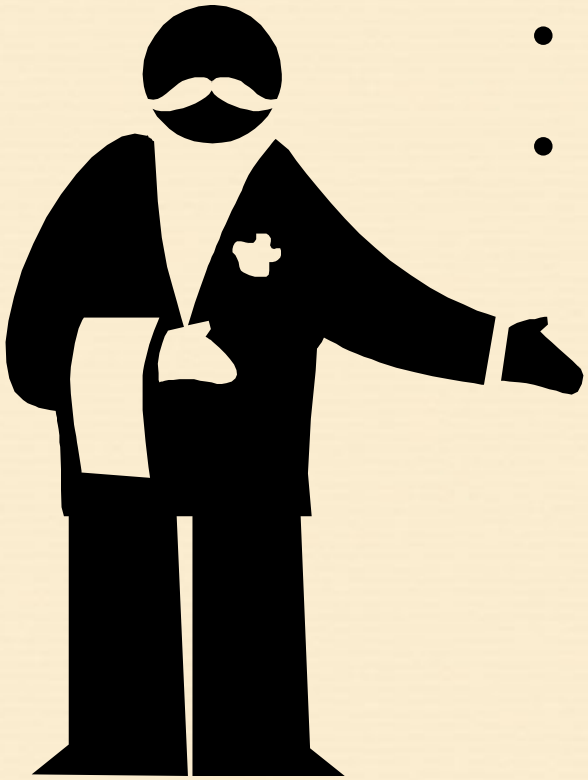
2.4 Gaia. Ada eta Python erabiltzeko oinarrizko gida

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Aurkibidea

- **Gaiaren helburuak**
- Garapen inguruneak
- Adaren oinarrizko gida
- Pythonen oinarrizko gida



Gaiaren helburuak

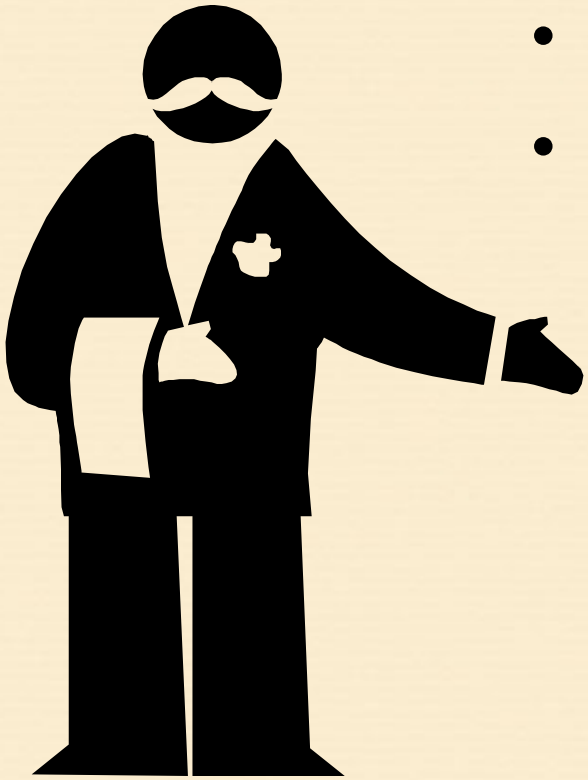
- Ikasgaian erabiliko ditugun lengoaien datu mota eta agindu oinarrizkoak aurkeztea
  - Eclipse + Ada-GNATbench pluginarekin lan egitea
  - CodeSkuptor-ekin lan egitea

Zergatik Ada eta Python?

- Programazioa ulertzeko bi modu oso ezberdin
  - Ada oinarrizko kontzeptuak barneratzeko oso egokia da
  - Python gaurkotasunean asko erabiltzen den lengoaia da

Aurkibidea

- Gaiaren helburuak
- **Garapen inguruneak**
- Adaren oinarrizko gida
- Pythonen oinarrizko gida



