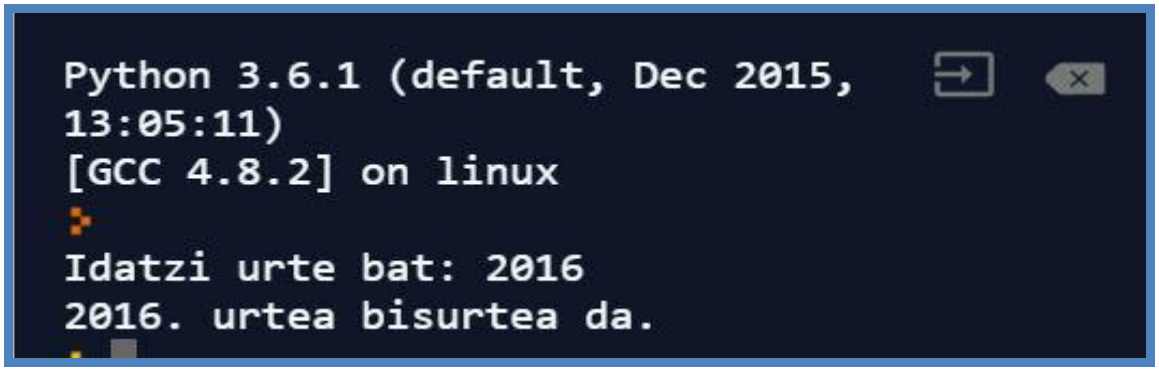
Idatzi programa bat parametro gisa urtea irakurriko duena eta urte hori bisurtea den ala ez adieraziko duena.

Oharra. Urte batek bisurtea izateko, ezaugarri hauek bete behar ditu: 4 zenbakiaren multiploa izatea eta 100 zenbakiaren multiploa ez izatea.

Ariketa ebazteko modu bat:





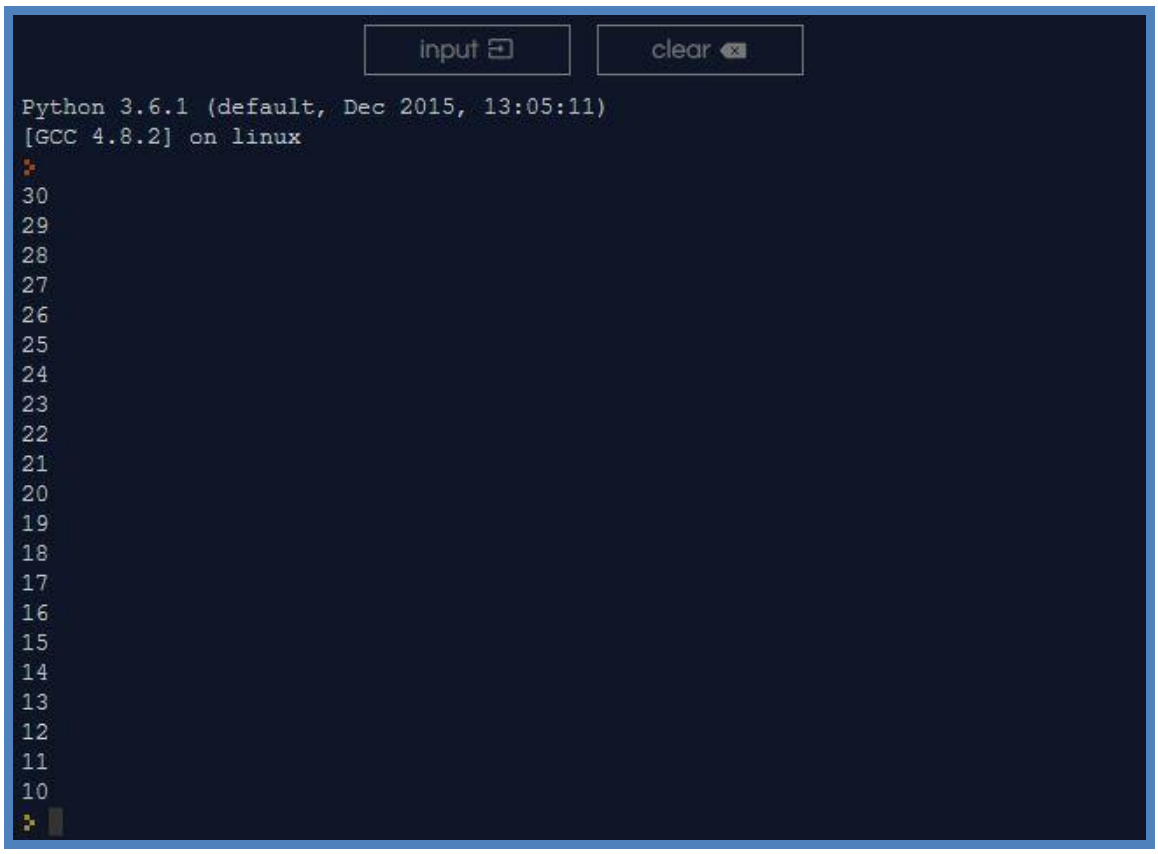


5. ATALA: BEGIZTAK

5.1. ariketa

Bistaratu, 30 eta 10 zenbakien artean dauden zenbaki oso guztiak, handitik txikira.

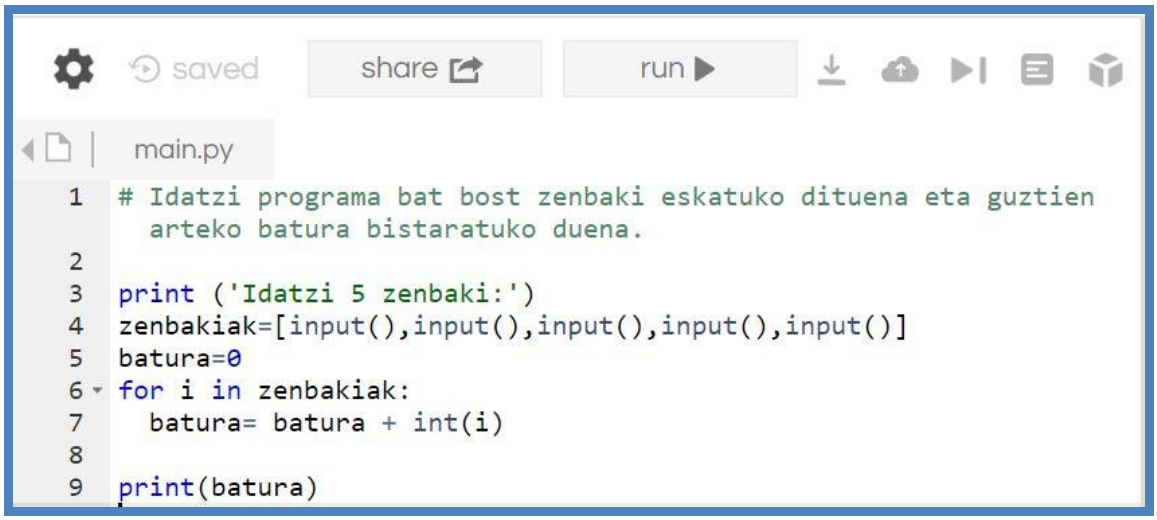
Ariketa ebazteko modu bat:

