လက်တွေ့အသုံးချ ပရိုဂရမ်းမင်း

(အခြေခံအရေးအသားဆိုင်ရာ မှတ်စုများ)

သူရအောင်

Types of Knowledges

- Declarative Knowledge နဂိုသိ
  - ရေခဲသည် အေးသည်
- Imperative Knowledge ထွင်သိ
  - သန်ျာညီမျှခြင်းတကြောင်းကို တွက်ထုတ်ခြင်း
  - စီစဉ်ထားသော ကွန်ပျူတာ ပရိုဂရမ်များ

Two types of Programs

- Fix program မူလပါ ပရိုဂရမ်များ
  - ဥပမာ - Calculator
- Stored program ပြန်ရေး ပရိုဂရမ်များ
  - အဆင့်ဆင့် စေခိုင်းချက်များ
  - Language များ ပေါ်ပေါက်လာရခြင်း
- Turing machine
  - သွင်း/ထုတ်
  - ဘယ်/ညာ ရွှေ့
  - ဖျက်
  - ဘာမှမလုပ်တဲ့အခြေအနေ (idle)

Aspects of language

Set of primitive constructions

အခြေခံ ပုံစံများ

- literals (numbers, strings)
- infix operations (simple operators)

Syntax ပုဒ် အထားအသို

< Object > < Operator > < Object >

```
>>> "Hello" 5 # not valid syntax
>>> 3 + 5 # valid syntax
```

Static Semantic ရေးထုံးရေးဟန်

```
>>> 3 + 5
>>> "Hello" + 5
```

Semantic အဓိပ္ပါယ် ပါခြင်း မပါခြင်း

- infinite loops ထပ်ခါတလဲလဲ ဆက်တိုက်လုပ်
- ထင်မထားသော ရလဒ် error ကင်းသော်လည်း အလုပ်မလုပ်

သတိပြုရန်

Synatx နဲ့ Sematic error တွေဟာ ရှာရလွယ်ကူတယ်။ Static Semantic တွေကတော့ ရှာရတာ ခက်ခဲတယ်။ သူတို့ကို error လို့ ခေါ်ဖို့ထက် **bug** လို့ ခေါ်ကြတယ်။

Python ဆိုသည်မှာ

- What is Python ?
- Why Python ?
- Installing Python

python.org

- Choosing Text Editor

https://marketplace.visualstudio.com/items…

- Or Anaconda Distribution

anaconda.org/Download

- First Python program

Basic Elements of Python

OOP ? - Literals တွေကို Objects တွေအဖြစ် အမျိုးအစားခွဲထားတယ်။ Python မှာ အရာရာတိုင်းသည် Object ဖြစ်တယ်။

Objects

- Scalar ထပ်မံခွဲခြားမရသော Object များ
- Non-Scalar ခွဲစိတ်ရန်ိုင်သေးသော Object များ
- Scalar Objects
    - Numbers (int, float)
    - Booleans (bool)
    - Null values (None)

Numerical Operations

<Object> <Operator> <Object>

- **+ - * /** ပေါင်း နုတ် မြှောက် စား
- ****** power ထပ်ကိန်း
- **//** integer division ကိန်းပြည့်အစား
- **%** modulo အကြွင်း

Orders of Operations

( ) ** * // / % + -

ကွင်း ၏ မြှောက် စား ပေါင်း နုတ်

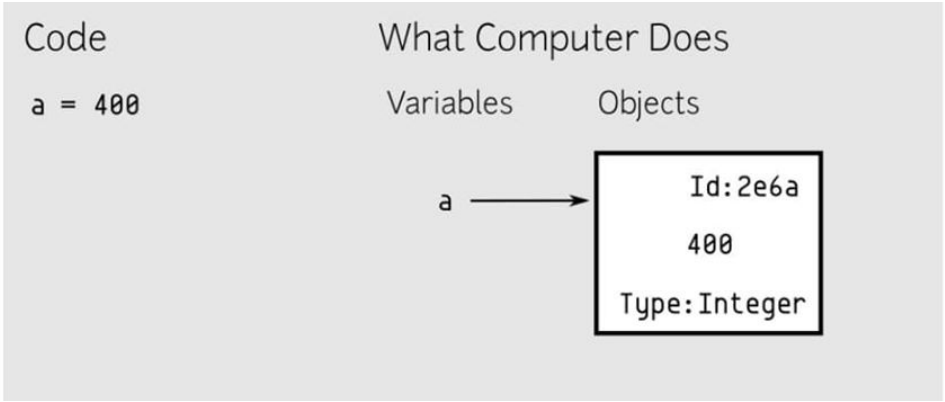
Variables ကိန်းရှင်များ

ဘာကြောင့် Variable ပေးရလဲ

To reuse name instead of value

- Variable Rebind လုပ်ခြင်း

- Multiple Assignments



Create a variable with assign (=)

(var_a = 12)

- သတိပြုရန် Python သည် dynamic language ဖြစ်သောကြောင့် int var_a ဟု သီးသန့် ကြေညာရန် မလိုပါ။
- Underscore (_), alpabets, numbers များမှ ကျန် စာလုံးများ မပါဝင်ရပါ။ numbers များနှင့် မစရ။ Python တွင် ပါဝင်အလုပ်လုပ်နေသော နာမည်များ မဖြစ်စေရ။

