Pràctica 1: Variables

September 14, 2021

1 Sintaxi

1.1 Definició de variables i primeres instruccions

Podem definir variables amb el signe **=** (igual), sense necessitat de declarar-ne el tipus abans.

[1]:
```
r=3
```

[2]:
```
r
```

[2]:
```
3
```

Per separar diferents assignacions (o instruccions), podem fer-ho mitjançant la tecla **Intro** o bé el **;** (punt i coma). Si volem veure el valor l'una variable, podem escriure el nom de la variable com a instrucció a executar. Si dins de la mateixa entrada volem veure el valor de més d'una variable, podem utilitzar la instrucció `print()`.

[3]:
```
pi=3.14159; e=2.71828
```

[4]:
```
pi; e
```

[4]:
```
2.71828
```

[5]:
```
print(pi)
print(e)
```

```
3.14159
2.71828
```

Les operacions bàsiques que hem vist al document anterior també es poden fer utilitzant variables enlloc de números.

[6]:
```
l=2*pi*r
print("La longitud és",l)
```

```
La longitud és 18.849539999999998
```

[7]:
```
a=pi*r**2
print("L'àrea és",a)
```

```
L'àrea és 28.27431
```

Una variable també pot contenir una cadena de caràcters, i s'accepten operacions amb la cadena (suma i multiplicació per enters):

[8]:
```
s="Hola món"
```

[9]:
```
print(s)
```

```
Hola món
```

[10]:
```
t=s+s
print(t)
```

```
Hola mónHola món
```

[11]:
```
u=r*s
print(u)
```

```
Hola mónHola mónHola món
```

[12]:
```
u=pi*s
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-12-c2303551e61d> in <module>
----> 1 u=pi*s


TypeError: can't multiply sequence by non-int of type 'float'
```

Amb un sol símbol igual podem assignar més d'una variable. Primer se substitueix els valors de la dreta del símbol = i després es fa l'assignació:

[16]:
```
a,b,c=2.71828,"Hola",3
```

[17]:
```
print("a=",a)
print("b=",b)
print("c=",c)
```

```
a= 2.71828
b= Hola
c= 3
```

La funció **type()** ens diu de quin tipus és cada variable:

[18]:
```
print(type(a))
print(type(b))
print(type(c))
```

```
<class 'float'>
<class 'str'>
<class 'int'>
```

1.2 Llistes

Un tipus de variable molt important és el tipus **llista**: una **llista** és una successió de valors (que poden ser nombres, cadenes, altres llistes, ...) entre claudàtors **[...]** separats per coma:

[19]:
```
l1=[1,2,3,4,5]
```

[20]:
```
print(l1)
```

```
[1, 2, 3, 4, 5]
```

Podem accedir a cada possició mitjançant claudàtors, començant per la posició **0**. L'accés pot ser per a obtenir informació o bé per a fixar-ne el valor.

[21]:
```
print(l1[0])
```

```
1
```

[22]:
```
print(l1[3])
```

```
4
```

[23]:
```
print(l1[2:4])
```

```
[3, 4]
```

[24]:
```
l1[2]=99
```

[25]:
```
print(l1)
```

```
[1, 2, 99, 4, 5]
```

1.3 Tuples

Les **tuples** són un tipus de llistes que no poden ser modificats. La sintaxi és entre **(...)** (parèntesis) i separant els valors per **,** (coma). Un cop defida la tupla, l'accés als valors és com a una llista.

[26]:
```
tup1=(1,2,3,4,5)
tup2=(100,)
```

[27]:
```
print(tup2)
```

```
(100,)
```

[28]:
```
print(tup1[2:4])
```

```
(3, 4)
```

1.4 Conjunts

Un conjunt és una col·lecció d'elements separats per coma on no hi ha ordre ni repeticions:

[29]:
```
con1={3,2,4,6,1,2}
print(con1)
```

```
{1, 2, 3, 4, 6}
```

1.5 Diccionaris

Un diccionari és un conjunt format per entrades on hi ha etiquetes i el seu valor. S'utitlitza el **:** (dos punts) per separar l'etiqueta de la definició (que poden ser nombres, caràcters o llistes). Si volem accedir a un valor concret, utilitzem la mateixa sintaxis que si volem accedir a una posició d'una llista, amb l'etiqueta enlloc de la posició.

[30]:
```
mesos={'Gener':'January','Febrer':'February','Març':'March','Abril':'April',
       'Maig':'May','Juny':'June','Juliol':'July','Agost':'August',
       'Setembre':'September','Octubre':'October','Novembre':'November',
      'Desembre':'December'}
```

[31]:
```
type(mesos)
```

[31]:
```
dict
```

[32]:
```
print(mesos['Juny'])
```

```
June
```

[33]:
```
print(mesos)
```

```
{'Gener': 'January', 'Febrer': 'February', 'Març': 'March', 'Abril': 'April',
'Maig': 'May', 'Juny': 'June', 'Juliol': 'July', 'Agost': 'August', 'Setembre':
'September', 'Octubre': 'October', 'Novembre': 'November', 'Desembre':
'December'}
```

[34]:
```
del mesos['Març'] # Esborrem l'entrada Març del diccionari
```

[35]:
```
print(mesos)
```

```
{'Gener': 'January', 'Febrer': 'February', 'Abril': 'April', 'Maig': 'May',
'Juny': 'June', 'Juliol': 'July', 'Agost': 'August', 'Setembre': 'September',
'Octubre': 'October', 'Novembre': 'November', 'Desembre': 'December'}
```

Un cas particular de diccionari és el que s'utilitza per a guardar dades. En aquest cas l'etiqueta dóna el nom de la variable, mentre que les dades (columnes) s'entren en format de llista.

