Llenguatge Python

1. Sintaxi de Python

La sintaxi de Python és coneguda per la seva claredat i senzillesa. Python utilitza la indentació (espais o tabulacions) per definir blocs de codi, com funcions, condicions i bucles, en lloc d'usar claudàtors `{}` com altres llenguatges. A més, Python no requereix punts i comes al final de les línies, cosa que fa que el codi sigui més net i llegible.

Exemples:

- Exemple 1: Bloc de codi amb print:

```
print("Hola, món!")
    print("Aquesta línia està indentada.")
print("Això és Python!")
```

En aquest exemple, la segona línia (print("Aquesta línia està indentada.")) està indentada amb 4 espais.

Python requereix que les línies de codi estiguin correctament alineades; si una línia està incorrectament indentada com en aquest exemple, generarà un error de sintaxi quan s'intenti executar.

2. Comentaris en Python

Explicació:

Els comentaris en Python s'utilitzen per afegir anotacions al codi sense que afectin l'execució. Són essencials per fer el codi més comprensible per a altres desenvolupadors o per a tu mateix en el futur. Els comentaris s'escriuen amb el símbol **#** per a comentaris d'una sola línia, mentre que `'''` o `"""` s'utilitzen per comentaris de múltiples línies.

Exemples:

- Exemple 1: Comentari de línia única:

```
# Això és un comentari
print("Hola, món!")
```

- Exemple 2: Comentari de múltiples línies:

```
"""
Aquest és un exemple
d'un comentari
de múltiples línies.
"""
print("Hola, món!")
```

3. Variables en Python

Explicació:

Les variables en Python s'utilitzen per emmagatzemar dades. No cal declarar el tipus de variable abans d'assignar-li un valor; Python detecta el tipus automàticament en funció del valor assignat.

Exemples:

- Exemple 1: Assignació de variables:

```
nom = "Anna"
edat = 25
```

`nom` conté una cadena de text, mentre que `edat` conté un número enter.

- Exemple 2: Canvi de tipus de variable:

```
x = 5       Enter
x = "Hola"   Ara, x és una cadena de text
print(x)
```

4. Tipus de dades en Python

Explicació:

Python té diversos tipus de dades bàsics, com ara enters (`int`), números de coma flotant (`float`), cadenes (`str`), booleans (`bool`), llistes (`list`), tuples (`tuple`), conjunts (`set`), i diccionaris (`dict`).

Exemples:

- Exemple 1: Diferents tipus de dades:

```
enter = 10
flotant = 10.5
cadena = "Hola"
boolea = True
```

- Exemple 2: Tipus complexos:

```
llista = [1, 2, 3]
tupla = (1, 2, 3)
conjunt = {1, 2, 3}
diccionari = {"nom": "Anna", "edat": 25}
```

5. Nombres en Python

Explicació:

Python gestiona diferents tipus de números, incloent enters (`int`), números de coma flotant (`float`), i números complexos. Les operacions aritmètiques són senzilles d'implementar en Python.

Exemples:

- Exemple 1: Operacions aritmètiques bàsiques:

```
a = 10
b = 3
suma = a + b
divisio = a / b
print(suma)      Mostra 13
print(divisio)   Mostra 3.3333...
```

- Exemple 2: Números complexos:

```
z = 1 + 2j
print(z.real)    Mostra 1.0
print(z.imag)    Mostra 2.0
```

6. Conversió de tipus en Python (Casting)

Explicació:

El "casting" o conversió de tipus permet convertir explícitament un valor d'un tipus de dada a un altre. Això és útil quan es vol controlar el tipus de dada d'una variable en particular.

Exemples:

- Exemple 1: Conversió a enter:

```
x = int(3.8)
print(x)   Mostra 3
```

- Exemple 2: Conversió a cadena:

```
x = 5
y = str(x)
print(y)   Mostra "5"
```

7. Cadenes de text en Python (Strings)

Explicació:

Les cadenes de text en Python són un tipus de dada que emmagatzema seqüències de caràcters. Són immutables, el que significa que no poden ser canviades un cop creades. No podem canviar un caràcter específic dins d'una cadena existent.

