Funtzioak

Funtzioak programen barruan dauden programa txikiak dira. Adibidez, funtzio honek pantailan agur bat besterik ez du ateratzen:

```
def agurra ():
  print "Kaixo"
```

Ikus daitekeenez, funtzio bat definitzeko `def`, hitza erabiltzen da, eta, ondoren, funtzioaren izena `agurra` eta `()` parametroen zerrenda, kasu honetan hutsik dagoena. Funtzioaren barruan, nahi ditugun aginduak jar ditzakegu.

Eta Pythonen estiloarekin jarraituz, funtzioaren barruan dagoen kodearen aurretik tabulazioa edo espazioak daudela kontutan hartu behar da.

Funtzio hori definitu ondoren, funtzio hori erabiltzen (edo deitzen) dugun bakoitzean, funtzio horretan dagoen kodea exekutatuko da:

```
agurra()
```

Horrek erakutsiko du:

```
Kaixo
```

4.0 Ariketa

Idatzi programa bat hiru funtzioekin: `egunak`, `arratsaldeak` eta `gauak`. Bakoitzak agur desberdina egin behar du: `"Egun on"`, `"Arratsalde on"` eta `"Gabon"`, hurrenez hurren. Gehitu hiru funtzioetarako deiak.

```
def egunak ():
    print("Egun on")

def arratsaldeak ():
    print("Arratsalde on")

def gauekoak ():
    print("Gabon")

egunak()
arratsaldeak()
gauekoak()
```

Emaitza:

```
Egun on
Arratsalde on
Gabon
```

Parametroak

Funtzioek parametroak jaso ditzakete. Horiek aldagai bihurtzen dira funtzioaren barruan, eta funtzioak gertatzen zaienaren arabera gauza desberdinak egitea ahalbidetzen digute.

Adibidez, norbait agurtzen duen funtzio bat sor dezakegu:

```
def agurtu (pertsona):
  print ("Kaixo", pertsona)
```

Funtzioaren barruan, pertsonen balioa desberdina izango da deian ematen zaionaren arabera. Honela erabil liteke:

```
agurtu("Neko") #pertsona aldagaia  "Neko" izango da
agurtu("Ada") #pertsona aldagaia "Ada" izango da
```

Hona hemen horren emaitza:

```
Kaixo Neko
Kaixo Ada
```

4.1 Ariketa

Idatz ezazu programa bat `karratua` izeneko funtzio batekin, parametro bat izango duena. Funtzioak parametroa bere buruarekin biderkatu (karratua) eta pantaila bakoitzean erakutsi behar du.

```
def karratua(a):
    print(a * a)

karratua(3)
```

Emaitza:

```
9
```

Lehenetsitako balioa duen parametroa

Funtzio baten parametroek lehenetsitako balioa izan dezakete. Hau da, balio jakin bat esleitu ahal zaie, eta deian parametro hori pasatzen ez bada, parametroak balio hori lehenetsiko du.

```
def agurtu(pertsona, aldiz = 3):
  for i in range (aldiz):
    print ("Kaixo", pertsona)
```

Honela deitzen bazaio:

```
agurtu("Neko", 2)
```

Pantailan ikusiko litzateke:

```
Kaixo Neko
Kaixo Neko
```

Aldiz, bigarren parametroa pasatzen ez badugu, `aldiz`ek lehenetsitako balioa (`3`) hartuko du:

```
agurtu("Bug")
```

Hau ikusiko litzateke:

```
Kaixo Bug
Kaixo Bug
Kaixo Bug
```

Parametro infinituak

Python-en funtzioei esker, parametro kopuru zehaztugabea pasatzeko aukera daukazu. Arraro samarra dirudi, baina baliagarria izan daiteke. Imajinatu funtzio bat sortu nahi duzula, pasatzen dizkiozun parametro guztiak batuko dituena:

```
def batu (*balioak):
  emaitza = 0

  for balioa in balioak:
    emaitza = emaitza + balioa

  return emaitza
```

Erreparatzen baduzu, `balioak` parametroa zehaztu dugu, aurretik `*` bat jarrita. Horrekin esan nahi dugu ez dela parametro bakarra, edozein luzera izan dezakeen zerrenda baizik.