Eclipse + Ada-GNATbench

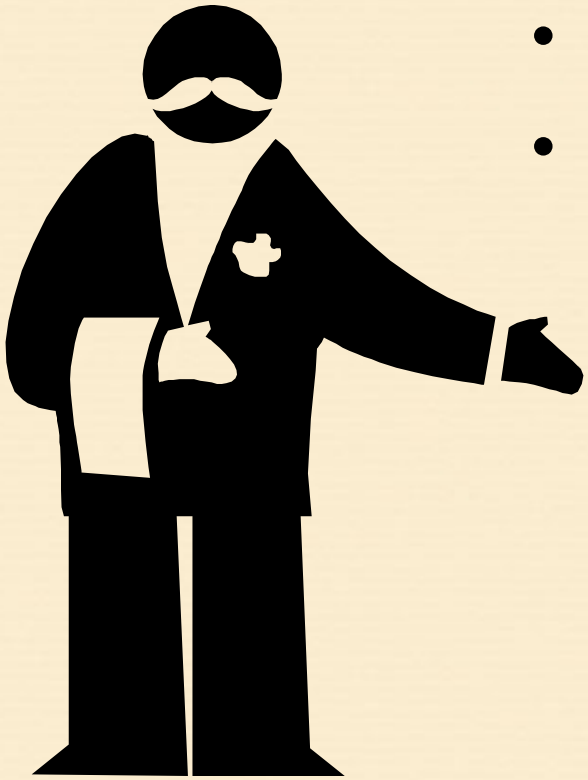
- eGelan duzue instalaziorako beharrezkoa den informazio guztia

CodeSkulptor

- Python-en idatzitako programak idazteko CodeSkulptor erabiliko dugu
  - http://www.codeskulptor.org/ (Bertsio originala)
- Ez du instalaziorik behar, web nabigatzailean dabil

Aurkibidea

- Gaiaren helburuak
- Garapen inguruneak
- **Adaren oinarrizko gida**
- Pythonen oinarrizko gida



Ezaugarri orokorrak

- Adak moten kudeaketa oso zorrotza egiten du
- Ez ditu letra larri eta xeheak bereizten aldagaien izenetan
- Ekintzak puntu eta komaz bukatzen dira (;)
  - Ohitura ona: lerro bat = ekintza bat

Programa baten oinarrizko egitura

```
with Package_Name;
use Package_Name;
procedure Program_Name is
   ---Specification
   Variable : Some_Type;
begin
   Statement_1;
   ...
   Statement_n;
end Program_Name;
```

Adibidea: Kaixo mundua

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure kaixo_mundua is
   ---Sarrera:-
   ---Aurre: -
   ---Irteera: mezu bat
   ---Post: “kaixo mundua!” idatzi
begin
   put(“kaixo mundua!");
end kaixo_mundua;
```

Oharrak

- Gidoietatik (--), lerro bukaerara arte

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
  -- Goiko bi lerroak testuinguaren klausulak dira, testua eta balio osokoak
  -- irakurri eta idatzi ahal izateko.
procedure proba is
Zen1, Zen2: Integer ; --Aldagaien erazagupena
begin
put("Zenbaki osoko bat idatzi: "); -- idatzi
get(Zen1);                         -- irakurri
put("Beste zenbaki osoko bat idatzi: ");
get(Zen2);
new_line; -- Lerro saltoa (pantailan)
put("Gehiketaren balioa: ");
put(Zen1 + Zen2);
end proba;
```

Oinarrizko datu motak Adaz (I)

- Integer (zenbaki osokoak)
  - Tartea [Integer'First, Integer'Last]
  - Eragiketak =, /=, >, >=, <, <=, +, -, *, /, rem, **
- Float (zenbaki errealak)
  - Tartea [Float'First, Float'Last]
  - Eragiketak =, /=, >, >=, <, <=, +, -, *, /, **
    (float** integer)

Oinarrizko datu motak Adaz (II)

- Character (karaktereak)
  - Eragiketak =, /=, >, >=, <, <=
- Boolean (boolearrak)
  - Eragiketak and, or, xor, not
- String (karaktere kateak)
  - Eragiketak =, /=, >, >=, <, <=

Aldagaien erazagupena

- Adibideak
  - Zen1, Zen2 : Integer;
  - Erro_karratua : Float;
  - Kar, Letra : Character;
  - Lerroa : String(1..80);
- Hasieraketarekin
  - Salataria : Boolean:= False;
  - Kont : Integer := 0;

Konstanteen erazagupena

- **constant** hitz erreserbatuarekin erazagutzen dira eta hasierako balioa ematen zaie
  - Izena : constant String(1..4) := “ACME”;
  - Pi : constant Float := 3.1416;
  - Max : constant Integer := 100;

