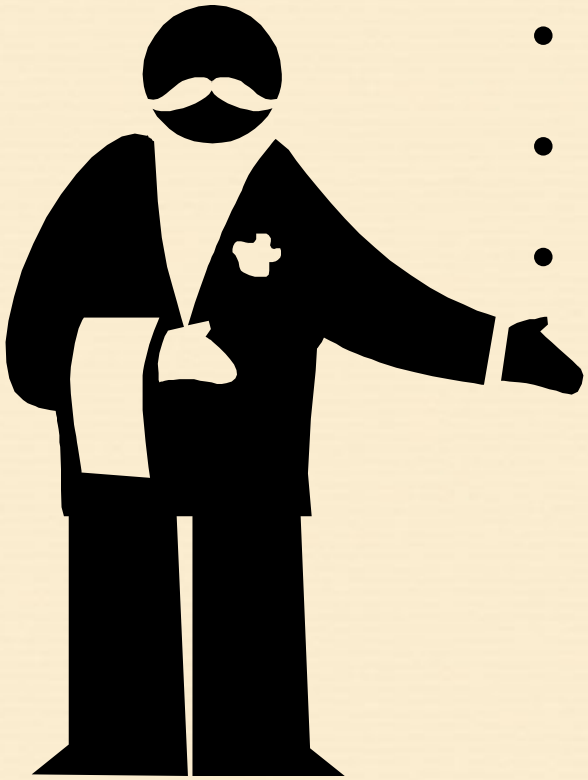
3.1 Gaia. Beheranzko diseinua: azpiprogramak

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Aurkibidea



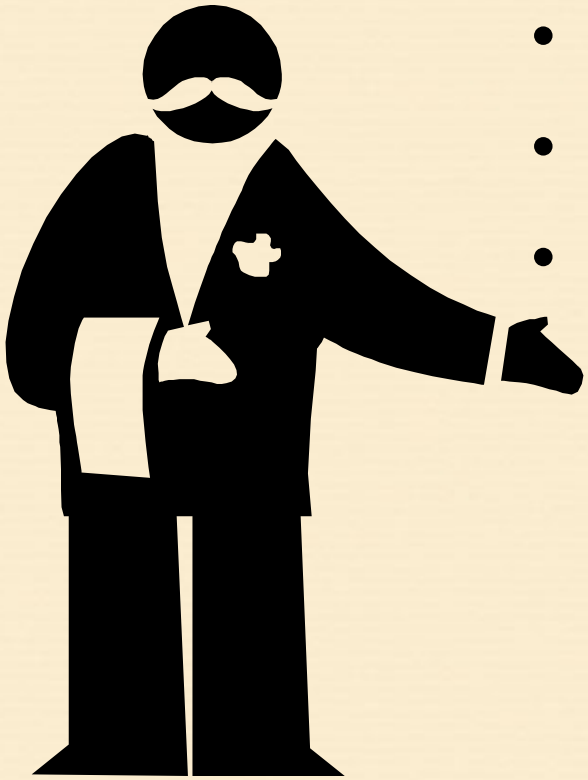
- **Gaiaren helburuak**
- Motibazioa
- Kontzeptu orokorrak
- Azpiprogramak Adaz
- Azpiprogramak Pythonez
- Ohitura onak

Gaiaren helburuak

- Atazak azpiprogrametan banatzeko beharra, bereziki eginbeharrekoa konplexua bada
- Azpiprograma motak: funtzioak eta prozedurak
  - Definizioa, parametrizazioa eta espezifikazioa
  - Lengoaia ezberdinetan erabilera
- Azpiprogramak programatzeko ohitura onak

Aurkibidea

- Gaiaren helburuak
- **Motibazioa**
- Kontzeptu orokorrak
- Azpiprogramak Adaz
- Azpiprogramak Pythonez
- Ohitura onak