Beraz, funtzio horri honela deitu al diogu:

```
batu (3, 4, 5) # 12
batu (2, 45) # 47
```

4.2 Ariketa

Idatz ezazu programa bat parametro mugagabeak edo dinamikoak jasotzen dituen `hitzak_batu` izeneko funtzio batekin. Hitzak batzeko funtzioak parametroak hartu behar ditu, eta horiekin guztiekin testu bat itzuli, esaldi bat osatuz.

```
def hitzak_batu(*hitza):
    esaldia = ""

    for h in hitza:
        esaldia = esaldia + " " + h

    return esaldia


print(hitzak_batu("kaixo", "zer", "moduz"))
```

Emaitza:

```
Kaixo zer moduz
```

Balioak itzultzen: `return`

Funtzioek beharrezkoak diren eragiketa guztiak egin ditzakete, baina emaitza bat itzultzen dutenean erabilgarriagoak dira. Horretarako, `return` agindua erabiltzen da, eta horrek datu jakin bat edo datu egitura bat baino ezin du itzuli.

Adibidez, bi balioren batura kalkulatzen duen funtzio bat:

```
def batu(a, b):
  emaitza = a + b
  return emaitza
```

Pixka bat laburtu daiteke zuzenean:

```
def batu(a, b):
  return a + b
```

Honela erabil liteke funtzioa:

```
print(batu(40, 2)) # 42
```

Gauza pare bat hartu behar dituzu kontuan:

1- Behin `return` egiten denean, programa funtziotik ateratzen da. 2- Funtzioan `return` bat baino gehiago izan dezakezu, baina horietako bat bakarrik exekutatuko da.

4.3 Ariketa

Idatzi programa bat `zatitu` izeneko funtzio batekin, eta jaso itzazu parametro hauek: `a` eta `b`. Funtzioak bi zenbakien arteko zatiketa itzuli behar du.

```
def zatitu (a, b):
  return a/b

print(zatitu(32, 4))
```

Emaitza:

```
8.0
```

Beste adibide bat, balio bat hainbat aldiz biderkatzen duen funtzio bat. Kopurua 1 baino txikiagoa bada, 0 bat itzuliko du:

```
def berredura (kopurua, aldiz):
  if (aldiz > 0):
    guztira = 1
    for i in range (aldiz):
      guztira = guztira * kopurua
    return guztira
  else:
    return 0

berredura (2, 3) # 8
```

Oharra: aurreko funtzioa argiago geratzen da horrela.

```
def berredura (kopurua, aldiz):
  if (batzuetan < 1): return 0

  guztira = 1
  for i in range (batzuetan):
    guztira = guztira * kopurua
  return guztira # TODO CHECK
```

Dei kabituak

Ez izan funtzio dei kabiatuak egiteko beldurrik. Oso gauza naturala da programazioan. Imajinatu erabiltzaileari zenbaki bat eskatzeaz arduratzen den funtzio hau dugula eta sartutako zenbaki itzultzen duela:

```
def irakur_zenbakia ():
  zenbakia = input("Idatzi zenbakia:")
  return int(zenbakia)
```

Demagun beste funtzio bat ere badugula zenbakiak kentzeko:

```
def kenketa (a, b):
  return a - b
```

Erabiltzaileari bi zenbaki eskatu eta kentzen dizkion programa bat egin nahi badugu, horrela egin genezake.

```
print (kenketa(irakur_zenbakia(), irakur_zenbakia()))
```

Zenbaki edo aldagai bat parametro gisa jarri beharrean, dei bat jar diezaiokegu funtzio bati.

Funtzio horiek `irakur_zenbakia()`, balio batzuk itzuliko dituzte. Lehenengo `irakur_zenbakia()` deitu bezain laster, honela izango da:

```
print (kenketa(5, irakur_zenbakia()))
```

Eta gero, bigarrenari `irakur_zenbakia()` deitu ondoren:

```
print (kenketa(5, 2))
```

Gero, kenketa egingo da eta balio bat itzuliko du:

```
print(3)
```

Eta, azkenik, `print` deituko da eta emaitza ikusiko dugu:

```
3
```

4.4 Ariketa

Idatzi bi funtzio. zenbaki bat jaso eta zenbaki bera positiboan itzultzen duena.

`positibo(balioa)`

Eta bi zenbaki jaso eta horien arteko potentzia kalkulatzen duen beste funtzio bat.

`berredura(balio1, balio2)`

Bigarren funtzioari deitu behar diozu, parametro hauek kontuan hartuta:

`berredura (positiboa (-5), positiboa (4))`

Oharra: ez erabili aurretik zeuden funtzioak.

```
def positibo (balioa):
    if balioa < 0:
        return -balioa

    return balioa


def berredura (balioa1, balioa2):
    return balioa1 ** balioa2

print(positibo(-1))
print(berredura(2, 3))
print(berredura(positibo(2), positibo(4)))
berredura(positibo(-5), positibo(4))
```

Emaitza:

```
1
8
16
```

Zergatik erabili funtzioak?