Exemples:

- Exemple 1: Concatenació de cadenes:

```
salutacio = "Hola"
nom = "Anna"
frase = salutacio + ", " + nom
print(frase)   Mostra "Hola, Anna"
```

- Exemple 2: Accés a caràcters dins d'una cadena:

```
text = "Python"
print(text[0])   Mostra "P"
print(text[-1])  Mostra "n"
```

Índexs positius: P(0), y(1), t(2), h(3), o(4), n(5)

Índexs negatius: P(-6), y(-5), t(-4), h(-3), o(-2), n(-1)

8. Booleans en Python

Explicació:

Els booleans en Python representen valors de veritat: `True` (cert) i `False` (fals). S'utilitzen sovint en condicions per controlar el flux del programa.

Exemples:

- Exemple 1: Comparacions booleanes:

```
a = 10
b = 20
print(a > b)   Mostra False
print(a < b)   Mostra True
```

- Exemple 2: Utilització de booleans en condicions:

```
x = True
if x:
    print("X és cert")
```

9. Operadors en Python

Explicació:

Python ofereix diversos tipus d'operadors per realitzar operacions en variables i valors. Aquests inclouen operadors aritmètics, operadors de comparació, operadors lògics, i operadors d'assignació.

Exemples:

- Exemple 1: Operadors aritmètics:

```
x = 10
y = 3
suma = x + y
multiplicacio = x * y
print(suma)          Mostra 13
print(multiplicacio)  Mostra 30
```

- Exemple 2: Operadors lògics i de comparació:

```
a = 5
b = 8
print(a > b and b > 0)   Mostra False
print(a < b or b == 0)   Mostra True
```

10. Llistes en Python

Explicació:

Les llistes són una col·lecció d'elements que poden ser de diferents tipus de dades, i es poden modificar (són mutables). Es defineixen amb claudàtors `[ ]`.

Exemples:

- Exemple 1: Creació i accés a elements:

```
fruits = ["poma", "plàtan", "cirera"]
print(fruits[0])   Mostra "poma"
```

- Exemple 2: Modificar una llista:

```
fruits = ["poma", "plàtan", "cirera"]
fruits[1] = "taronja"
print(fruits)   Mostra ["poma", "taronja", "cirera"]
```

11. Python Tuples

Explicació:

Les tuples són similars a les llistes, però són immutables, és a dir, no es poden modificar després de ser creades. Es defineixen amb parèntesis `( )`.

Exemples:

- Exemple 1: Creació i accés a elements:

```
colors = ("vermell", "verd", "blau")
print(colors[1])   Mostra "verd"
```

- Exemple 2: Immutabilitat de les tuples:

```
colors = ("vermell", "verd", "blau")
colors[1] = "groc"   Això donarà un error, ja que les tuples no es poden modificar
print(colors)
```

12. Conjunts en Python (Sets)

Explicació:

Els conjunts (`sets`) són col·leccions no ordenades i sense elements duplicats. Es defineixen amb claus `{ }`.

Exemples:

- Exemple 1: Creació i accés:

```
numeros = {1, 2, 3, 4, 4, 5}
print(numeros)   Mostra {1, 2, 3, 4, 5} (el duplicat 4 es elimina automàticament)
```

- Exemple 2: Operacions de conjunts:

```
a = {1, 2, 3}
b = {3, 4, 5}
print(a.union(b))     Mostra {1, 2, 3, 4, 5}
print(a.intersection(b))   Mostra {3}
```

13. Diccionaris en Python (Dictionaries)

Explicació:

Els diccionaris (`dictionaries`) són col·leccions desordenades de parelles clau-valor, on cada clau és única. Es defineixen amb claus `{ }`.

