Előadás -5

Programozás Pythonban

Instructor : Dr. AALWAHAB DHULFIQAR

Advisor : Dr. Tejfel Mate

Mit fogsz tanulni:

- Hibák, kudarcok és egyéb csapások Kivételek
- Beépített kivételek
- Procedurális vs. objektumorientált megközelítés Osztályhierarchiák
- Mi az az objektum?

PYTHON INSTITUTE
Open Education & Development Group

CISCO

Hibák, kudarcok és egyéb csapások

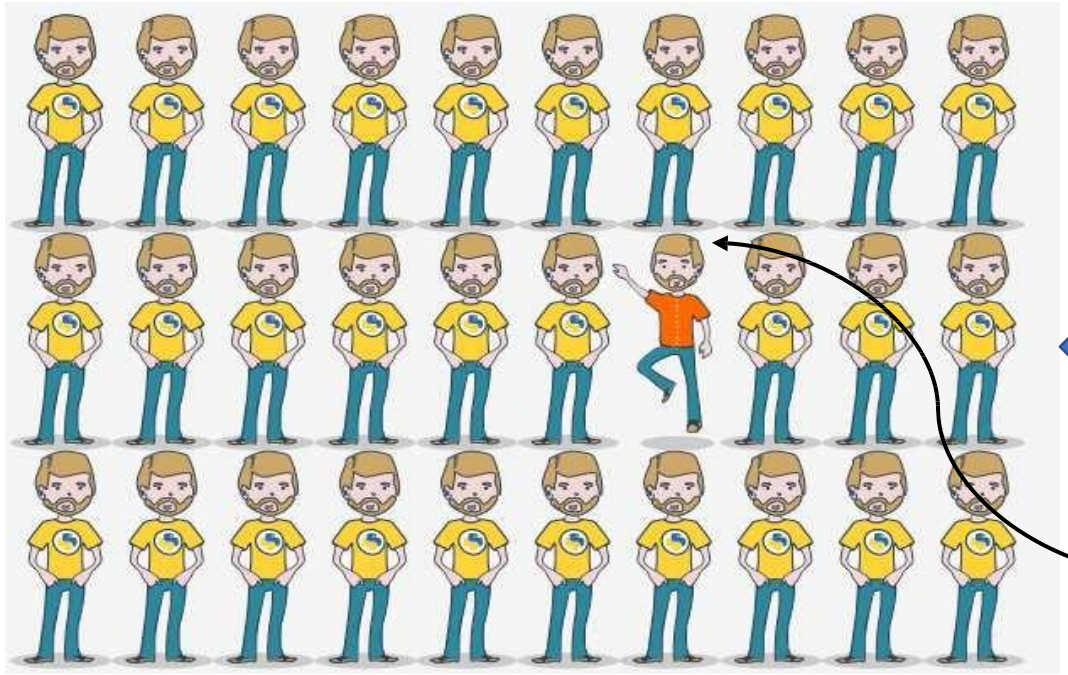
Bármi, ami elromolhat, el is fog romlani.

```
import math

x = float(input("Enter x: "))
y = math.sqrt(x)

print("The square root of", x, "equals to", y)
```

Meg tudja védeni magát az ilyen meglepetésektől? Természetesen igen. Sőt, meg is kell tennie ahhoz, hogy jó programozónak tekintsék.



Minden alkalommal, amikor a kódod megpróbál valami rosszat/bolondot/felelőtlenséget/őrültséget/elkövethetetlent tenni, a Python két dolgot tesz:

1. leállítja a programodat;
2. létrehoz egy speciális adatfajtát, amit kivételnek nevezünk.

kivétel lépett fel

Hibák, kudarcok és egyéb csapások

```
value = 1
value /= 0
```

```
Traceback (most recent call last):
  File "main.py", line 2, in <module>
    value /= 0
ZeroDivisionError: division by zero
```

```
my_list = []
x = my_list[0]
```