Adi, galdera

- Zergatik da interesgarria konstante baten erazagupena eta erabilera, bere balioa zuzenean aldagai batean erabili beharrean?

eman ta zabal zazu

Esleipena

- := bidez adierazten da
- Moten baliokidetza egotea ezinbestekoa da
  - Aldagaiaren mota, adierazpenaren motarekin bat egin behar du

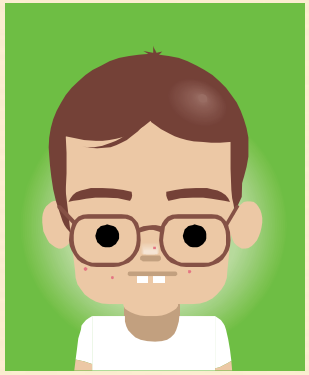
```
Batazbestekoa : Float ;
Notak, Ikasle_kop := Integer ;

Batazbestekoa := Notak / Ikasle_kop ;
-- ERROREA!

Batabestekoa    :=    float(Notak)    /
float(Ikasle_kop) ;
```

Datuen sarrera eta irteera

- get (irakurri) eta put (idatzi) eragileek egiten dute
  - get/put bikote bat dago osokoentzat, beste bat errealentzat, eta abar.
    - with Ada.Text_IO; use Ada.Text_IO;
    - with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
    - with Ada.Float_Text_IO; use Ada.Float_Text_IO;



Modu honetan, put edo get eragileakerabiltzean, ez dugu mota zehatza adierazi beharko, ADAk testuinguruagatik ebatziko baitu. Horri, eragileen gainkarga deritzo.

Baldintzazko egiturak

- if *baldintza* then
  ekintza(k);
  end if;

- *if baldintza* then
  ekintza(k);
  else
  ekintza(k);
  end if;

- if *baldintza* then
  ekintza(k);
  elsif *baldintza* then
  ekintza(k);
  ...
  [else
  ekintza(k);]
  end if;

Adibideak

```
...


if Zen < 0 then
   put("Zenbakia negatiboa da");
end if;


...


if Zen rem 2 = 0 then
   put("Zenbakia bikoitia da");
else
   put("Zenbakia bakoitia da");
end if;


...
```

```
...


if Zen < 0 then
   put("Zenbakia negatiboa da");
elsif Zen = 0 then
   put("Zenbakia zero da");
else
   put("Zenbakia positiboa da");
end if;


...
```

Iteraziozko egiturak

- loop
  ekintza(k);
  exit when *baldintza;*
  end loop;

- loop exit when *baldintza;*
  ekintza(k);
  end loop;

- while *baldintza* loop
  ekintzak(k);
  end loop;

- for *aldagaia* in n1 .. n2 loop
  ekintza(k);
  end loop;

- for *aldagaia* in reverse n2..n1 loop
  ekintza(k);
  end loop;

Adibideak

```
...

Kont := 1;
loop exit when Kont > 0;
    put(Kont);
    Kont := Kont + 1;
end loop;


...


Kont := 1;
while Kont <= Zenb loop
   put(Kont);
   Kont := Kont + 1;
end loop;


...
```

```
...

for Kont in 1 .. Zenb loop
   put(Kont);
end loop;

...

for Kont in reverse Zenb .. 1 loop
   put(Kont);
end loop;

...

loop
    put("Zenbaki positibo bat idatzi");
    get(Zenb);
    exit when Zenb > 0;
end loop;

...
```