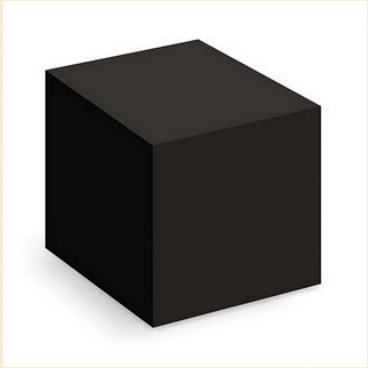
Beheranzko diseinua

- “Zatitu eta irabaziko duzu”
  - Hasierako ataza, ataza sinpleagoetan banatzen da
    - Azpiprograma bakoitzak ataza nagusiaren zati bat egingo du
    - Azpiprograma bakoitzak bere helburu propioa izango du

eman ta zabal zazu

Abstrakzioa

- Azpiprograma bati dei dakiokegu **nola** egiten duen ezagutu gabe, beti ere **zer** egiten duen jakinda
- Abstrakzioa = **nola** eta **zer** arteko bereizketa
  - Ataza konplexuak ebazteko gakoa da
  - Ez da kodea irakurri behar, azpiprogramaren espezifikazioa irakurtzearekin nahikoa da



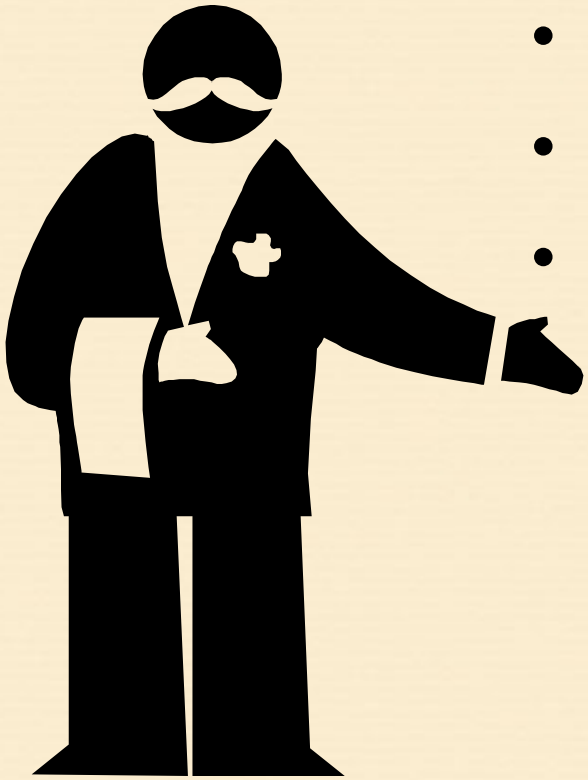
Aurredefinitutako azpiprogramak

- Programazio-lengoaiek azpiprogama estandarrak dituzten moduluak (edo liburutegiak) eskaintzen dizkigute, oso erabilgarriak direnak
  - Ez dira sortu behar, eta programatzaileok adituek sortutako algoritmoak berrerabil ditzakegu
  - Adibidez, funtzio matematikoak (sin, cos, sqrt,...)
    - Ada: Ada.Numerics.Generic_Elementary_Functions paketea
    - Python: math modulua
    - Java: Math klasea
    - C: math.h buruko fitxategia liburutegi estandarretik

Abantailak

- Ez da programazioa ulertzen azpiprogramarik gabe
  - Azpiprograma bakoitza bere aldetik definitu eta erabiltzen da, eta horrela...
    - Abstrakzioan, irakurgarritasunean eta aldakortasunean laguntzen du
    - Kodearen berrerabilera sustatzen du
    - Lantaldeko programazioa eta probak errazten ditu
    - Erroreak bilatzea eta zuzentzea errazten du
    - Besteen programak ulertzeko denbora gutxitzen du

Aurkibidea



- Gaiaren helburuak
- Motibazioa
- **Kontzeptu orokorrak**
- Azpiprogramak Adaz
- Azpiprogramak Pythonez
- Ohitura onak

eman ta zabal zazu

Datuen elkartrukaketa

- Azpiprogramek datuak jaso ditzakete beraien ataza egin ahal izateko edota emaitzak itzuli ahal izateko.
  - Parametroen bitartez eta *return* ekintzaren bitartez

```
...

n <-- ... ;

fakt_n <-- faktoriala(n);

idatzi(n "-ren faktoriala: " fakt_n);

...
```

Berrerabilpena

- Azpiprograma baten definizioa orokorra da
  - Ondorio berdina lor daiteke datuen balio ezberdinetarako (aldagai edo adierazpenak)

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

```
...
n <-- ... ;
k <-- ... ;
fakt_n <-- faktoriala(n);
fakt_k <-- faktoriala(k);
fakt_nk <-- faktoriala(n - k);
konb <-- fakt_n / (fakt_k * fakt_nk);
idatzi("Emaitza: " konb);
...
```