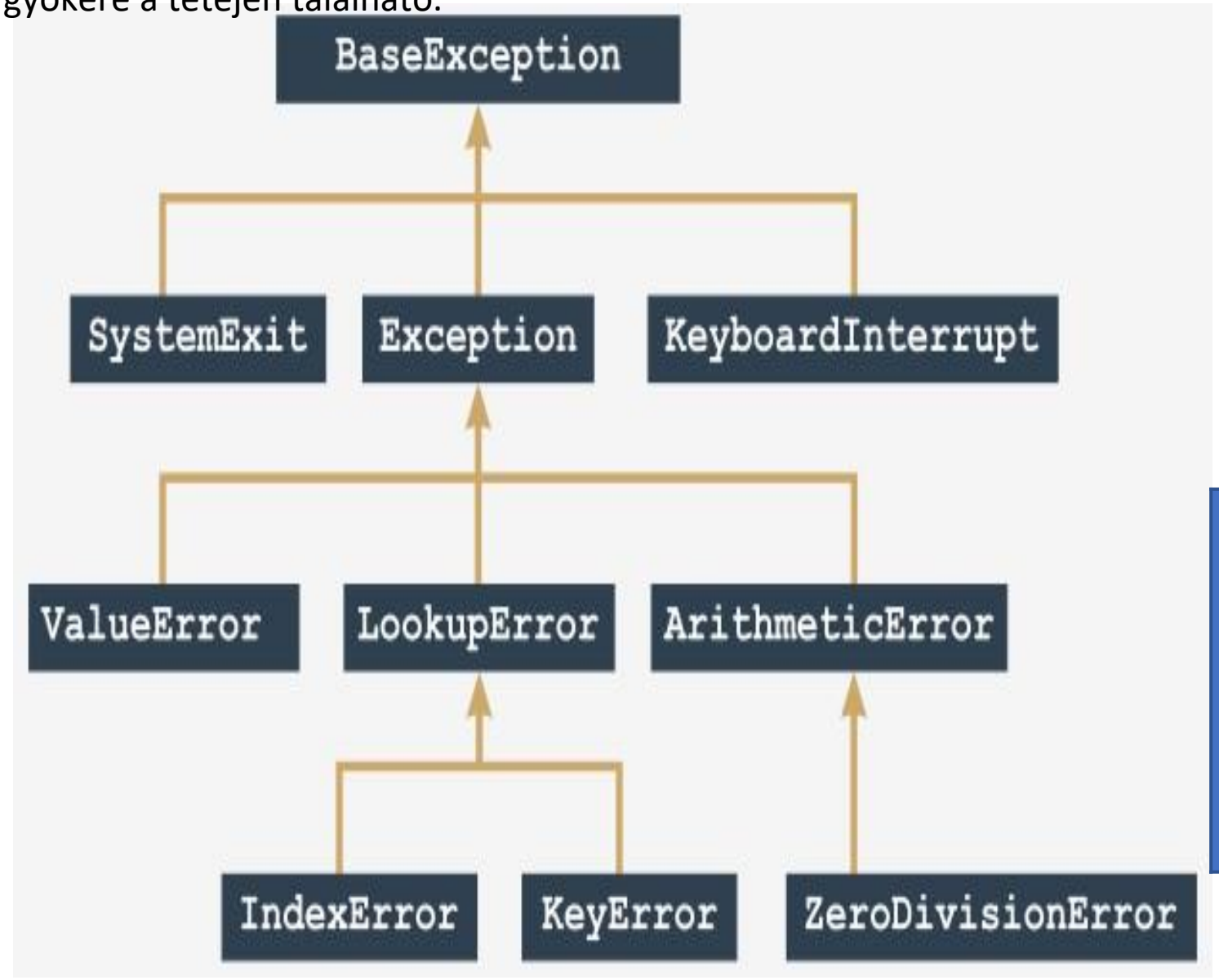
```
Traceback (most recent call last):
  File "lst.py", line 2, in
    x = list[0]
IndexError: list index out of range
```

```
try:
    x = int(input("Enter a number: "))
    y = 1 / x
    print(y)
except ZeroDivisionError:
    print("You cannot divide by zero, sorry.")
except ValueError:
    print("You must enter an integer value.")
except:
    print("Oh dear, something went wrong...")

print("THE END.")
```

Exceptions -><- Kivételek

A Python3 63 beépített kivételt definiál, és ezek mindegyike egy fa alakú hierarchiát alkot, bár a fa egy kicsit furcsa, mert a gyökere a tetején található.