Azken adibidea: berreketa kalkulatu

Algoritmoa

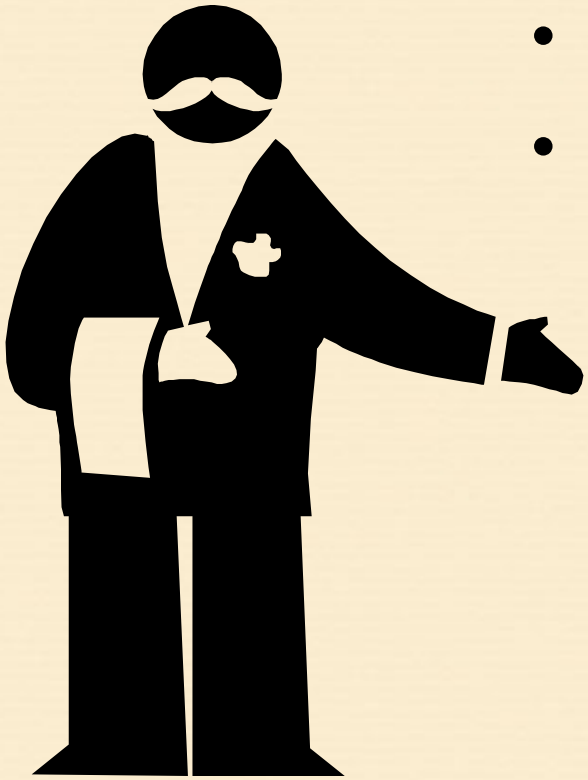
```
zen1,zen2,akum,kont: Integer;
irakurri(zen1);
irakurri(zen2);
baldin zen1=0 eta zen2=0 orduan
   idatzi("Balio zehaztugabea");
bestela
   akum <-- 1;
   kont <-- 0;
   errepikatu kont=zen2 bete arte;
      akum <-- akum * zen1;
      kont <-- kont + 1;
   amerrepikatu;
   idatzi(akum);
ambaldin;
```

Programa

```
with Ada.Text_IO,Ada.Integer_Text_IO;
use Ada.Text_IO,Ada.Integer_Text_IO;
procedure berreketa_kalkulatu is
  Zen1,Zen2,Akum,Kont:Integer;
begin
   get(Zen1);--datuak jaso
   get(Zen2);
   if Zen1=0 and Zen2=0 then -- kasu kritikoa
      put("Balio zehaztugabea");
   else -- kasu orokorrak
      Akum := 1;
      Kont := 0;
      loop exit when Kont=Zen2;
        Akum := Akum * Zen1;
        Kont := Kont + 1;
      end loop;
      put("Emaitza: ");
      put(Akum);
   end if;
end potentzia_kalkulatu ;
```

Aurkibidea

- Gaiaren helburuak
- Garapen inguruneak
- Adaren oinarrizko gida
- **Pythonen oinarrizko gida**



Ezaugarri orokorrak

- Pythonek ez ditu aldagaien motak ez erazagutzen, ez kontrolatzen ere
- Letra larri eta xeheen artean bereizten du
- Kodearen tabulazioa sintaxiaren parte da
- Lerro bakoitzean ekintza bakarra egon daiteke

Programa baten oinarrizko egitura

```
def program_name():
    #Specification
    Statement_1
    ...
    Statement_n;
```

Adibidea: Kaixo mundua

```
def kaixo_mundua()
    #Sarrera:-
    #Aurre: -
    #Irteera: mezu bat
    #Post: “kaixo mundua!” idatzi
    print(“kaixo mundua!")

#erabiltzeko:
kaixo_mundua()
```

Oharrak

- Almohadilatik (#) lerro bukaerara

```
# Proba programa
def proba():
    zen1 = 3 # Aldagaiak ez dira erazagutzen!
    zen2 = 2
    print("Gehiketaren balioa:")
    print(zen1 + zen2)

#Probari deia, exekutatu ahal izateko
proba()
```

Oinarrizko datu motak Pythonez

- int, long (zenbaki osokoak)
  - Eragiketak ==, !=, >, >=, <, <=, +, -, *, /, //, %, **
    - / zatiketa erreala, // zatiketa osokoa
- float (zenbaki errealak)
  - Eragiketak ==, !=, >, >=, <, <=, +, -, *, /, **
- str (ez da karaktare mota existitzen, kateak dira)
  - Eragiketak ==, !=, >, >=, <, <=, + (kateaketa)
- bool (boolearrak)
  - Eragiketak and, or, not

Aldagaien erazagupena

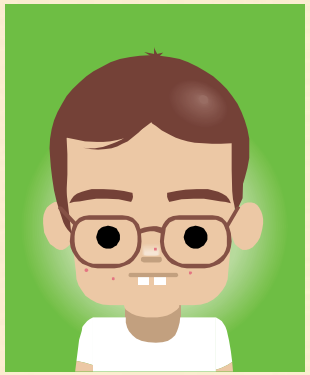
- Pythonen ez dira aldagaien motak kontrolatzen
  - Ez dira aldagaiak erazagutzen
  - Ez zaie datu mota bat esplizituki esleitzen
    - Exekuzioan zehar, datu mota alda daiteke
  - Ez dago konstanteak adierazteko modu espliziturik