Parametro formalak

- Azpiprograma baten parametroen izenak, bere burukoan adierazten diren moduan
  - Azpiprograma barruan bakarrik existitzen dira

Parametro errealak

- Parametro formalei esleitzen zaien balioak (aldagaiak, adierazpenak...) azpiprogramari deitzen zaionean
  - Dei ezberdinak == parametro erreal ezberdinak
  - Dei bakoitzean parametro errealen kopurua eta horien motak bat egin behar dute

Azpiprograma bati deia

- A (azpi)programa batek, beste B azpiprograma bati deitzen badio
  - Aren egikaritzea gelditzen da, eta Bri ematen dio kontrola
    - Parametro errealen eta formalen arteko harremana ezartzen da
    - Bren aldagaientzat memoria erreserbatzen da
  - Bren egikaritzea bukatzen denean, Ak emaitza jasotzen du eta egikaritzea jarraitzen du

return ekintza

- Azpiprograma baten emaitzak itzultzeko erabiltzen da
  - Emaitza gisa zein adierazpen ebatziko den adierazten du
- *return* egikaritzen denean azpiprograma bukatzen da

```
(azpi)programa nagusia
  ...
  a1 <-- ... ;
  a2 <-- ... ;
  ema <-- konbinatoria(a1,a2);
  ...
```

```
azpiprograma konbinatoria(n, k : Integer)
return Integer

     fakt_n, fakt_k, fakt_nk : Integer;
hasiera
         fakt_n  <-- faktoriala(n);
         fakt_k  <-- faktoriala(k);
         fakt_nk <-- faktoriala(n-k);
         return fakt_n / (fakt_k * fakt_nk);
amaiera konbinatoria;
```

Aldagaien ikusgarritasuna

- Aldagaiak, definituta dauden azpiprogramaren gorputzean baino ez dira ikusgarriak
  - Ondorioz, ez dira existitzen eta ezin dira beste batzuetan erabili
    - Salbuespenak (aldagai globalak...) ez ditugu oraindik ikusiko

```
(azpi)programa nagusia
  a1,a2:Integer;
hasiera
  ...
  a1 <-- ... ;
  a2 <-- ... ;
  fakt_n <-- ...; -- ERROREA
  ...
amaiera nagusia;
```

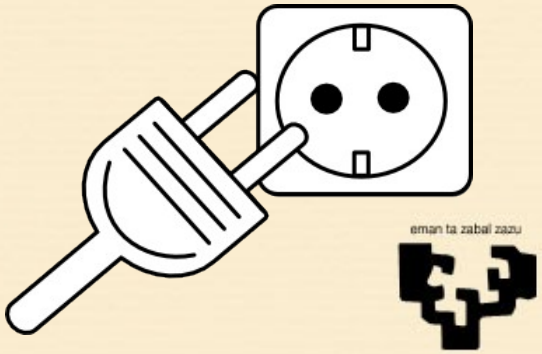
```
azpiprograma konbinatoria(n, k : Integer) return
Integer

    fakt_n, fakt_k, fakt_nk : Integer;
hasiera
        fakt_n  <-- faktoriala(a1); -- ERROREA
        ...
amaiera konbinatoria;
```

Aldagaien ikusgarritasuna

- Azpiprogramaren egikaritzea bukatzean, definitutako aldagai guztiak desagertzen dira (existitzeari uzten diote, memoria hori askatzen da)
  - Azpiprogramari berriz deitzen bazaio, aldagai berriak sortuko dira

Azpiprograma bati deia

- Oso garrantzitsua da azpiprogramen espezifikazioa
  - Espezifikazioaren irakurketarekin soilik, beste programatzaile batek azpiprogramak egiten duena ulertu beharko du
- Espezifikazioan ez dira parametroak edota euren motak adierazten, soilik euren ezaugarriak