BaseException
↑
Exception
↑
ArithmeticError
↑
ZeroDivisionError

```
try:
    y = 1 / 0
except ZeroDivisionError:
    print("Oooppsss...")

print("THE END.")
```

```
Oooppsss...
THE END.
```

```
try:
    y = 1 / 0
except ZeroDivisionError:
    print("Zero Division!")
except ArithmeticError:
    print("Arithmetic problem!")

print("THE END.")
```

Zero division!
THE END.

raise

A raise utasítás az adott, exc… nevű kivételt úgy fogja felvetni, mintha az a szokásos (természetes) módon lenne felvetve:

```
def bad_fun(n):
    raise ZeroDivisionError
try:
    bad_fun(0)
except ArithmeticError:
    print("What happened? An error?")

print("THE END.")
```

What happened? An error?

THE END.

Két vagy több kivétel kezelése

```
def bad_fun(n):
    try:
        return 1 / n
    except ArithmeticError:
        print("Arithmetic Problem!")
    return None

bad_fun(0)

print("THE END.")
```

Arithmetic Problem!

THE END.

```
def bad_fun(n):
    return 1 / n

try:
    bad_fun(0)
except ArithmeticError:
    print("What happened? An exception was raised!")

print("THE END.")
```

What happened? An exception was raised!
THE END.

Exceptions: continued

```
def bad_fun(n):
    try:
        return n / 0
    except:
        print("I did it again!")
        raise


try:
    bad_fun(0)
except ArithmeticError:
    print("I see!")

print("THE END.")
```

```
I did it again!

I see!

THE END.
```

assert kifejezés

```
import math

x = float(input("Enter a number: "))
assert x >= 0.0

x = math.sqrt(x)

print(x)
```

```
Traceback (most recent call last):
  File ".main.py", line 4, in
    assert x >= 0.0
AssertionError
```

Hogyan használható?
olyan esetekben érdemes a kódodba tenni, amikor teljesen biztos akarsz lenni a nyilvánvalóan téves adatokban, és amikor nem vagy teljesen biztos benne, hogy az adatokat korábban alaposan megvizsgálták (pl. egy más által használt függvényen belül).egy AssertionError kivétel kiváltása biztosítja a kódot a következőktőlérvénytelen eredményeket produkáljon, és egyértelműen megmutatja a hiba természetét;
az assertionok nem váltják ki a kivételeket, és nem érvényesítik az adatokat - ezek kiegészítői.

Beépített kivételek

`ArithmeticError`
Elhelyezkedés: `BaseException ← Exception ← ArithmeticError`
Leírás: absztrakt kivétel, amely magában foglalja az olyan aritmetikai műveletek által okozott kivételeket, mint a nulla osztás vagy egy argumentum érvénytelen tartománya.

AssertionError
Elhelyezkedés: BaseException ← Exception ← AssertionError
Leírás: Az assert utasítás által kiváltott konkrét kivétel, amikor az argumentuma False, None, 0 vagy üres karakterláncra értékelődik ki

BaseException
Elhelyezkedés: BaseException
Leírás: Az összes Python kivétel közül a legáltalánosabb (legabsztraktabb) - az összes többi kivétel benne van; azt lehet mondani, hogy a következő két except ág egyenértékű: except: és except BaseException.

Beépített kivételek

IndexError
Elhelyezkedés : BaseException ← Exception ← LookupError ← IndexError
Leírás: konkrét kivétel, amely akkor lép fel, ha egy nem létező szekvencia eleméhez (pl. egy lista eleméhez) próbál hozzáférni.

KeyError
Elhelyezkedés : BaseException ← Exception ← LookupError ← KeyError
Leírás: konkrét kivétel, amely akkor lép fel, ha egy gyűjtemény nem létező eleméhez (pl. egy szótár eleméhez) próbál hozzáférni.

KeyboardInterrupt
Elhelyezkedés : BaseException ← KeyboardInterrupt
Leírás: konkrét kivétel, amely akkor lép fel, amikor a felhasználó a program végrehajtásának befejezésére szolgáló billentyűparancsot használja (Ctrl-C a legtöbb operációs rendszerben); ha a kivétel kezelése nem vezet a program befejezéséhez, a program folytatja a végrehajtást..