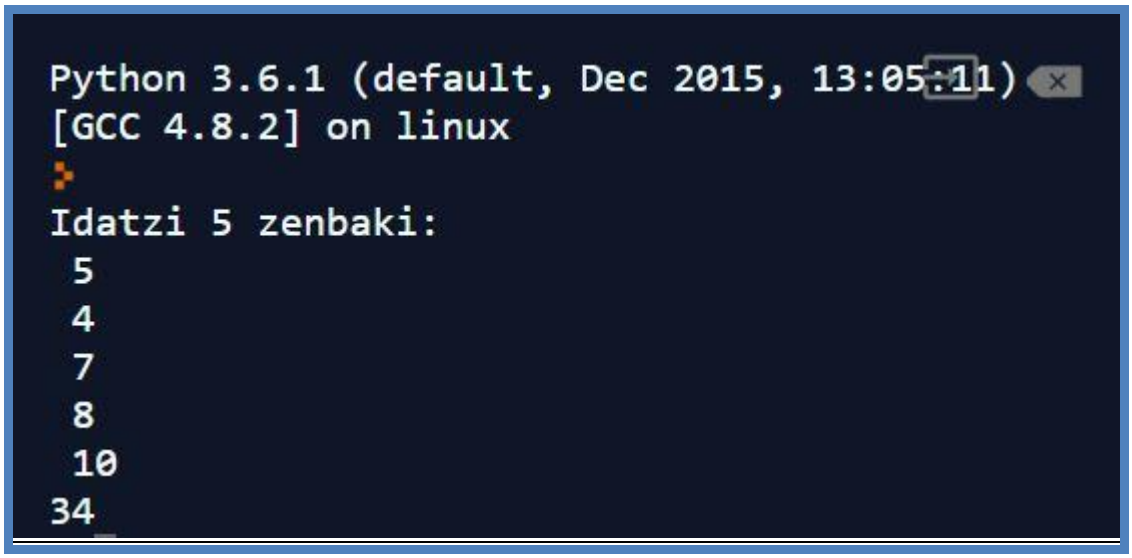


5.2. ariketa

Idatzi programa bat bost zenbaki eskatuko dituena eta guztien arteko batura bistaratuko duena.

Ariketa ebazteko modu bat:

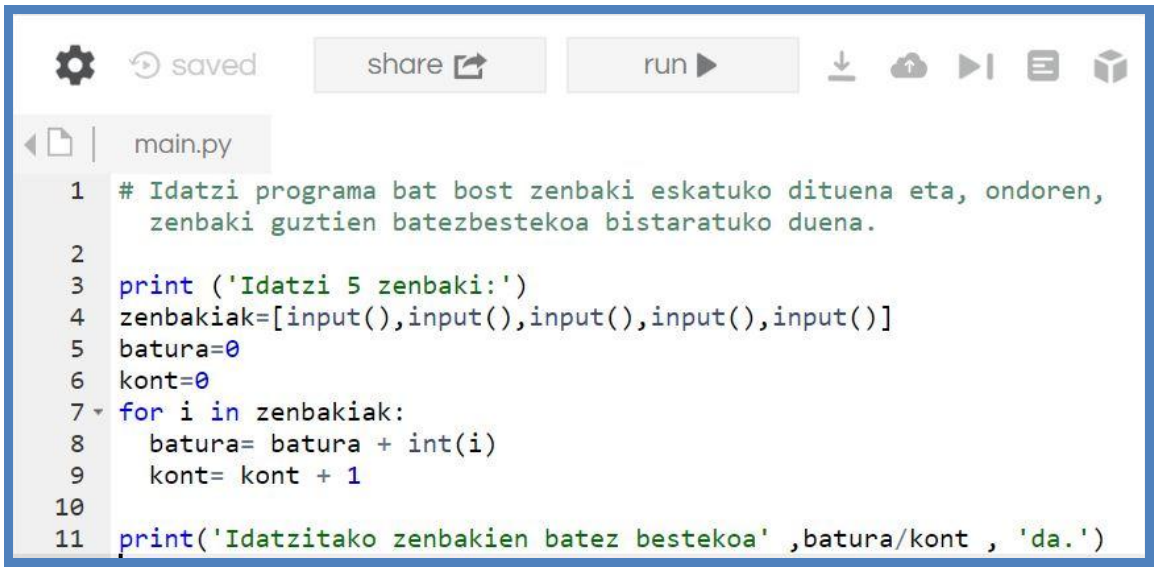


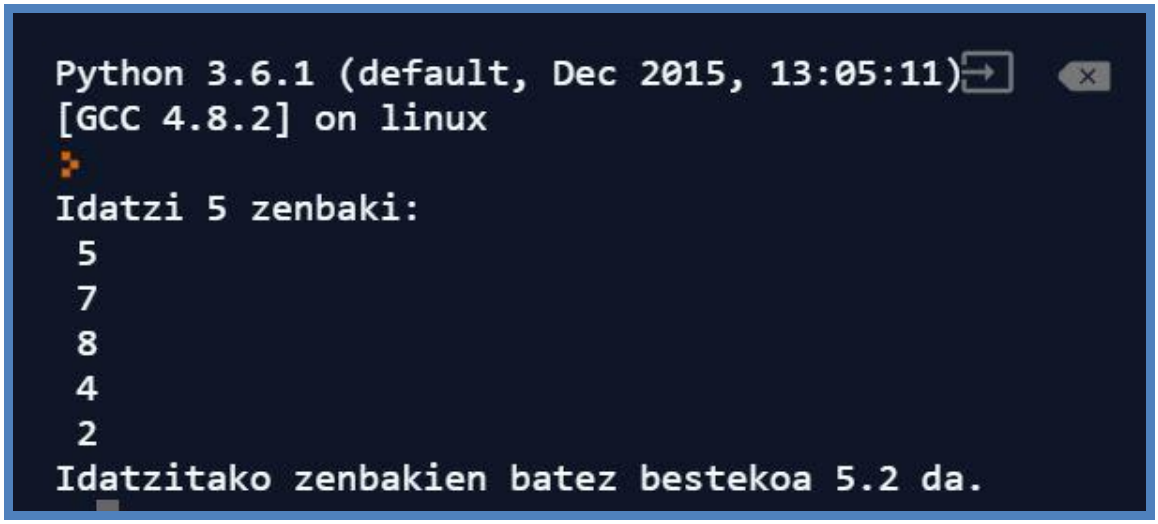


5.3. ariketa

Idatzi programa bat bost zenbaki eskatuko dituena eta, ondoren, zenbaki guztien batezbestekoa bistaratuko duena.