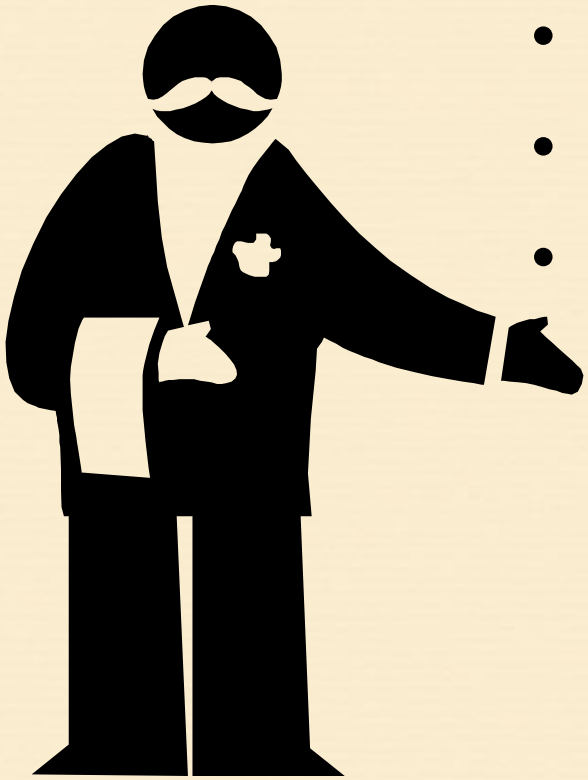


Adibideak

```
azpiprograma asteriskoak_idatzi (Zenbat : Integer)

    ---Aurrebaldintza: Zenbat >=0
    ---Postbaldintza: Zenbat asterisko sinbolo idatzi dira
```

```
azpiprograma letra_da (Kar : Character) return Boolean

    ---Aurrebaldintza:
    ---Postbaldintza: Kar letra bat bada, True itzultzen du.
False bestela
```

Aurkibidea



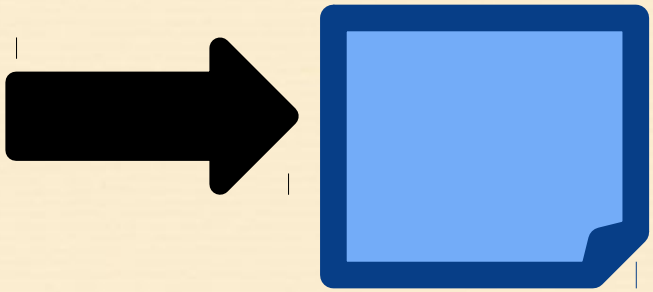
- Gaiaren helburuak
- Motibazioa
- Kontzeptu orokorrak
- **Azpiprogramak Adaz**
- Azpiprogramak Pythonez
- Ohitura onak

Parametro motak Adaz

- Hiru parametro mota definitzen dira, azpiprogramari datuak emateko edota bertatik jasotzeko erabiliko denaren arabera
  - Sarrera parametroak
  - Irteera parametroak
  - Sarrera-irteera parametroak

Sarrera parametroak

- Azpiprograma bati datuak emateko erabiltzen dira
  - Sarrerako parametro formalek parametro errealetatik jasotzen dituzte balioak
  - Azpiprograma egikaritzen den bitartean euren balioak ezin dira aldatu (konstante lokalak bailira)



Sarrera parametroak

- Adaz, **in** hitz erreserbatua erabiltzen da (hautazkoa) parametro bat edo batzuk sarrerakoak direla adierazteko
  - Baliokideak dira
    - azpiprograma idatzi_asteriskoak(Zenbat : Integer)
    - azpiprograma idatzi_asteriskoak(Zenbat : in Integer)

```
azpiprograma asteriskoak_idatzi(Zenbat : in Integer)
  ...
begin
  for I in 1..Zenbat loop
     put("*");
  end loop;
end asteriskoak_idatzi;
```

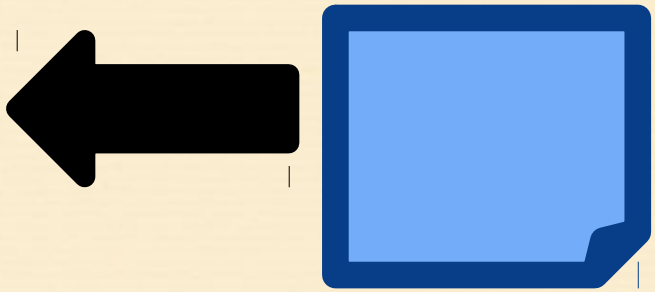
Parámetro errealak (*in*): Parametro formalen mota berdineko **adierazpenak**