Exemples:

- Exemple 1: Creació i accés:

```
persona = {"nom": "Anna", "edat": 25}
print(persona["nom"])   Mostra "Anna"
```

- Exemple 2: Afegir i modificar elements:

```
persona = {"nom": "Anna", "edat": 25}
persona["ciutat"] = "Barcelona"
persona["edat"] = 26
print(persona)   Mostra {"nom": "Anna", "edat": 26, "ciutat": "Barcelona"}
```

14. Condicionals Python If...Else

Explicació:

Les estructures condicionals `if`, `elif` i `else` permeten executar blocs de codi diferents segons certes condicions.

Exemples:

- Exemple 1: Condicional simple:

```
edat = 18
if edat >= 18:
    print("Ets major d'edat")
```

- Exemple 2: Condicional amb `elif` i `else`:

```
edat = 16
if edat >= 18:
    print("Ets major d'edat")
elif edat >= 13:
    print("Ets adolescent")
else:
    print("Ets un nen")
```

15. Bucles While en Python

Explicació:

Els bucles `while` s'utilitzen per repetir un bloc de codi mentre una condició és certa.

Exemples:

- Exemple 1: Bucle `while` bàsic:

```
i = 1
while i < 6:
    print(i)
    i += 1
```

- Exemple 2: Bucle amb `break`:

```
i = 1
while i < 10:
    print(i)
    if i == 5:
        break   surt completament del bucle
    i += 1
```

- Exemple 3: Bucle amb `continue`:

```
i = 0
while i < 10:
    i += 1
    if i == 5:
        continue    ens saltem el valor 5
    print(i)
```

16. Bucles For en Python

Explicació:

Els bucles `for` s'utilitzen per iterar sobre una seqüència (llistes, tuples, diccionaris, conjunts, o cadenes de text).

Exemples:

- Exemple 1: Iterar sobre una llista:

```
fruits = ["poma", "plàtan", "cirera"]
for fruit in fruits:
    print(fruit)
```

Sortida:

poma

plàtan

cirera

- Exemple 2: Ús de `range()` en un `for`:

```
for i in range(5):
    print(i)
```

Sortida:

0

1

2

3

4

17. Funcions en Python

Explicació:

Les funcions són blocs de codi que es poden reutilitzar. Es defineixen amb la paraula clau `def` i poden acceptar paràmetres i retornar valors.

Exemples:

- Exemple 1: Funció bàsica:

```
def saluda(nom):
    print("Hola, " + nom)
saluda("Anna")
```

- Exemple 2: Funció amb valor de retorn:

```
def suma(a, b):
    return a + b
resultat = suma(5, 3)
print(resultat)   Mostra 8
```

18. Funcions Lambda en Python

Explicació:

Les funcions `lambda` són funcions anònimes i petites que es poden definir en una sola línia. Es solen utilitzar per a operacions ràpides i senzilles.

Exemples:

- Exemple 1: Funció `lambda` simple:

```
suma = lambda a, b: a + b
print(suma(5, 3))   Mostra 8
```

- Exemple 2: Funció `lambda` dins d'una altra funció:

```
def multiplicador(n):
    return lambda x: x * n
doble = multiplicador(2)
print(doble(5))   Mostra 10
```

19. Arrays en Python

Explicació:

Tot i que Python no té arrays com altres llenguatges de programació, les llistes es poden utilitzar com arrays. També es poden utilitzar arrays amb la llibreria `array` o amb llibreries com `NumPy` per a treballs més avançats.

Exemples:

- Exemple 1: Ús d'una llista com array:

```
numeros = [1, 2, 3, 4, 5]
print(numeros[2])   Mostra 3
```

- Exemple 2: Array amb la llibreria `array`:

```
import array as arr
a = arr.array('i', [1, 2, 3, 4, 5])
print(a[2])   Mostra 3
```