Zertarako sortu behar ditugu funtzioak? Bada, egia esan... funtsezkoak dira.

Orain arte, agindu-sekuentzia soil bat diren programak ikusi ditugu, hasieratik bukaeraraino exekutatzen direnak.

Hori ondo dago programak sinpleak direnean eta gauza gutxi egin behar dituztenean, baina programak zerbait konplexuagoa egin behar duenean, litekeena da funtzioak erabili behar izatea, hainbat arrazoirengatik.

0 arrazoia: zatitu eta irabaziko duzu

Programak arazoak konpontzen saiatzen dira, soluzio bat eskainiz. Batzuetan arazoak oso konplexuak izan daitezke. Funtzioek arazo horiei zatika heltzeko aukera ematen dizute. Funtzio bakoitzak arazoaren zati baterako konponbidea eman diezazuke. Beraz, arazo handia arazo txiki askotan bana dezakezu eta arazo bakoitza funtzio batekin konpon dezakezu. Programak funtzioetan banatzea programa konplexuagoak diseinatzeko lehen urratsa da.

1. arrazoia: kodea ez errepikatzea

Zure programak zerbait egin behar badu hainbat aldiz, kodea behar beste aldiz errepikatu beharko zenuke. Imajinatu erabiltzailearen hainbat datu jaso behar dituzula, eta hori egiten duzun bakoitzean datua hutsik ez dagoela egiaztatu behar duzula:

```
datua = ""

while datua == "":
  datua = input("Mesedez, sartu datu bat:")
  if datua == "":
    print ("Datua hutsik dago!")
```

Erabiltzaileari 3 datu eskatzen badizkiozu, kode hori 3 aldiz errepikatu beharko zenuke! Aldiz, kode bera duen funtzio bat sortzen baduzu, behin bakarrik idatzi beharko duzu, eta gero behar adina aldiz erabili ahal izango duzu.

Oharra: kodea ez errepikatzea da programatzaile on orok jarraitu beharreko araurik garrantzitsuenetako bat. **"3aren araua"** ere aplika dezakezu. Zerbait errepikatu behar duzun hirugarren aldian, automatizatu edo funtzio biurtu egin behar duzu.

2. arrazoia: berrerabili kodea

Kodea ez errepikatzeaz gain, funtzio batek aukera ematen digu kode berak hainbat datu-motatarako balio dezan. Horretarako erabiltzen dira parametroak!

```
def agurtu (agurra, batzuetan):
  for i in range (batzuetan):
    print (agurra)
```

Balio desberdinekin dei dakioke:

```
agurtu ("Holi", 3)
agurtu ("Aupa", 1)
```

Hau litzateke:

```
Holi
Holi
Holi
Aupa
```

3. arrazoia: mantentze lanak erraztea

Kodea leku bakar batean badago, errazagoa da zuzentzea, aldatzea, hobetzea eta orokorrean mantentzea. Programa bat zerotik sortzea oso polita da, baina programatzaileen benetako lana kodea denboran zehar mantentzea da. Gure funtzioak ondo diseinatuak baditugu, lana PIIIIILA aurreztuko dugu.

4. arrazoia: testak egiteko aukera ematen du

Agian gazte samarra zara honetarako, baina profesional onek programak probatzen dituzte. Zer esan nahi du horrek? Beraien programek behar dutena egiten dutela egiaztatzea helburu bakarra duten programak idazten dituztela. Zure kodeak funtzioak baditu, testak idatzi ahal izango dituzu funtzio horiek behar dutena egiten dutela egiaztatzeko.

Berez, aditua zarenean, zure eginkizuna bera baino lehenago... testa idaztea da zure betebeharra!

Nola egin funtzio onak

Edonork idatz ditzake funtzioak eta kodea zati txikitan antolatu. Baina *pro* gisa funtzioak idatzi nahi badituzu, honako hau lortu behar duzu:

- Funtzio batek gauza bakarra egin behar du. Hobe da funtzio txiki asko izatea, gauza asko egiten dituzten funtzio gutxi izatea baino. Zure funtzioa pantailan sartzen ez bada edo 24 lerro gainditzen baditu, agian zati txikitan banatu beharko duzu.
- Funtzio batek ez luke bere kanpoan dagoen ezer aldatu behar. Ezustekorik izan nahi ez baduzu, funtzio batek ez luke programaren barruan ezustekorik eragin behar.
- Funtzio batek zerbait itzuli beharko luke, eta horrek beti berdina izan beharko luke parametro jakin batzuetarako.