Deien adibideak:

```
asteriskoak_idatzi(7);
asteriskoak_idatzi(2*N+4);
```

Irteera parametroak

- Azpiprograma batetatik balioak itzultzeko erabiltzen dira
  - Irteerako parametro formal bakoitzak, azpiprogramako aldagai gisa erabiltzen da
  - *aldagai* honek azpiprogramaren exekuzioa bukatzen denean duen balioa esleituko zaio parametro errealari
    - Irteerako parametro erreal bat beti izango da aldagai bat



Irteera parametroak

- Adaz, **out** hitz erreserbatua erabiltzen da parametro bat edo batzuk irteerakoak direla adierazteko

Parametro errealak (*in*): Parametro formalen mota berdineko **adierazpenak**

Parametro errealak (*out*): Parametro formalen mota berdineko **aldagaiak**

```
azpiprograma zatiketa_moztua
              (Zatikizuna, Zatitzailea: in Integer;
               Zatiketa, Hondarra: out Integer)

  -- Azpiprograma honek bi datu jasotzen
  -- ditu eta bi itzultzen ditu
   ...
```

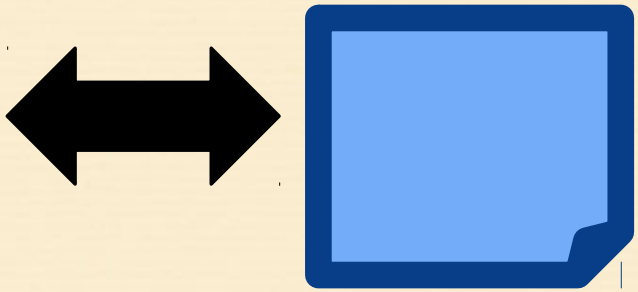
Deien adibideak:

```
zatiketa_moztua (17, 7, N1, N2);
zatiketa_moztua (X, Y, Zat, Ema);
```

```
begin
   Zatiketa := Zatikizuna / Zatitzailea;
   Hondarra := Zatikizuna rem Zatitzailea;
end zatiketa_moztua;
```

Sarrera-irteera parametroak

- Aldagaien balioak eguneratzeko erabiltzen dira
  - Parametro errealaren balioa hartzen da eta aldatzen da
  - Irteera parametroen moduan funtzionatzen dute, baina kasu honetan hasierako balio bat izango dute (sarrerakoa)



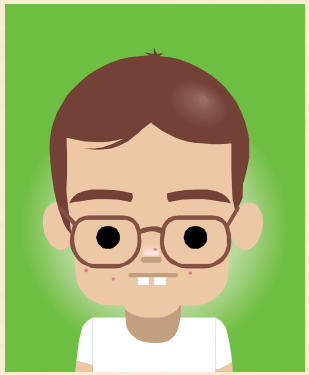
Sarrera-irteera parametroak

- Adaz, **in out** hitz erreserbatuak erabiltzen dira parametro bat edo batzuk sarrera-irteerakoak direla adierazteko

```
azpiprograma ainkrementatu(Kont : in out Integer)
   ...
begin
     Kont := Kont + 1;
end inkrementatu;
```

Parámetro errealak (*in out*): Parametro formalen mota berdineko **aldagaiak**

```
Deien adibideak:

        inkrementatu(N);
        inkrementatu(Kont);
```



Bigarren adibidean, parametro errealaren izena parametro formalaren berdina da (Kont).

Arazo bat izan daiteke?

eman ta zabal zazu

Azpiprograma motak Adaz

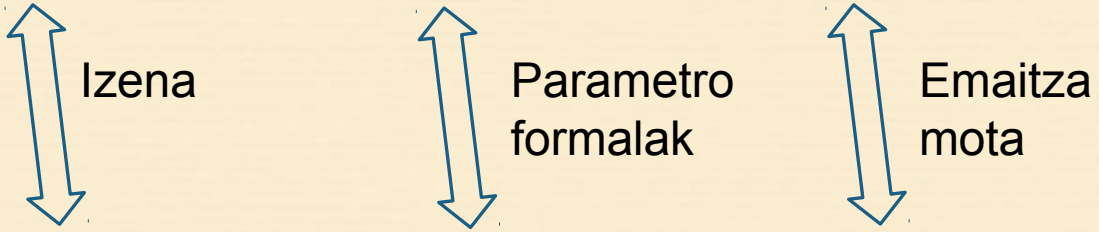
- Adak bi azpiprograma moten arteko bereizketa esplizitua egiten du
  - Funtzioak
  - Prozedurak