Built-in exceptions

MemoryError
Elhelyezkedés : BaseException ← Exception ← MemoryError
Leírás: Konkrét kivétel, amely akkor lép fel, ha egy műveletet nem lehet befejezni a szabad memória hiánya miatt.

OverflowError
Elhelyezkedés : BaseException ← Exception ← ArithmeticError ← OverflowError
Leírás: konkrét kivétel, amely akkor lép fel, ha egy művelet túl nagy számot eredményez a sikeres tároláshoz.

ImportError
Location:): BaseException ← Exception ← StandardError ← ImportError
Leírás: konkrét kivétel, amely az importálási művelet sikertelensége
esetén lép fel.

Az objektumorientált koncepció alapfogalmai

Az informatika évtizedeiben a szoftverfejlesztés domináns megközelítése a procedurális programozási stílus volt, és ma is ezt alkalmazzák. Sőt, a jövőben sem fog eltűnni, mivel nagyon jól működik bizonyos típusú projektek esetében (általában nem nagyon összetett és nem nagy projektek esetében, de ez alól rengeteg kivétel van).

Az objektumos megközelítés meglehetősen fiatal (sokkal fiatalabb, mint a procedurális megközelítés), és különösen akkor hasznos, ha nagy és összetett projekteknél alkalmazzák, amelyeket nagy, sok fejlesztőből álló csapatok hajtanak végre.

Procedurális vs. objektumorientált szemlélet

Procedural

A procedurális megközelítésben két különböző és teljesen különálló világot különböztethetünk meg: az adatok világát és a kód világát. Az adatok világát különböző típusú változók népesítik be, míg a kód világát modulokba és függvényekbe csoportosított kódok lakják.

A függvények képesek adatokat használni, de fordítva nem. Továbbá a függvények képesek visszaélni az adatokkal, azaz jogosulatlanul felhasználni az értéket (pl. amikor a szinusz függvény paraméterként megkapja a bankszámla egyenlegét).

Az adatok nem használhatnak függvényeket

object-oriented

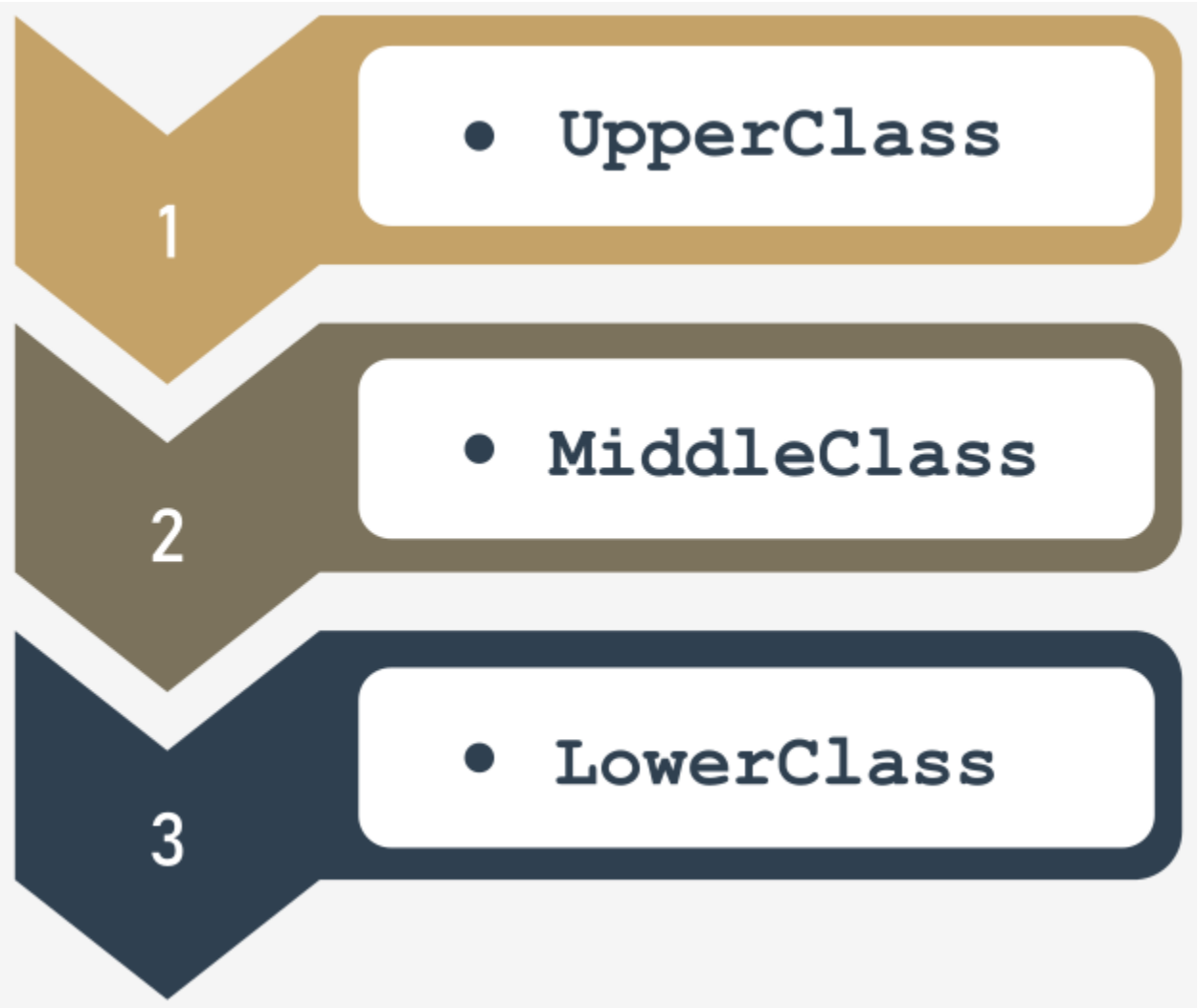
Az adatok és a kód egyazon világba vannak zárva, osztályokra osztva.