Esleipena

- = bidez adierazten da
  - aldagaia = 1
  - aldagaia = 3.0
  - aldagaia = 'a'
  - aldagaia = “a”
  - aldagaia = True

Komila bikoitzak edo soilak erabil daitezke, alderik egon gabe. Koherentzia mantendu behar da ireki eta ixteko. Horrela komilak habiara daitezke, adibidez:

print(“Lehenik 'Kaixo mundua!' esango dut.”)



eman ta zabal zazu

Datuen sarrera

- input() eragiketak karaktereak irakurtzen ditu, eta zenbaitetan zenbaki bilakatu beharko ditugu
  - adina = int(input(“Sartu zure adina: ”))
  - nota = float(input(“Sartu lortutako nota: “))
  - izena = input(“Sartu zure izena: “)

Datuen irteera

- print() eragiketak pantailatik idazten du, eta defektuz lerro saltoa jartzen du
  - Lerro saltoa ekiditeko *end* parametroa erabili behar da → print('Kaixo',end=' ')
    - print(“Kaixo mundua”)
    - print('honako izena dauka: ' + izena)
    - print(“ta adina: ” + str(adina))
    - print(adina)

Baldintzazko egiturak

- if *baldintza:*

  ekintza(k)

- *if baldintza* :

  ekintza(k)

  else:

  ekintza(k)

- if *baldintza:*

  ekintza(k)

  elif *baldintza:*

  ekintza(k)

  ...

  [else:

  ekintza(k)]

Adibideak

```
...


if zen < 0:
   print("Zenbakia negatiboa da")


...


if zen rem 2 == 0:
   print("Zenbakia bikoitia da")
else:
   print("Zenbakia bakoitia da")


...
```

```
...


if zen < 0:
   print("Zenbakia negatiboa da")
elif zen == 0:
   print("Zenbakia zero da")
else:
   print("Zenbakia positiboa da")


...


if zen != 0:
    print(“Zenbakia ez da zero”)
    if zen 0:
        print(“Zenbakia positiboa da”)
    else:
        print(“Zenbakia negatiboa da”)
else:
    print(“Zenbakia zero da”)
```

Iteraziozko egiturak

- while *baldintza* :
  ekintzak(k)

- for *aldagaia* in range(n1,n2):
  ekintza(k)

- for *aldagaia* in range(n2,n1,-1):
  ekintza(k)

Tarteak (range)

- Pythonen tarteak funtzionamendu berezia du
  - range(hasiera, bukaera, saltoa)
    - Hasiera barne dago, baino ez bukaera
    - saltoa hautazkoa da, eta defektuz 1 balio du
    - hasiera ere hautazkoa da, eta defektuz 0 balio du

Adi, galdera

- Zer inprimatuko du hurrengo adibideetako bakoitzak?

```
zen = 20
...
i = 1
while i < zen :
    print(i)
    i = i + 1
...
for i in range (1, zen) :
    print(i)
...
for i in range (zen, 0, -1) :
    print(i)
...
for i in range (num) :
    print(i)
```

Azken adibidea: berreketa kalkulatu

Algoritmoa

```
zen1,zen2,akum,kont: Integer;
irakurri(zen1);
irakurri(zen2);
baldin zen1=0 eta zen2=0 orduan
   idatzi("Balio zehaztugabea");
bestela
   akum <-- 1;
   kont <-- 0;
   errepikatu kont=zen2 bete arte;
      akum <-- akum * zen1;
      kont <-- kont + 1;
   amerrepikatu;
   idatzi(akum);
ambaldin;
```

Programa

```
def berreketa_kalkulatu
    # datuak jaso
    zen1 = int(input())
    zen2 = int(input())
    if zen1 == 0 and zen2 == 0:
        # kasu kritikoa
        print("Balio zehaztugabea")
    else:
        # kasu orokorrak
        akum = 1
        kont = 0
        while kont < zen2:
            akum = akum * zen1
            kont = kont + 1
        print("Emaitza: "+str(akum))
```

Informazio gehiago

- Ada
  - Urretavizcaya: MiniManual Lenguaje Ada, eGelan
- Python
  - Python Software Foundation documentation, https://www.python.org/doc web orrian atzigarri
  - Greiner: CodeSkulptor documentation, www.codeskulptor.org/docs.html web orrian atzigarri
  -