Funtzioak

- Adierazpen baten edo eragiketa baten parekoak dira
  - Parametro guztiak sarrerakoak dira
  - Emaitza bakar bat kalkulatzen dute eta *return* ekintzaren bidez itzultzen dute
  - Exekuzioan, itzultzen duten balioagatik ordezkatuak dira

Adibideak (burukoak)

```
function letra_da (Kar : in Character) return Boolean is ...
```




```
function balio_absolutua (N : in Integer) return Integer
is ...
```

Adibideak (gorputzak)

```
function letra_da (Kar : in Character) return Boolean is
    Emaitza : Boolean := False;
begin
    if (Kar >= 'A' and Kar <= 'Z') or (Kar >= 'a' and Kar <= 'z') then
        Emaitza := True;
    end if;
    return Emaitza;
end letra_da;
```

```
Dei adibidea:
        get(Iniziala);
        if letra_da(Iniziala) then
...
```

```
function balio_absolutua (N : in Integer) return Integer is
    Emaitza : Integer := N;
begin
    if (Emaitza < 0) then
        Emaitza := Emaitza * -1;
    end if;
    return Emaitza;
end balio_absolutua;
```

```
Dei adibidea:
        N := ...;
        B:= balio_absolutua(N);
        put(balio_absolutua(N-1));
```

Garrantzitsua

- Funtzioen parametroak sarrerakoak dira (*in*)
  - Beraz, konstanteak bezelakoak dira, eta ezin zaie balioa aldatu

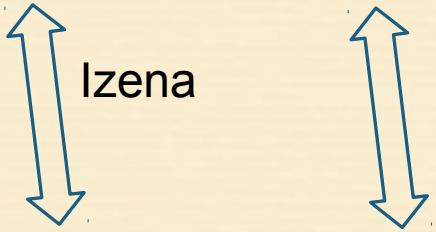
```
function proba(Zenbakia: in Integer) return Boolean is

    Lag : Integer := Zenbakia;

begin

    Lag := Lag - 1;
    ...
    Zenbakia := Zenbakia - 1; -- ERROREA!!!

            ...
```

Prozedurak

- Ekintza oso baten parekoak dira
  - Emaitza bat, batzuk edo bat ere ez itzul ditzake
  - Sarrerako, irteerako edota sarrera-irteerako parametroak izan ditzake
  - Ez dute inongo *return* ekintzarik gauzatzen
    - Zerbait itzuliz gero, irteerako parametroen bidez egiten dute

Adibideak (burukoak)

```
procedure agurra_inprimatu () is ...
```




```
procedure zatiketa_moztua (Zatikizuna, Zatitzailea : in Integer;
                           Zatiketa, Hondarra: out Integer) is ...
```

```
Dei adibideak:

    agurra_inprimatu();
    N1 : = .... ;
    zatiketa_moztua(N1,2,Ema,Hondarra);
    asteriskoak_idatzi(Ema);
```

Adibidea: bi zenbaki ordenatu

```
procedure ordenatu() is
    Zenbaki1, Zenbaki2: Integer;
begin
    put("Zenbaki osoko bat idatzi: ");
    get(Zenbaki1);
    put("Beste zenbaki osoko bat idatzi: ");
    get(Zenbaki2);
    ordenatu1(Zenbaki1,Zenbaki2);
    if balioztatu(Zenbaki1,Zenbaki2) then
        put(Zenbaki1);
        put(Zenbaki2);
    else
        ordenatu2(Zenbaki1,Zenbaki2);
        if balioztatu(Zenbaki1,Zenbaki2) then
            put(Zenbaki1);
            put(Zenbaki2);
        else
            put("Ezin izan dira zenbakiak ordenatu");
        end if;
    end if;
end ordenatu;
```

Adibidea: bi zenbaki ordenatu

```
procedure ordenatu1(N1,N2: in out Integer) is
    -- Aurre: 2 zenbaki osoko
    -- Post: zenbaki osokoak ordenatzen ditu (mota 1)
begin
    put("ORDENATU1");
    new_line;
    if N2 < N1 then
        N1 := N1 + N2;
        N2 := N1 - N2;
        N1 := N1 - N2;
    end if;
end ordenatu1;
```