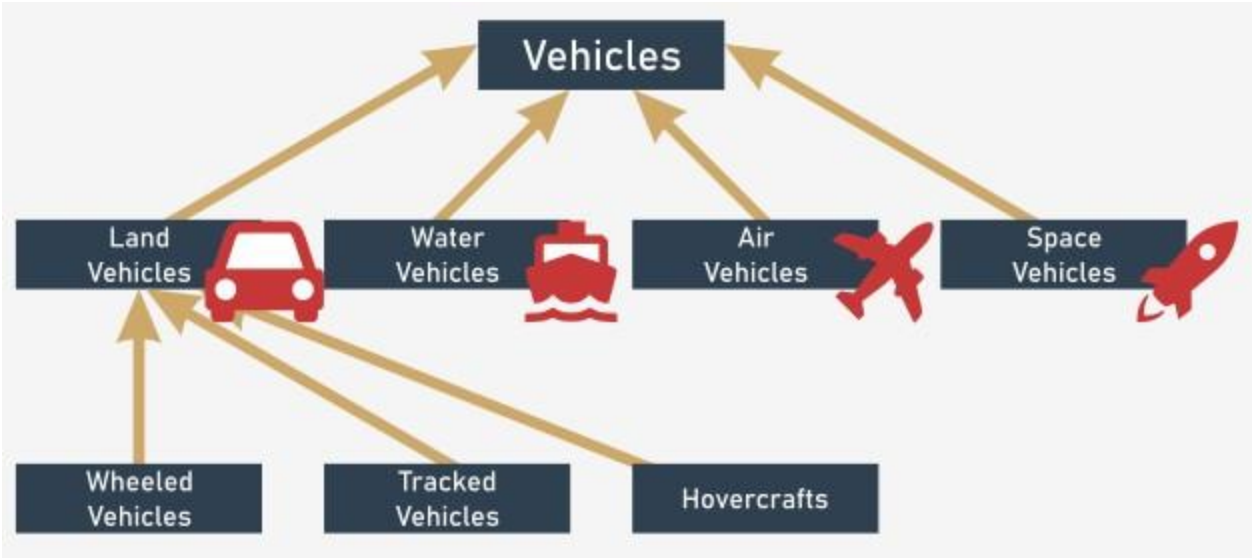
osztály olyan, mint egy recept, amelyet akkor lehet használni, amikor egy hasznos objektumot akarsz létrehozni.