Proposatutako ariketak

4.0 Ariketa

Idatzi bi zenbaki jasotzen dituen funtzio bat, detektatu zein den handiena eta erakutsi horien arteko aldea.

```
def diferentzia (balio1, balio2):
    kenketa = 0
```

```
    if balio1 > balio2:
        kenketa = balio1 - balio2
    else:
        kenketa = balio2 - balio1

    print("Aldea hau da: ", kenketa)

diferentzia(10, 5)
diferentzia(4, 12)
```

Emaitza:

```
Aldea hau da: 5
Aldea hau da: 8
```

4.1 Ariketa

Idatzi bi zenbaki eta eragiketa zeinu aritmetiko bat jasotzen dituen programa bat: +, −,∗, /. Funtzioak eragiketa horren emaitza itzuli behar du bi zenbakien artean.

```
def kalkulatu (balioa1, balioa2, eragiketa):
    if eragiketa ==  "+": return balioa1 + balioa2
    elif eragiketa == "-": return balioa1 - balioa2
    elif eragiketa == "*": return balioa1 * balioa2
    elif eragiketa == "/": return balioa1 / balioa2
    else: return "Eragiketa ezezaguna"

emaitza = kalkulatu(4, 6, "*")
print(emaitza)

emaitza = kalkulatu(5, 5, "-")
print(emaitza)

emaitza = kalkulatu(4, 3, "@")
print(emaitza)
```

Emaitza:

```
24
0
Eragiketa ezezaguna
```

4.2 Ariketa

```
def agurtu (momentua)
```

Funtzio honek parametro gisa eguneko une bat hartzen du: `"goizean"`, `"arratsaldean"` edo `"gauean"`, eta dagokion agurra itzuli behar du: `"Egun on"`, `"Arratsalde on"`, eta `"Gabon"`, hurrenez hurren.

```
def agurtu(momentua):
    if momentua == "goizean":
        return "Egun on"
    elif momentua == "arratsaldean":
        return "Arratsalde on"
    elif momentua == "gauean":
        return "Gabon"
    else:
        return ""

print(agurtu("arratsaldean"))

# beste bertsio bat
def agurtu2(momentua):
    return {
        "goizean": "Egun on",
        "arratsaldean": "Arratsalde on",
        "gauero": "Gabon"
    }[momentua]

print(agurtu2("arratsaldean"))
```

Emaitza:

```
Arratsalde on.
Arratsalde on.
```

4.3 Ariketa

`hasiZenbakiarekin (luzera, zenbakia)`

Funtzio honek array bat sortu behar du, bi parametroak kontutan hartuz: `luzera` zenbaki kopuruarekin, eta zenbaki guztiak `zenbakia` balioarekin bete behar dira.

```
def hasieratuZenbakiarekin(luzera, zenbakia):
    zenbakiak = []
    for i in range(luzera):
        zenbakiak.append(zenbakia)
    return zenbakiak

def hasieratuZenbakiarekin2(luzera, zenbakia): return [zenbakia] * luzera

print(hasieratuZenbakiarekin(10, 3))
print(hasieratuZenbakiarekin(10, 3))
```

Emaitza:

```
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

4.4 Ariketa

`def ausazkoa (max)`

Funtzio horrek `0` eta parametro gisa pasatzen den gehieneko balioaren arteko ausazko (random) zenbaki bat itzuli behar du.

```
import random

def ausazkoa (max):
    return random.randint(0, max)

print(ausazkoa(5))
```

Emaitza:

```
3
```

4.5 Ariketa

`izenaSortu(silabak)`

Silaba kopuru bat dela eta, ausazko izen bat sortzen duen funtzio bat idatzi (kontsonanteak eta bokalak txandakatu). Aurreko ariketako funtzioa erabil dezakezu.

```
import random

def ausazkoa(max):
    return random.randint(0, max - 1)

def izenaSortu(silabak):
    oinak = ["a", "e", "i", "o", "u"]
    konsonanteak =
["b","c","d","f","g","h","j","k","l","m","n","ñ","p","q","r","s","t","v","
w","x","y","z"]
    izena = ""

    for i in range(silabak):
        konsonantea = konsonanteak[ausazkoa(len(konsonanteak))]
        oina = oinak[ausazkoa(len(oinak))]
        izena = izena + konsonantea + oina
```

```
    return izena

print(izenaSortu(3))
```

Emaitza:

```
xamozu
```

4.6 Ariketa

`def pasahitzaSortu(luzera)`

Luzera bat emanda, zorizko karaktereak dituen string bat sortzen duen funtzioa. Karakteredun *string* array bat erabil dezakezu, eta ausazko karaktereak atera arraytik, izen bat sortzeko. Ausazko zenbakiak sortzeko `random.randint` funtzioa erabil dezakezu.