Ariketa ebazteko modu bat:

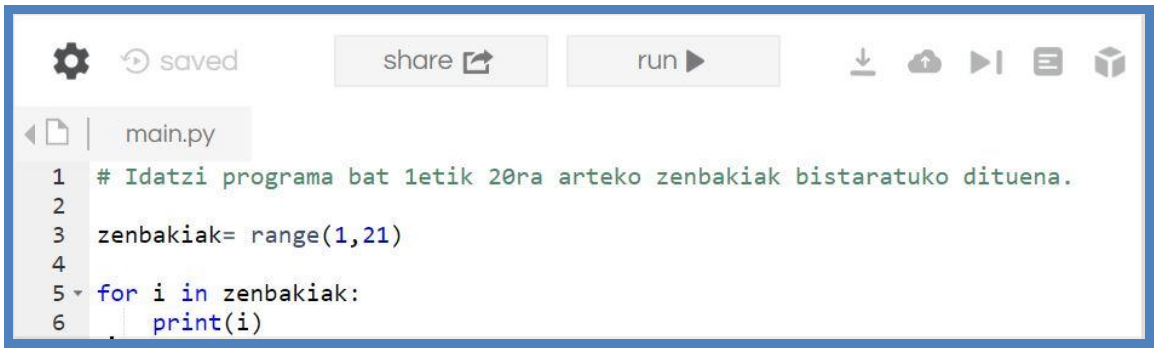


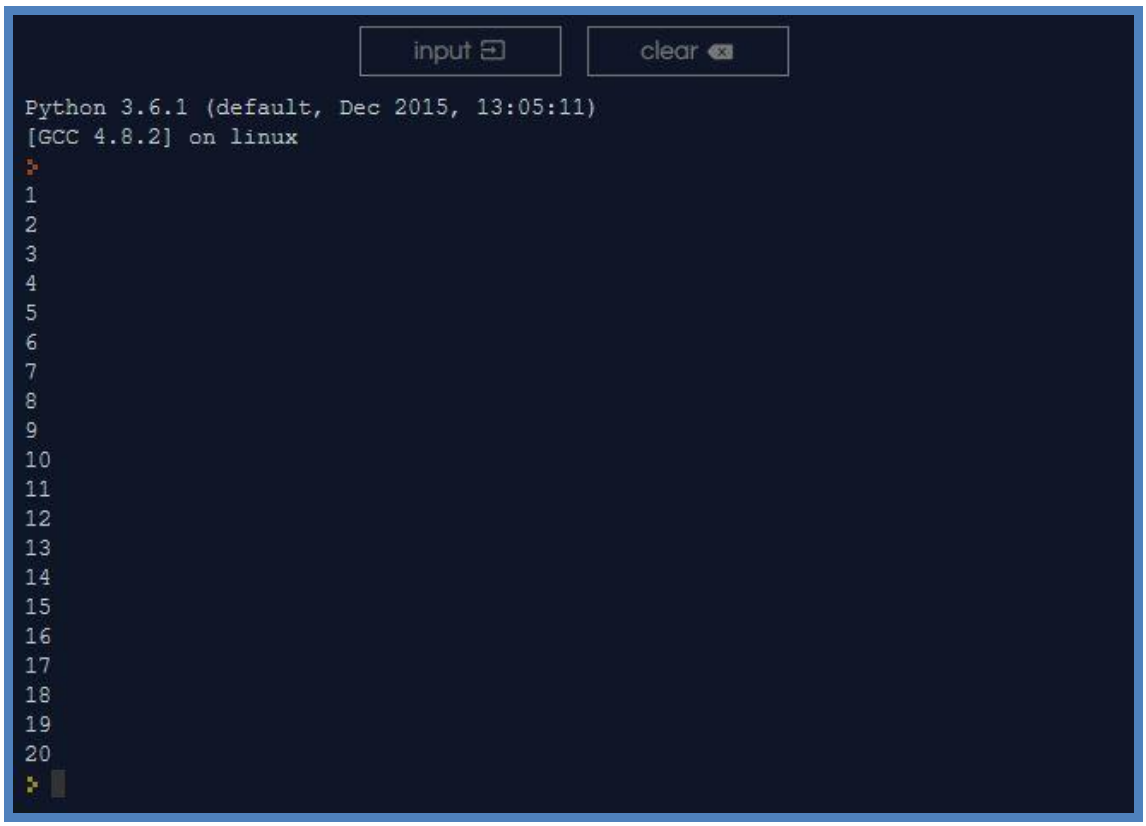


5.4. ariketa

Idatzi programa bat 1etik 20ra arteko zenbakiak bistaratuko dituena.

Ariketa ebazteko modu bat:

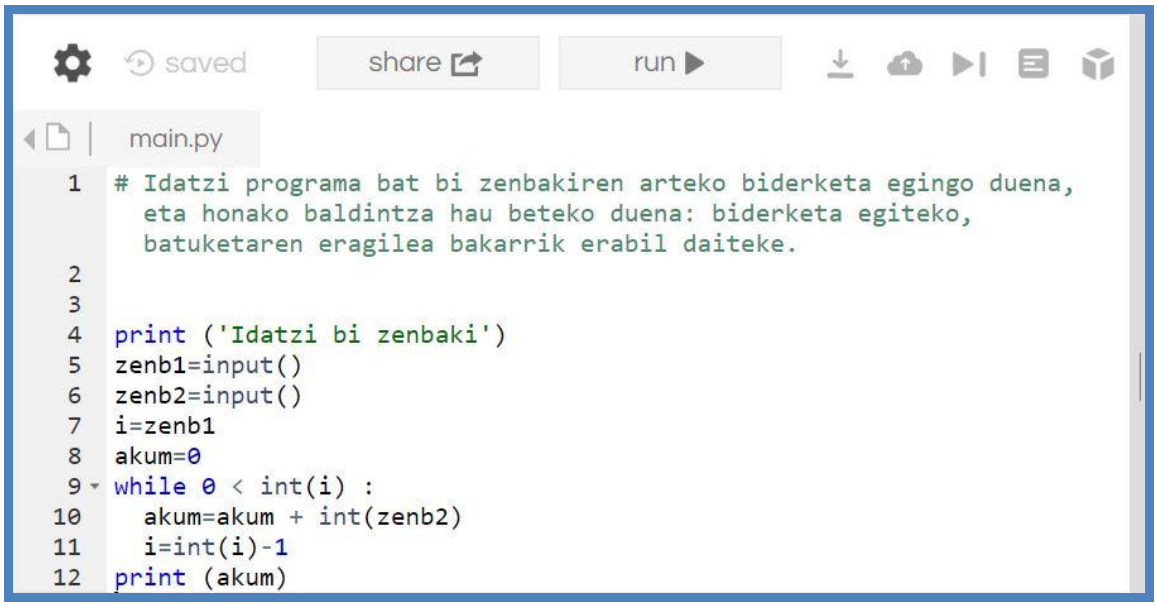


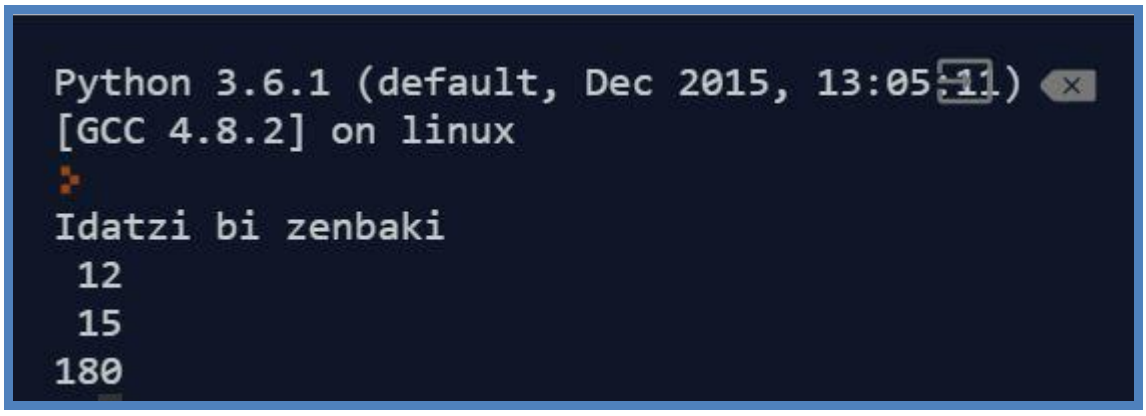


5.5. ariketa

Idatzi programa bat bi zenbakiren arteko biderketa egingo duena, eta honako baldintza hau beteko duena: biderketa egiteko, batuketaren eragilea bakarrik erabil daiteke.

Ariketa ebazteko modu bat:





5.6. ariketa

Idatzi programa bat parametro gisa bi zenbaki irakurriko dituena eta zenbaki hauen artean dauden zenbaki lehen guztiak bistaratuko dituena.

Ariketa ebazteko modu bat:

main.py

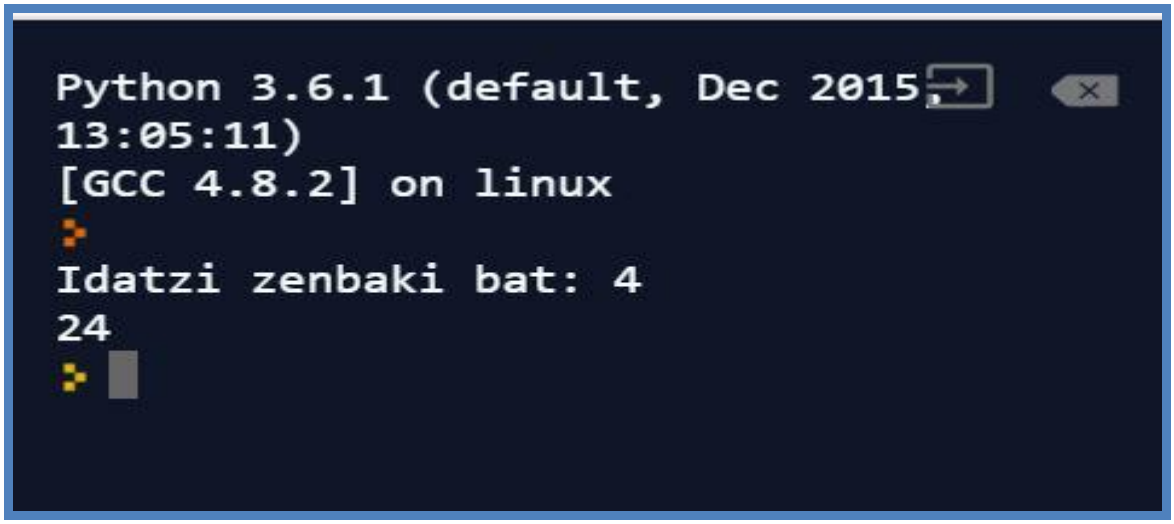
```
# Idatzi programa bat parametro gisa bi zenbaki irakurriko dituena eta zenbaki hauen artean
  dauden zenbaki lehen guztiak bistaratuko dituena.
zenb1=input('Idatzi zenbaki bat:')
zenb2=input('Idatz zenbaki bat:')
a=0

z=int(zenb1)
while (z>=int(zenb1)) and (z<=int(zenb2)) :
  for i in range(1,int(z)+1):
   if(int(z) % i==0):
     a=a+1
  if(a!=2):
   print (zenb1, 'zenbakia ez da zenbaki lehena.')
   a=0
  else:
   print (zenb1, 'zenbakia zenbaki lehena da.')
   a=0
  z=z+1
  zenb1=int(zenb1)+1
```

```
Python 3.6.1 (default, Dec 2015, 13:05:1
[GCC 4.8.2] on linux
Idatzi zenbaki bat: 10
Idatz zenbaki bat: 20
10 zenbakia ez da zenbaki lehena.
11 zenbakia zenbaki lehena da.
12 zenbakia ez da zenbaki lehena.
13 zenbakia zenbaki lehena da.
14 zenbakia ez da zenbaki lehena.
15 zenbakia ez da zenbaki lehena.
16 zenbakia ez da zenbaki lehena.
17 zenbakia zenbaki lehena da.
18 zenbakia ez da zenbaki lehena.
19 zenbakia zenbaki lehena da.
20 zenbakia ez da zenbaki lehena.
```

5.7. ariketa

Idatzi programa bat zenbaki baten faktoriala kalkulatzeko.

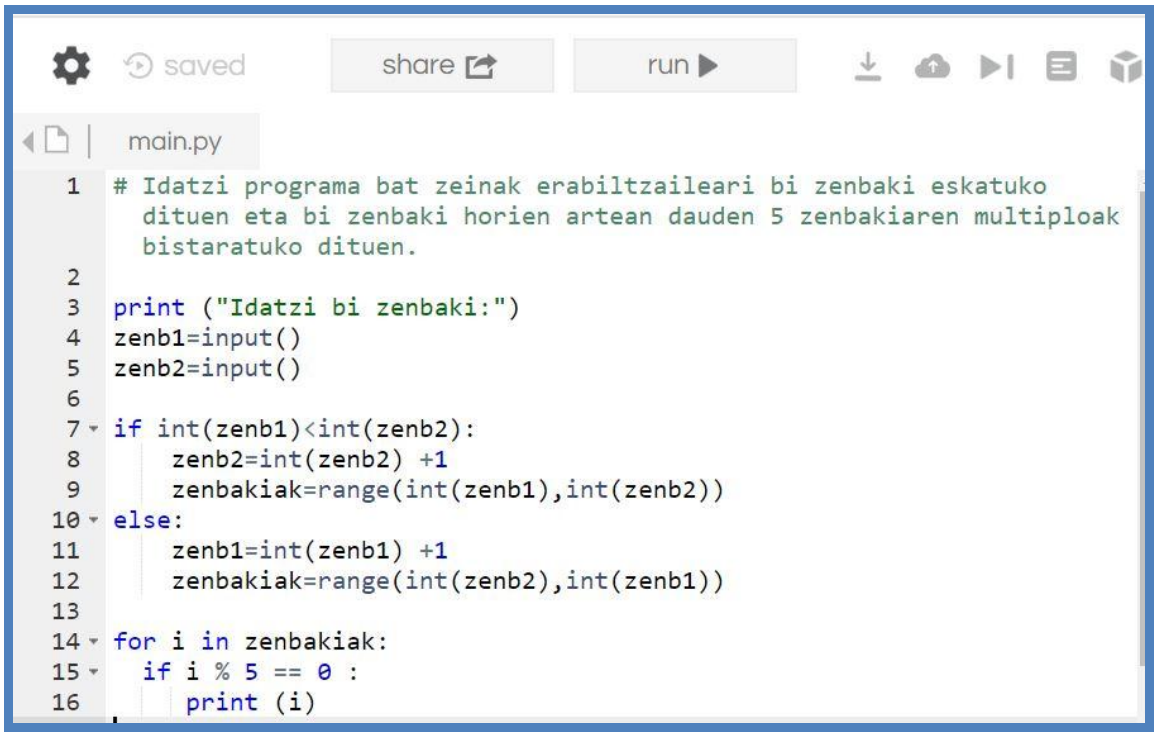
Ariketa ebazteko modu bat:

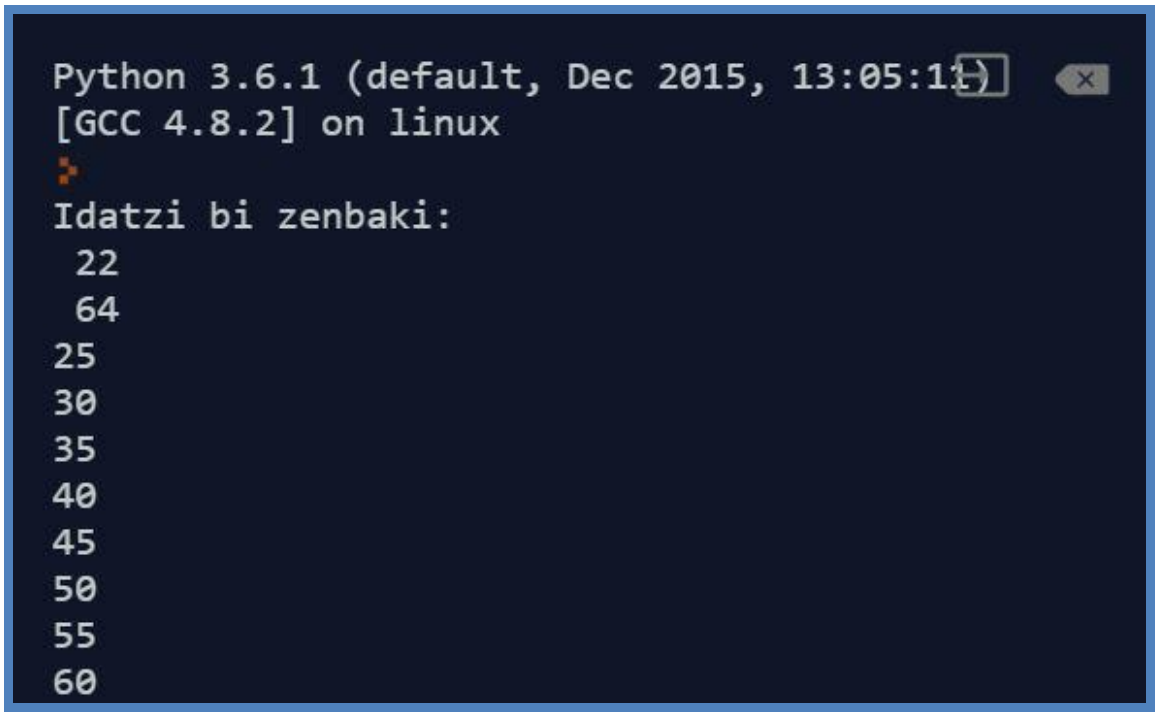


5.8. ariketa

Idatzi programa bat zeinak erabiltzaileari bi zenbaki eskatuko dituen eta bi zenbaki horien artean dauden 5 zenbakiaren multiploak bistaratuko dituen.

Ariketa ebazteko modu bat:





5.9. ariketa

Idatzi programa bat erabiltzaileak parametro gisa sartzen duen zenbakiaren karratua kalkulatzeko eta parametro gisa sartutako zenbakia 0 denean bukatzeko.

Ariketa ebazteko modu bat:

saved share run

main.py

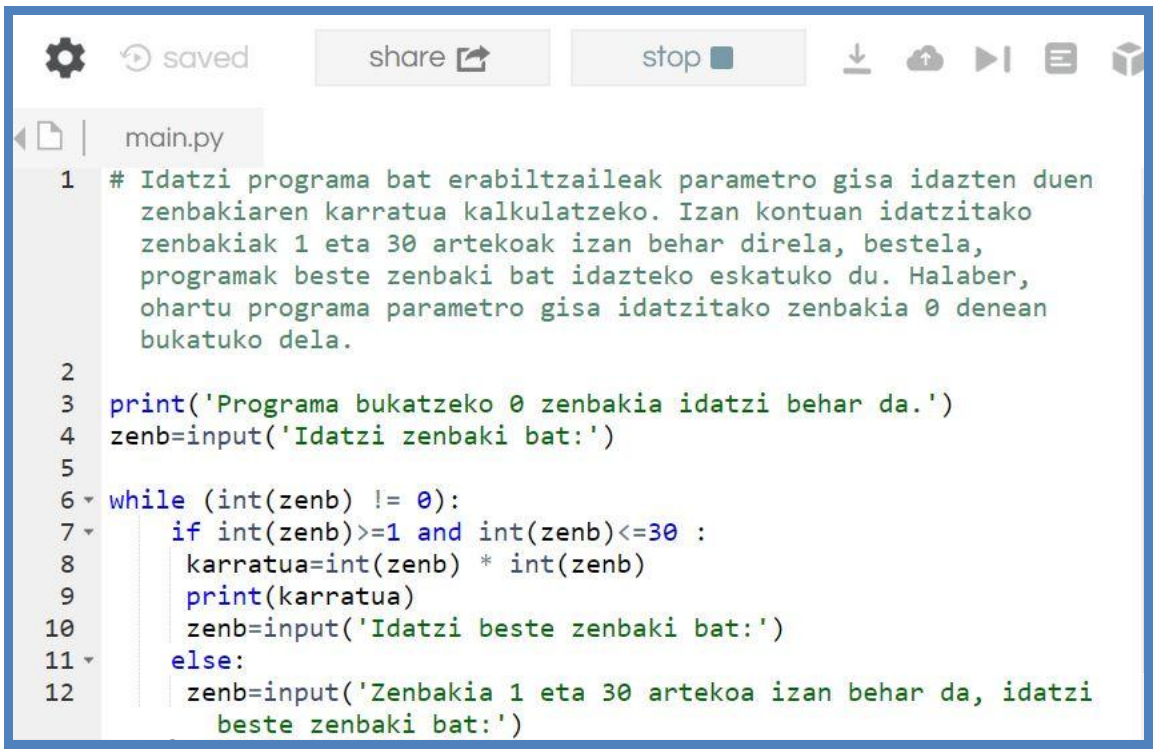
```
1  # Idatzi programa bat erabiltzaileak parametro gisa sartzen
     duen zenbakiaren karratua kalkulatzeko eta parametro gisa
     sartutako zenbakia 0 denean bukatzeko.
2
3  zenb=input('Idatzi zenbaki bat:')
4
5  while zenb != 0:
6      karratua=int(zenb) * int(zenb)
7
8      if int(zenb) != 0 :
9       print('Idatzitako zenbakiaren karratua',karratua , 'da.')
10      zenb=input('Idatzi zenbaki bat:')
11     else:
12       input('Programa bukatu da.')
```