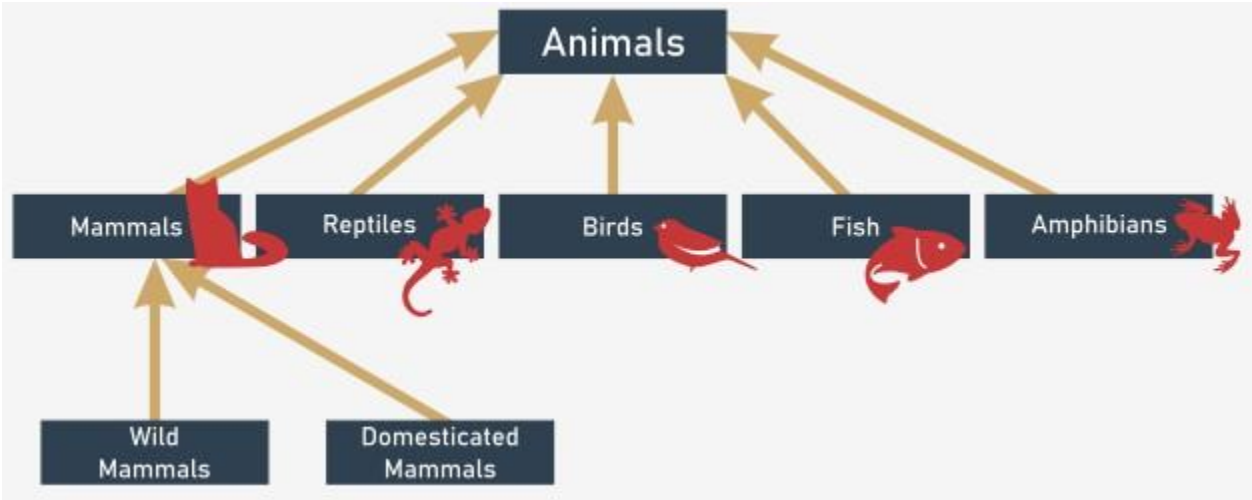
Minden objektumnak van egy halmaz jellemvonása

Objektumok az osztályokban kifejezett eszmék megtestesülései, mint ahogy a tányérodon lévő sajttorta is egy régi szakácskönyvben kinyomtatott receptben kifejezett eszme megtestesülése.