```
import random

def ausazkoa (max):
    return random.randint(0, max - 1)

def pasahitzaSortu (luzera):
    hizkiak = ["a","b","c","d","e","f","g","h","i","j","k","l",
        "m","n","ñ","o","p","q","r","s","t","u","v","w","x","y","z",
        "0","1","2","3","4","5","6","7","8","9",".","-","_","!","$"]
    pasahitza = ""

    for i in range(luzera):
        hizkia = hizkiak[ausazkoa(len(hizkiak))]
        pasahitza = pasahitza + hizkia

    return pasahitza


print(pasahitzaSortu(8))
```

Emaitza:

```
_! 5_flg $
```

4.7 Ariketa

Sortu `faktura` izeneko funtzioa (produktuak, kantitateak, prezioak), tamaina bereko hiru array jasotzen dituena, honako eduki hauekin:

1. `produktuak`: produktuen izenak.
2. `kantitateak`: zenbaki osoak, kantitatea adierazteko.
3. `prezioak`: zenbaki hamartarrak, produktu bakoitzaren prezioa adierazteko.

Funtzioak produktu bakoitza aztertu eta prezio osoa kalkulatu behar du, kantitatearen eta prezioaren arabera. Produktu bakoitzaren prezio osoa erakutsi behar da, eta programaren amaieran, batura osoaren prezioa erakutsi behar da.

```
def faktura(produktuak, kantitateak, prezioak):
    faktura = "FAKTURA\n--------------------\n"
    totala = 0

    for i in range(len(produktuak)):
        faktura = faktura + produktuak[i]
        faktura = faktura + " x " + str(kantitateak[i])
        faktura = faktura + " : " + str(prezioak[i]) + "\n"

        totala = totala + (kantitateak[i] * prezioak[i])

    faktura = faktura + "\n----------------------\n"
    faktura = faktura + "Totala: " + str(totala)

    return faktura


# Adibidea
fakturaGuztira = faktura(["Ogia","Arraultza","Irina"],[1,6,2],[1.2, 0.2,
0.8])
print(fakturaGuztira)
```

Emaitza:

```
FAKTURA
- ------------------
Ogia x 1: 1.2
Arraultzak x 6: 0.2
Irina x 2: 0.8

- ----------------------
Guztira: 4.0
```

4.8 ariketa

Hau zailagoa izango da. RPG jokuetako pertsonaiak sortzeko funtzio bat duen programa bat sortzen du.

`def atributuakSortu (mailaKonpentsazioa)`

Funtzio horrek hiru aldagai definitu behar ditu: `indarra`, `abiadura` eta `adimena`. Programak `20` puntu banatu behar ditu hiru aldagaien artean. Bestela esanda, hiru aldagaien artean `20` batu behar dira.

`mailaKonpentsazioa` parametroak balio behar du puntu oso bereiziak edo berdinduak banatzen diren adierazteko, balio desorekatuena zenbat eta handiagoa izan, hau da, atributuen arteko aldea handiagoa da; nola egin programatzailearen kontua da.

Azkenean, programak esleitutako puntuen laburpena erakutsi behar du.

```
import random


def ausazkoa(maximoa):
  return random.randint(0, maximoa)


def atributuakSortu(konpentsaketaMaila):
  puntuakEman = 0
  adimena = 0
  indarra = 0
  abiadura = 0

  gainontzekoPuntuak = 20
  puntuak = 0

  while gainontzekoPuntuak > 0:
    if konpentsaketaMaila > gainontzekoPuntuak:
      puntuak = gainontzekoPuntuak
      gainontzekoPuntuak = 0
    else:
      puntuak = ausazkoa(konpentsaketaMaila + 1)
      gainontzekoPuntuak = gainontzekoPuntuak - puntuak

    puntuakEman = ausazkoa(3)

    if puntuakEman == 0:
      adimena = adimena + puntuak
    elif puntuakEman == 1:
      indarra = indarra + puntuak
    elif puntuakEman == 2:
      abiadura = abiadura + puntuak

  print("\nBalioak konpentsazioaren arabera adierazita: ",
konpentsaketaMaila)
  print("Adimena: ", adimena)
  print("Indarra: ", indarra)
  print("Abiadura: ", abiadura)


atributuakSortu(3)
```

Emaitza:

```
Balioak konpentsazioaren arabera adierazita:  3
Adimena:  3
Indarra:  8
Abiadura:  5
```