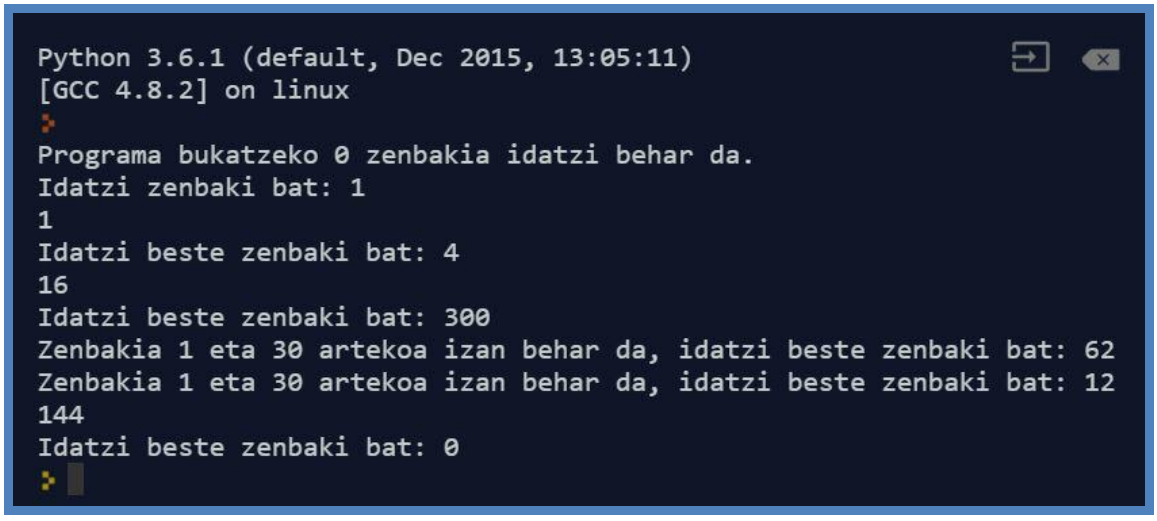
```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Idatzi zenbaki bat: 2
Idatzitako zenbakiaren karratua 4 da.
Idatzi zenbaki bat: 3
Idatzitako zenbakiaren karratua 9 da.
Idatzi zenbaki bat: 7
Idatzitako zenbakiaren karratua 49 da.
Idatzi zenbaki bat: 6
Idatzitako zenbakiaren karratua 36 da.
Idatzi zenbaki bat: 10
Idatzitako zenbakiaren karratua 100 da.
Idatzi zenbaki bat: 0
Programa bukatu da.
```

5.10. ariketa

Idatzi programa bat erabiltzaileak parametro gisa idazten duen zenbakiaren karratua kalkulatzeko. Izan kontuan idatzitako zenbakiak 1 eta 30 artekoak izan behar direla, bestela, programak beste zenbaki bat idazteko eskatuko du. Halaber, ohartu programa parametro gisa idatzitako zenbakia 0 denean bukatuko dela.

Ariketa ebazteko modu bat:





6. ATALA: FUNTZIOAK

6.1 ariketa

Definitu `max_hiru` izeneko funtzioa, zeinak parametro gisa hiru zenbaki hartzen dituen eta handiena itzultzen duen.

```
# Definitu max_hiru izeneko funtzioa, zeinak parametro gisa hiru zenbaki hartzen dituen
  eta handiena itzultzen duen.

def max_hiru (zenb1, zenb2, zenb3):
  handiena=0
  txikiena=0
  if (zenb1>zenb2) and (zenb1>zenb3):
    handiena=zenb1
  elif (zenb2>zenb1) and (zenb2>zenb3):
      handiena=zenb2
  else:
     handiena=zenb3
  print('Idatzitako hiru zenbakietatik handiena', handiena, 'da.')

print('Idatzi hiru zenbaki:')
zenb1=input()
zenb2=input()
zenb3=input()
max_hiru(zenb1,zenb2,zenb3)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Idatzi hiru zenbaki:
 52
 65
 42
Idatzitako hiru zenbakietatik handiena 65 da.
>
```

6.2 ariketa

Idatzi funtzio bat hizki bat parametro gisa hartuko duena; gainera, bokala bada, `True` balio itzuliko duena; baina, bokala ez bada, `False` balioa itzuliko duena.

```
#Idatzi funtzio bat hizki bat parametro gisa hartuko duena; gainera, bokala bada,
  True balio itzuliko duena; baina, bokala ez bada, False balioa itzuliko duena.

def f_bokala (hizkia):

  if (hizkia=='A')or(hizkia=='E')or(hizkia=='I')or(hizkia=='O')or(hizkia=='U'):
    return True
  elif (hizkia=='a')or(hizkia=='e')or(hizkia=='i')or(hizkia=='o')or(hizkia=='u'):
    return True
  else:
   return False

hizkia=input('Idatzi hizki bat:')
if f_bokala(hizkia)==True:
  print('Idatzitako hizkia bokala da.')
else:
  print('Idatzitako hizkia ez da bokala.')
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Idatzi hizki bat: k
Idatzitako hizkia ez da bokala.
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Idatzi hizki bat: a
Idatzitako hizkia bokala da.
```

6.3 ariketa

Definitu `palindromoa` izeneko funtzioa. Esaterako, `palindromoa('amama')` aginduak `True` balioa itzuli beharko du.

main.py

```
# Definitu palindromoa izeneko funtzioa. Esaterako,
  palindromoa('amama')aginduak True balioa itzuli beharko
  du.

def palindromoa (hitza):
  aux = hitza.replace(' ','')
  if aux == aux[::-1]:
    print('Idatzitako karaktere-katea palindromoa da')
  else:
    print('Idatzitako karaktere-katea ez da palindromoa')
esaldia = input('Idatzi esaldi bat edo hitz bat:')
palindromoa(esaldia)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Idatzi esaldi bat edo hitz bat: amama
Idatzitako karaktere-katea palindromoa da
>
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Idatzi esaldi bat edo hitz bat: eguraldia
Idatzitako karaktere-katea ez da palindromoa
>
```

6.4 ariketa

Definitu `sortu_k_hizki` izeneko funtzioa, zeinak parametro gisa *k* zenbaki osoa eta hizki bat hartuko dituen, eta karakterea zenbaki horrekin biderkatuta itzuliko duen. Adibidez, `sortu_k_hizki(6,'p')` funtzioak `'pppppp'` itzuliko du.