Osztály hierarchiák

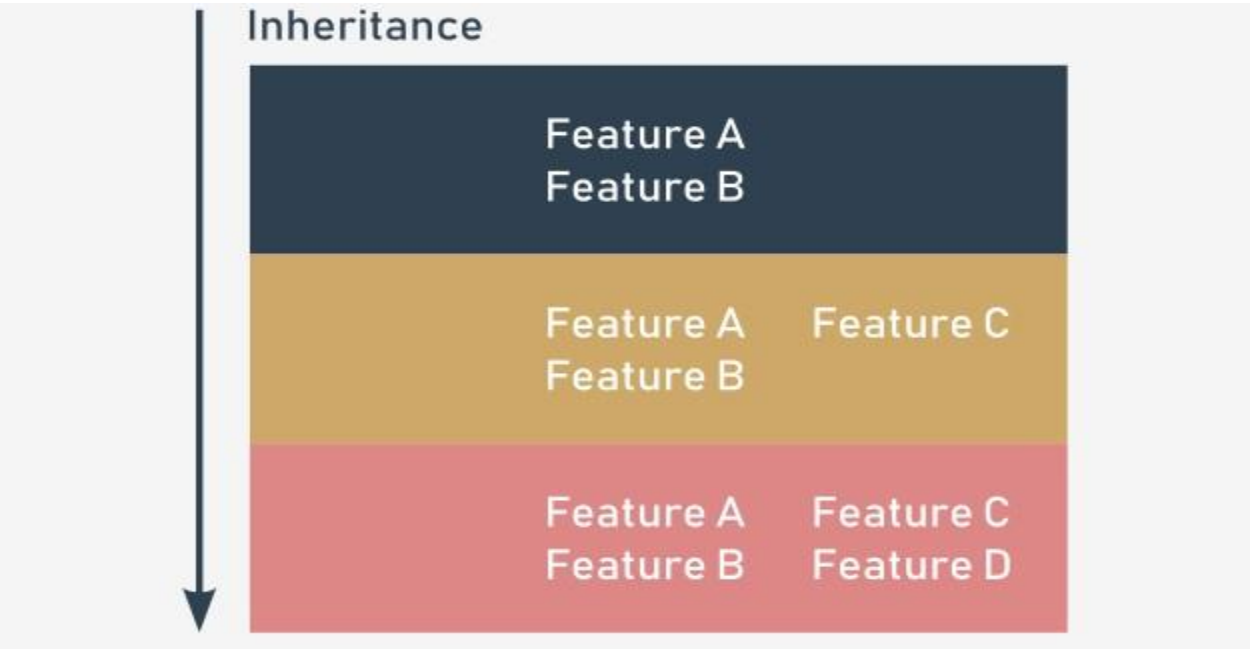


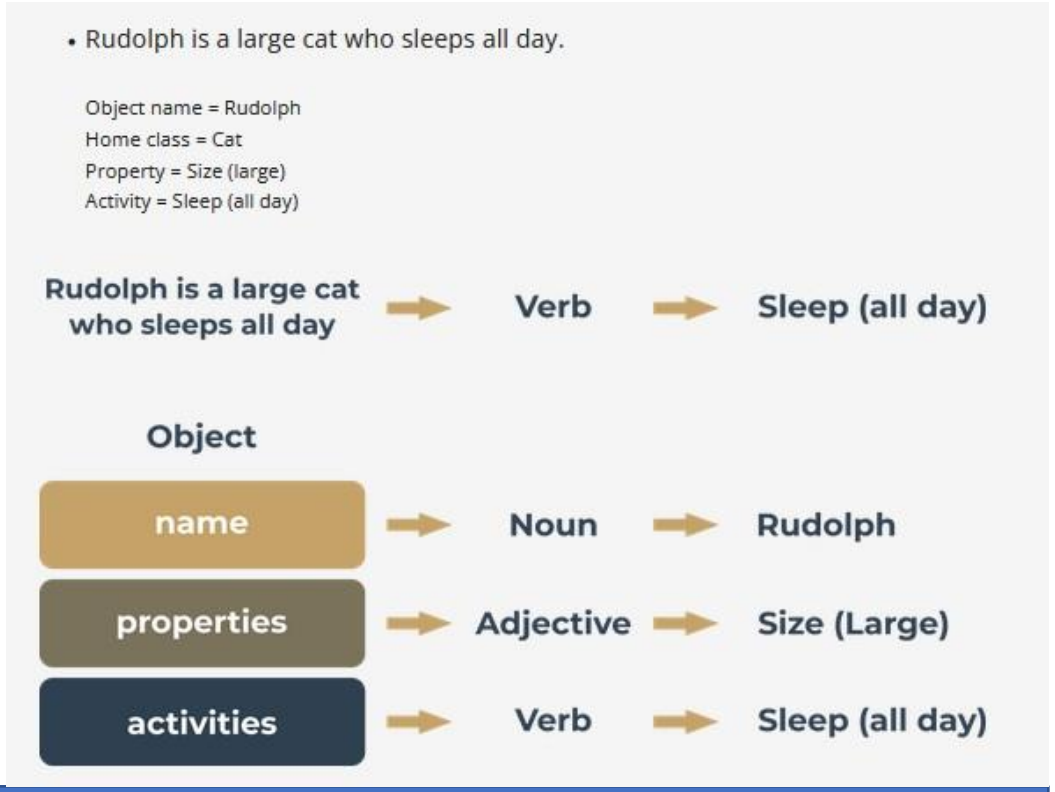




Mi az a objektum?

Az objektum egy adott osztályhoz rendelt követelmények, jellemzők és vonások megtestesülése (incarnation).





Mit birtokol egy objektum?

- Egy objektumnak van egy neve, amely egyedileg azonosítja azt a saját névterében (bár lehetnek névtelen objektumok is)
- Egy objektumnak van egy halmaz egyedi tulajdonságai, amelyek eredetivé, egyedivé vagy kiemelkedővé teszik (bár lehetséges, hogy egyes objektumok egyáltalán nem rendelkeznek tulajdonságokkal)
- Egy objektumnak van egy halmaza olyan képességeknek, amelyekkel meghatározott tevékenységeket végezhet, és amelyek képesek megváltoztatni magát az objektumot, vagy néhány más objektumot.

Találkozunk a laborban☺