[36]:
```
dades={'Comarca':['Bages','Barcelonès','Vallès Occidental','Vallès Oriental'],
       2013:[184182,2217065,894638,400982],
       2014:[182795,2197818,893038,400426],
       2015:[171193,2195271,895166,397396],
       2016:[173143,2205803,900516,399036],
       2017:[173724,2226828,908026,401820],
       2018:[174703,2239915,915486,405236],
       2019:[176891,2264301,923976,408672]}
```

[37]:
```
dades['Comarca']
```

[37]:
```
['Bages', 'Barcelonès', 'Vallès Occidental', 'Vallès Oriental']
```

[38]:
```
dades[2014]
```

[38]:
```
[182795, 2197818, 893038, 400426]
```

[39]:
```
dades.get('Comarca')
```

[39]:
```
['Bages', 'Barcelonès', 'Vallès Occidental', 'Vallès Oriental']
```

[40]:
```
dades.values()
```

[40]:
```
dict_values([['Bages', 'Barcelonès', 'Vallès Occidental', 'Vallès Oriental'],
[184182, 2217065, 894638, 400982], [182795, 2197818, 893038, 400426], [171193,
2195271, 895166, 397396], [173143, 2205803, 900516, 399036], [173724, 2226828,
908026, 401820], [174703, 2239915, 915486, 405236], [176891, 2264301, 923976,
408672]])
```

[41]:
```
dades.items()
```

[41]:
```
dict_items([('Comarca', ['Bages', 'Barcelonès', 'Vallès Occidental', 'Vallès
Oriental']), (2013, [184182, 2217065, 894638, 400982]), (2014, [182795, 2197818,
893038, 400426]), (2015, [171193, 2195271, 895166, 397396]), (2016, [173143,
2205803, 900516, 399036]), (2017, [173724, 2226828, 908026, 401820]), (2018,
[174703, 2239915, 915486, 405236]), (2019, [176891, 2264301, 923976, 408672])])
```

1.6 Manipulacions de llistes

Als exemples següents veiem diferents opcions per afegir, esborrar, ordenar, ... llistes:

[42]:
```
l=[10,2,30,4,50]
```

[43]:
```
print(l)
```

```
[10, 2, 30, 4, 50]
```

[44]:
```
ll=l # en el fons són les mateixes llistes
```

[45]:
```
print(ll)
```

```
[10, 2, 30, 4, 50]
```

[46]:
```
ll[1]=99 # Modifiquem la segona posició de ll
```

[47]:
```
print(ll)
```

```
[10, 99, 30, 4, 50]
```

[48]:
```
print(l) # També s'ha modificat l!!!
```

```
[10, 99, 30, 4, 50]
```

[49]:
```
lll=l.copy() # es fa una còpia dels valors d'una llista en una altra llista
```

[50]:
```
print(lll)
```

```
[10, 99, 30, 4, 50]
```

[51]:
```
l[1]=2
```

[52]:
```
print(lll)
```

```
[10, 99, 30, 4, 50]
```

[53]:
```
print(l)
```

```
[10, 2, 30, 4, 50]
```

[54]:
```
l.sort() # Ordena la llista l amb el mateix nom (en aquest cas, també afecta ll)
```

[55]:
```
ll
```

[55]:
```
[2, 4, 10, 30, 50]
```

[56]:
```
l.reverse() # Canvia l'ordre dels elements, de l'últim al primer sobre la llista␣
↪l
```

[57]:
```
a=l.pop() # Esborra l'últim element i el guarda a la variable a
```

[58]:
```
print(l)
```

```
[50, 30, 10, 4]
```

[59]:
```
print(a)
```

```
2
```

[60]:
```
l.append(99) # afegim un 99 com a últim element
```

[61]:
```
print(l)
```

```
[50, 30, 10, 4, 99]
```

[62]:
```
ls=["hola","món"]
```

[63]:
```
l+ls # és una manera de contatenar cadenes
```

[63]:
```
[50, 30, 10, 4, 99, 'hola', 'món']
```

[64]:
```
ls.insert(1,'a tot el') # inserta la caden 'a tot' a la posició 1
```

[65]:
```
print(ls)
```

```
['hola', 'a tot el', 'món']
```

[66]:
```
del l # Esborra la llista
```

[67]:
```
print(l)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-67-e6f469464218> in <module>
----> 1 print(l)

NameError: name 'l' is not defined
```

[68]:
```
print(ll)
```

```
[50, 30, 10, 4, 99]
```

[69]:
```
ll.clear() # Buida la llista
```

[70]:
```
print(ll)
```

```
[]
```

1.7 Manipulació de conjunts

Es poden fer manipulacions molt semblants a les de les llistes, però no podem fer referència a la posició (els conjunts no estan ordenats).

[71]:
```
p={4,6,2,8,10,2}
```

[72]:
```
s={1,3,5,7,3,11,9}
```

[73]:
```
print(p)
```

```
{2, 4, 6, 8, 10}
```

[74]:
```
p & s # Retorna els elements de comuns a p i s
```

[74]:
```
set()
```

[75]:
```
p | s # Retorna el elements que són a p o s
```

[75]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

[76]:
```
p = p | {12} # Aprofitem aquesta propietat per afegir un element
```

[77]:
```
print(p)
```

```
{2, 4, 6, 8, 10, 12}
```

[78]:
```
p.remove(4) # Esborra un element al conjunt p
```

[79]:
```
print(p)
```

```
{2, 6, 8, 10, 12}
```

[80]:
```
p=p|{4}
```

[81]:
```
print(p)
```

```
{2, 4, 6, 8, 10, 12}
```

[82]:
```
n={1,2,3,4,5,6}
```

[83]:
```
p-n # Torna els elements de p que no són a n
```

[83]:
```
{8, 10, 12}
```

[ ]:
```

```