main.py

```
# Definitu sortu_k_hizki izeneko funtzioa, zeinak parametro
  gisa k zenbaki osoa eta hizki bat hartuko dituen, eta
  karakterea zenbaki horrekin biderkatuta itzuliko duen.
  Adibidez, sortu_k_hizki(6,'p') funtzioak 'pppppp' itzuliko
  du.
def sortu_k_hizki(k, hizkia):
    print(k*hizkia)

print('Idatzi zenbaki bat eta hizki bat:')
zenbakia=input()
hizkia=input()
sortu_k_hizki(int(zenbakia),hizkia)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Idatzi zenbaki bat eta hizki bat:
 5
 k
kkkkk
>
```

6.5 ariketa

Definitu `histograma` izeneko funtzioa zeinak hiru zenbakiko zerrenda bat jasoko duen parametro gisa. Adibidez, `histograma([4,10,6])`funtzioak honako hau bistaratu beharko du:

```
****

**********

******
```

main.py

```
#Definitu histograma izeneko funtzioa zeinak hiru zenbakiko zerrenda bat
  jasoko duen parametro gisa. Adibidez, histograma([4,10,6])funtzioak
  honako hau bistaratu beharko du:
#****
#**********
#******


def f_histograma(zerrenda):
      for i in zerrenda:
        print (int(i) * 'x')
print ('Idatzi hiru balio:')
zenb1=input()
zenb2=input()
zenb3=input()
zerrenda=[zenb1,zenb2,zenb3]
f_histograma(zerrenda)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Idatzi hiru balio:
 9
 3
 6
xxxxxxxxx
xxx
xxxxxx
```

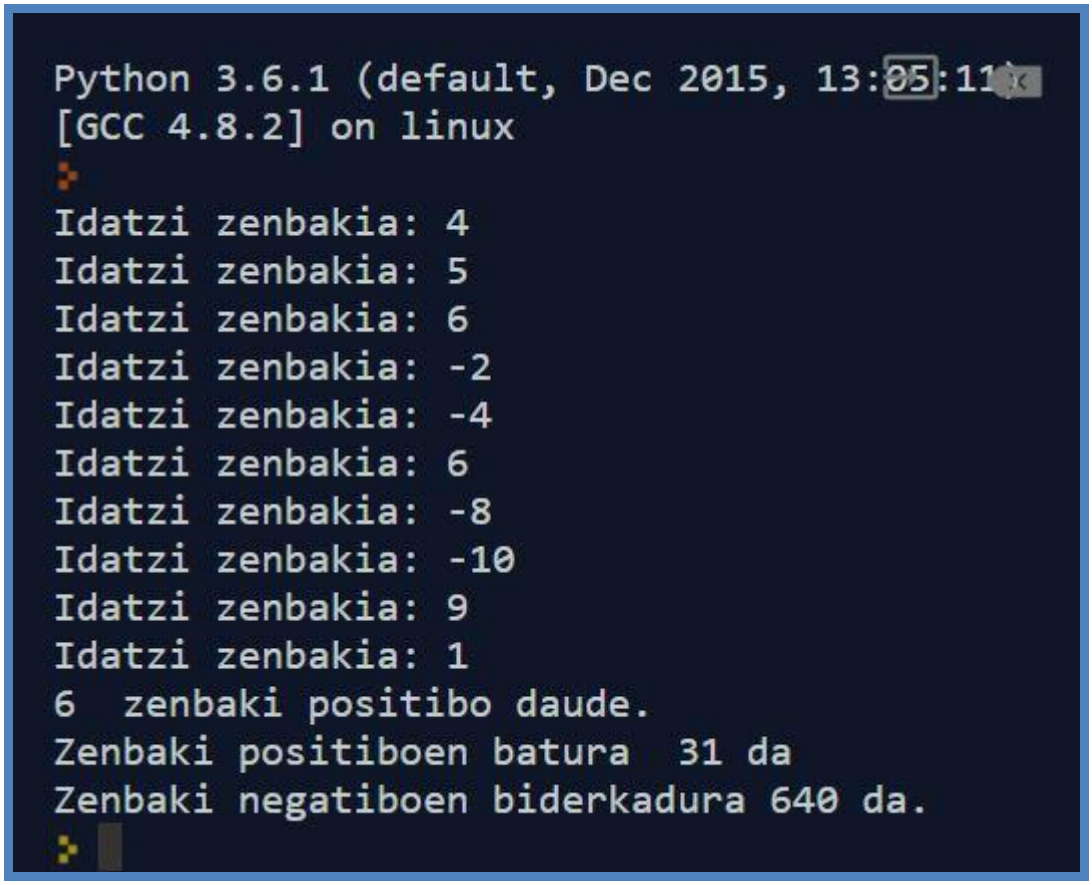
7. ATALA: ASKOTARIKO ARIKETAK

7.1. ariketa

Idatzi programa bat erabiltzaileari parametro gisa 10 zenbaki eskatzeko eta hurrengo balioak bistaratzeko:

- zenbaki positiboen kopurua
- zenbaki positiboen batuketa
- zenbaki negatiboen biderkadura

```
# Idatzi erabiltzaileari parametro gisa 10 zenbaki eskatuko dizkion programa eta hurrengo balioak
  bistaratuko dituena:
# zenbaki positiboen kopurua
# zenbaki positiboen batuketa
# zenbaki negatiboen biderkadura
zenbakiak=range(1,11)
zzenbakiak=[]
kont_positiboak=0
kont_negatiboak=0
sum_positiboak=0
bider_negatiboak=1
for i in zenbakiak:
  zenb=input('Idatzi zenbakia:')
  zzenbakiak.insert(i,zenb)
for i in zzenbakiak:
  if int(i)>0:
    kont_positiboak=kont_positiboak+1
    sum_positiboak=sum_positiboak+int(i)
  else:
    kont_negatiboak=kont_negatiboak+1
    bider_negatiboak=bider_negatiboak*int(i)
print (kont_positiboak,' zenbaki positibo daude.')
print('Zenbaki positiboen batura ',sum_positiboak,'da')
if kont_negatiboak == 0:
  print('Zenbaki negatiboen biderkadura 0 da.')
else:
  print('Zenbaki negatiboen biderkadura', bider_negatiboak, 'da.')
```


Python 3.6.1 (default, Dec 2015, 13:05:11
[GCC 4.8.2] on linux
Idatzi zenbakia: 4
Idatzi zenbakia: 5
Idatzi zenbakia: 6
Idatzi zenbakia: -2
Idatzi zenbakia: -4
Idatzi zenbakia: 6
Idatzi zenbakia: -8
Idatzi zenbakia: -10
Idatzi zenbakia: 9
Idatzi zenbakia: 1
6 zenbaki positibo daude.
Zenbaki positiboen batura 31 da
Zenbaki negatiboen biderkadura 640 da.