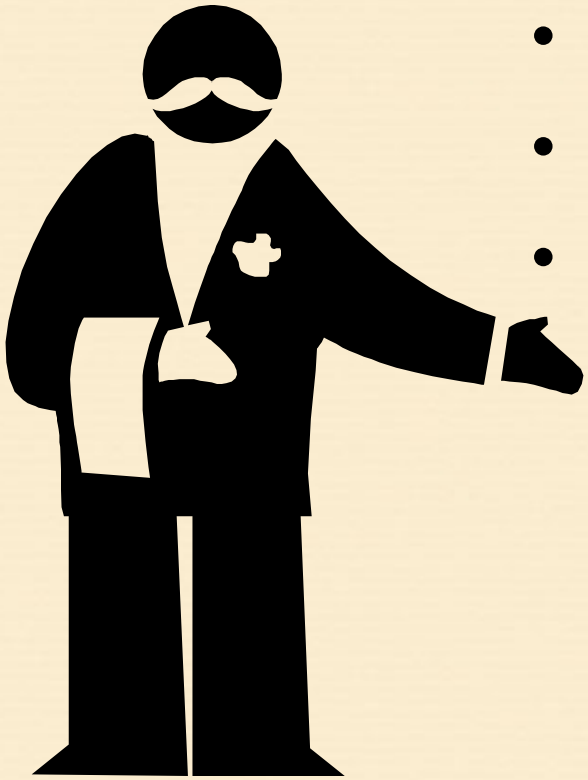
Adibidea: bi zenbaki ordenatu

```
procedure ordenatu2(N1,N2: in out Integer) is
    -- Aurre: 2 zenbaki osoko
    -- Post: zenbaki osokoak ordenatzen ditu (mota 2)
    Lag : Integer;
begin
    put("ORDENATU2");
    new_line;
    if N2 < N1 then
        Lag := N1;
        N1 := N2;
        N2 := Lag;
    end if;
end ordenatu2;
```

Adibidea: bi zenbaki ordenatu

```
function balioztatu(N1,N2: in Integer) return Boolean is
    -- Aurre: 2 zenbaki osoko
    -- Post: True itzultzen du N1 <= N2 bada, bestela False

    Emaitza : Boolean := True;
begin
    if N1 > N2 then
        Emaitza := False;
    end if;
    return Emaitza;
end ordenatu1;
```

Aurkibidea



- Gaiaren helburuak
- Motibazioa
- Kontzeptu orokorrak
- Azpiprogramak Adaz
- **Azpiprogramak Pythonez**
- Ohitura onak

Parametro motak Pythonez

- Datu motak parametro mota zehazten du
  - Aldaezina bada (osokoa, erreala, karakterea, string,...), orduan parametroa sarrerakoa izango da
  - Aldakorra bada (zerrenda,...), orduan sarrerakoa (ez bada aldatzen) edo sarrera-irteerakoa (balioa aldatzen bazaio) izango da

Azpiprogramak Pythonez

- Pythonen ez dira funtzioak eta prozedurak esplizituki ezberdintzen
- Burukoan ez da parametroen mota adierazten, ez eta itzulitako datuarena
  - Azpiprograma batek ez du zertan ***return*** ekintza izan
  - *return* batek emaitza bat baino gehiago itzul dezake

Adibidea: bi zenbaki ordenatu

```
def ordenatu1(n1,n2):
    print("ORDENATU1:")
    if n2 < n1:
        n1 = n1 + n2
        n2 = n1 - n2
        n1 = n1 - n2
    return n1,n2
```

```
def ordenatu2(n1,n2):
    print("ORDENATU2:")
    if n2 < n1:
        lag = n1
        n1 = n2
        n2 = lag
    return n1,n2
```