7.2. ariketa

Idatzi programa bat zenbaki oso batek dituen digitu-kopurua bistaratzeko.

main.py

```
# Idatzi programa bat zenbaki oso batek dituen digitu-kopurua
  bistaratzeko.

zenbakia=input('Idatzi zenbaki bat:')
kont=0
digitu = 1

for i in str(zenbakia):
  kont=kont+1
print('Idatzitako zenbakiak',kont,'digitu ditu.')
```

```
Python 3.6.1 (default, Dec 2015,
13:05:11)
[GCC 4.8.2] on linux
>
Idatzi zenbaki bat: 5478964
Idatzitako zenbakiak 7 digitu ditu.
>
```

7.3. ariketa

Idatzi programa bat erabiltzaileak parametro gisa zenbaki bat eta posizio bat idazteko eta posizio horretan zein zenbaki dagoen bistaratzeko.

main.py

```
# Idatzi programa bat erabiltzaileak parametro gisa zenbaki bat
  eta posizio bat idazteko eta posizio horretan zein zenbaki
  dagoen bistaratzeko.

print('Idatzi zenbaki bat eta posizio bat')
zenb=input()
pos=int(input())
print (zenb[pos])
```

```
Python 3.6.1 (default, Dec 2015,
13:05:11)
[GCC 4.8.2] on linux
Idatzi zenbaki bat eta posizio bat
 9874563219
 5
6
```

7.4. ariketa

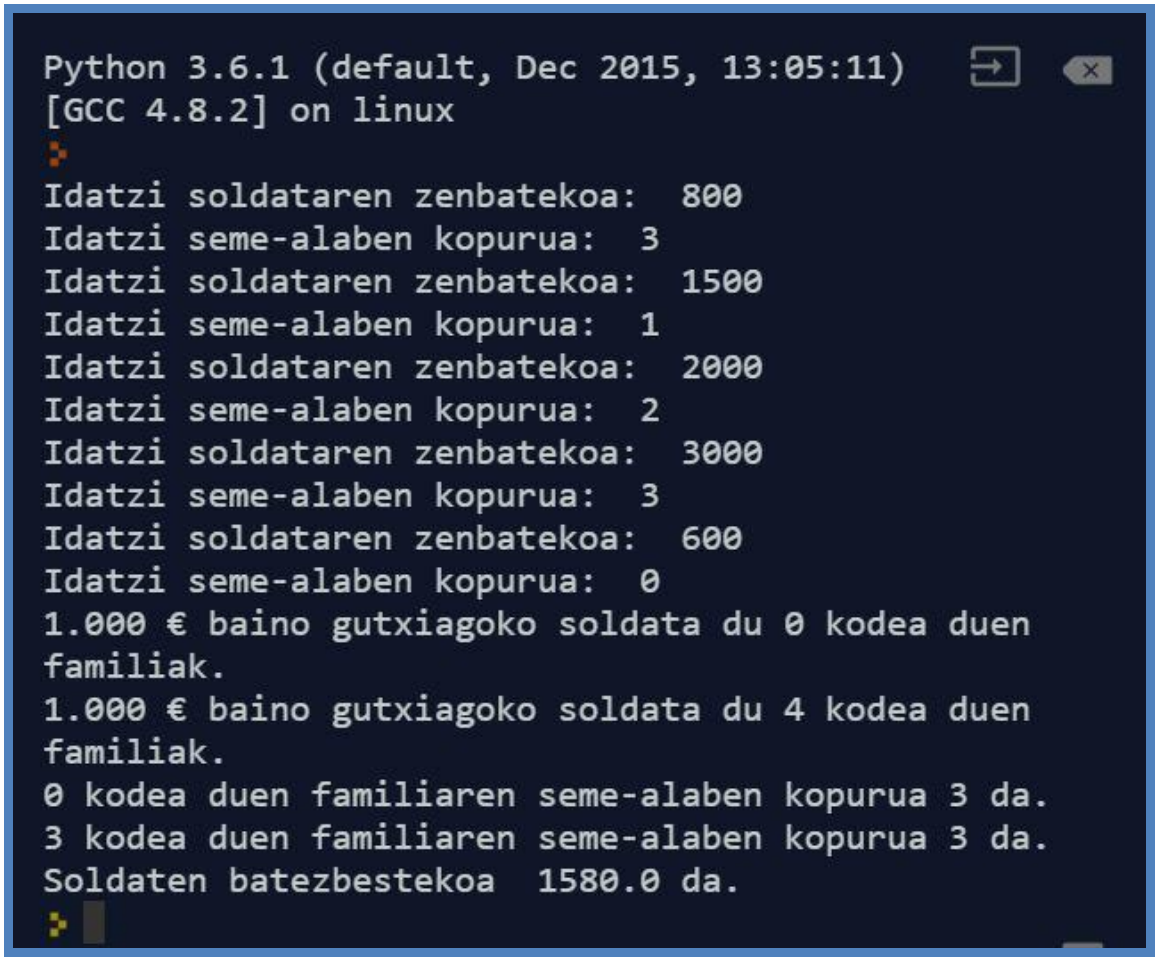
Idatzi programa bat zeinak erabiltzaileari 5 familien inguruko datu hauek eskatuko dizkion:

- kodea
- soldataren zenbatekoa
- seme-alaben kopurua

Bistaratu:

- soldata 1.000 eurokoa baino baxuagoa duten familien kodeak eta soldatak.
- familia anitzen kodea eta seme-alaben kopurua.
- familia guztiak kontuan izanik, soldaten batezbestekoa

```
i=0
hiztegia1={}
hiztegia2={}
batu=0
while i<5:
  soldata=input("Idatzi soldataren zenbatekoa: ")
  s_a_kop=input("Idatzi seme-alaben kopurua: ")
  hiztegia1[i]=soldata
  hiztegia2[i]=s_a_kop
  i+=1
# aztertu soldataren zenbatekoa
for z in hiztegia1:
  if int(hiztegia1[z])< 1000:
    print('1.000 € baino gutxiagoko soldata du', z, 'kodea duen familiak.')
#aztertu seme-alaben kopurua
for z in hiztegia2:
  if int(hiztegia2[z])>2:
    print(z,'kodea duen familiaren seme-alaben kopurua', hiztegia2[z], 'da.')
#soldaten batezbestekoa
for z in hiztegia1:
  batu=batu + int(hiztegia1[z])
print('Soldaten batezbestekoa ', batu/5 ,'da.')
```


Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Idatzi soldataren zenbatekoa: 800
Idatzi seme-alaben kopurua: 3
Idatzi soldataren zenbatekoa: 1500
Idatzi seme-alaben kopurua: 1
Idatzi soldataren zenbatekoa: 2000
Idatzi seme-alaben kopurua: 2
Idatzi soldataren zenbatekoa: 3000
Idatzi seme-alaben kopurua: 3
Idatzi soldataren zenbatekoa: 600
Idatzi seme-alaben kopurua: 0
1.000 € baino gutxiagoko soldata du 0 kodea duen familiak.
1.000 € baino gutxiagoko soldata du 4 kodea duen familiak.
0 kodea duen familiaren seme-alaben kopurua 3 da.
3 kodea duen familiaren seme-alaben kopurua 3 da.
Soldaten batezbestekoa 1580.0 da.