Adibidea: bi zenbaki ordenatu

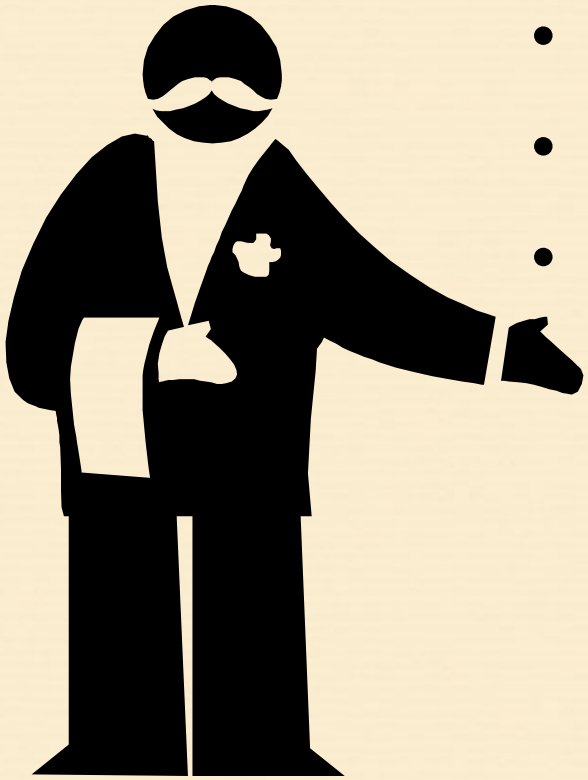
```
def balioztatu(n1,n2):
    ## Aurre: 2 zenbaki osoko
    ## Post: True itzultzen du n1 <= n2 bada, bestela False
    emaitza = True
    if n1 > n2:
        emaitza = False
    return emaitza
```

Adibidea: bi zenbaki ordenatu

```
def ordenatu():
    zenbaki1 = int(input("Zenbaki osoko bat idatzi: "))
    zenbaki2 = int(input("Beste zenbaki osoko bat idatzi: "))
    zenbaki1,zenbaki2 = ordenatu1(zenbaki1,zenbaki2)
    if balioztatu(zenbaki1,zenbaki2):
        print(Zenbaki1)
        print(Zenbaki2)
    else:
        zenbaki1,zenbaki2 = ordenatu2(zenbaki1,zenbaki2)
        if balioztatu(zenbaki1,zenbaki2):
            print(zenbaki1)
            print(zenbaki2)
        else:
            print("Ezin izan dira zenbakiak ordenatu")
```

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- Kontzeptu orokorrak
- Azpiprogramak Adaz
- Azpiprogramak Pythonez
- **Ohitura onak**



eman ta zabal zazu

Izendatzea

- Aldagaiek eta azpiprogramek izen egokiak badituzte, inplementazioa *testu moduan* irakurriko da
  - Aldagaien erabilera deskribatzen duten izenak erabili
  - Azpiprograma baten helburua sinplea bada, izen egokia ematea erraza izango da

```
...
zenbaki_positiboa_eskatu(adina);
if adina > 18 then
 ...
```

Espezifikazioa

- Garrantzitsua da azpiprogramak espezifikatzea eta anbiguotasuna ekiditea
  - Gainera, gomendagarria da oharrak erabiltzea
- Erraza da azpiprograma baten bertsioa beste batengatik ordezkatzea, espezifikazioa aldatzen ez den bitartean
  - Bestela, beste programen kodea aldatu beharko litzateke

return ekintza

- Adako funtzioetan eta Pythoneko azpiprograma batzuk, momenturen batean *return* egikaritzea eskatzen dute
  - *return* ekintza bat baino gehiago egon daiteke, baina bide guztiek *return* ekintza batera eraman behar dute
  - Adierazpena emaitzaren mota berdinekoa izan behar da (Ada)
  - *return* ekintza ta geroko kodea ez da inoiz egikarituko

return ekintza

- Ohitura ona: ***return* bakarra** erabiliko da, eta azpiprogramaren azken ekintza izango da
  - Askoz argiagoa da, eta arazketa errazten du. Are gehiago, azpiprogramen kopurua eta tamaina handia denean

Parametroak aldatzea

- Adak argi uzten du ze parametro alda daitezkeen (*out, in out)* eta zeintzuk ez (*in)*
- Pythonen, parametroak aldatzeak espero ez diren emaitzak sor ditzake
  - Ezagutza *aurreratuak* behar dira (punteroak, ...)
  - Ondorioz, ikasgai honetan, aldatu beharrean, hainbat elementu itzuliko ditugu

```
zenbaki1,zenbaki2=ordenatu(zenbaki1,zenbaki2)
```

Begizta habiaratuak

- Orokorrean, ez da ohitura ona begizta bat bestearen barruan jartzea
  - Gehienetan, barruko begizta azpiprograma bati deia bezala ikus liteke
  - Salbuespenen artean, matrizeen korritzea dago