  (အကြံပြုထားသောပုံစံ - var_a)

Strings and Input

Strings **စာသားများ**

+ concatenate

* repeat

len(str)

str.upper( )

str.lower( )

Output ➡️ print("print **ထုတ်မယ့်စာ**")

Input ➡️ input("message")

သတိပြုရန် input function က ရသမျှသည် string ဖြစ်သောကြောင့် int(input(" ")) စသဖြင့် ဘာဏ်ရှိသလို ပြောင်းလဲရမည်။ eval(input(" ")) လည်း သုံးနိုင်သည်။

Conditionals

- Conditionals : tools to test
- Comparison Operators သလား

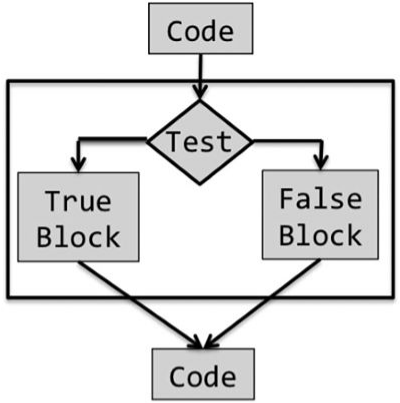


Figure 2.3 Flow chart for conditional statement

== equal ညီသလား
!= မညီဘူးလား
> greater than ကြီးသလား
>= at most အများဆုံးဖြစ်သလား
< less than နည်းသလား
<= at least အနည်းဆုံး ဖြစ်သလား

Logical Operators
and အကုန်ပြေလည်
or တခုမဟုတ် တခု
not မဟုတ်

Conditionals : how to use
Condition တခု Statement ၂ ခု - if else
Condition ၁ ခုထက်ပို Statement ၂ ခုထက်ပို - if elif else
Condition တခုပြေလည်ရင် နောက်တခု ပြေလည်သေးလား - nested if

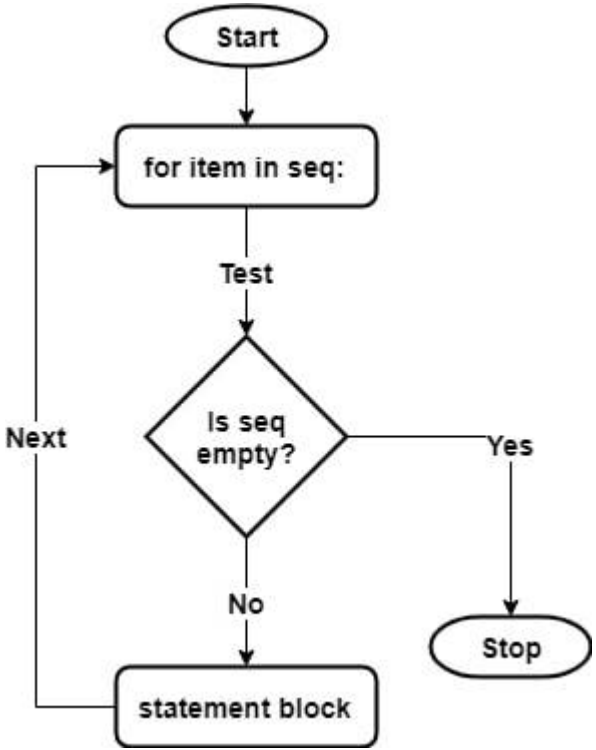
Iterations အကြိမ်ကြိမ် လုပ်ဆောင်ခြင်းများ

For loop

- ကြိုတင်သတ်မှတ်ထားသော Sequence ကြီး

```
for <item> in <sequence>:
        < expression >
        < expression >
print("Out of loop")
```

Tips : range(start,stop,steps)



Iterations အကြိမ်ကြိမ် လုပ်ဆောင်ခြင်းများ

While loop

သတ်မှတ်ထားသော condition
ပြေလည်နေသေးသမျှ

while <condition> :

    <expression>

    <expression>

print("Out of loop")

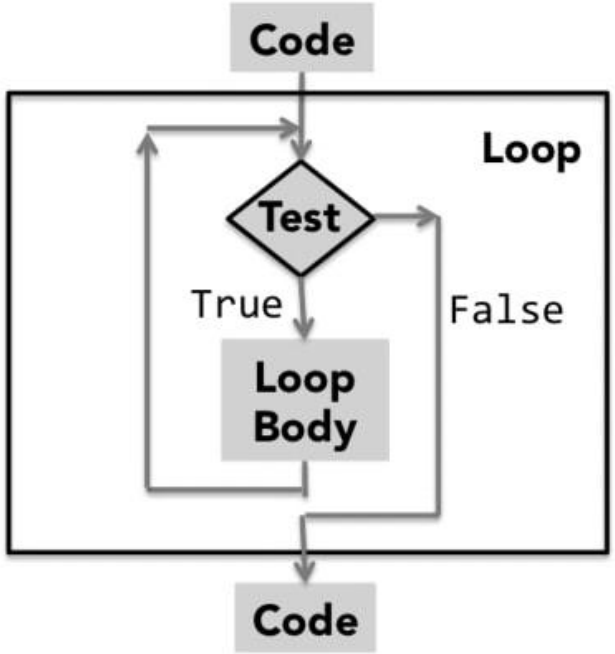
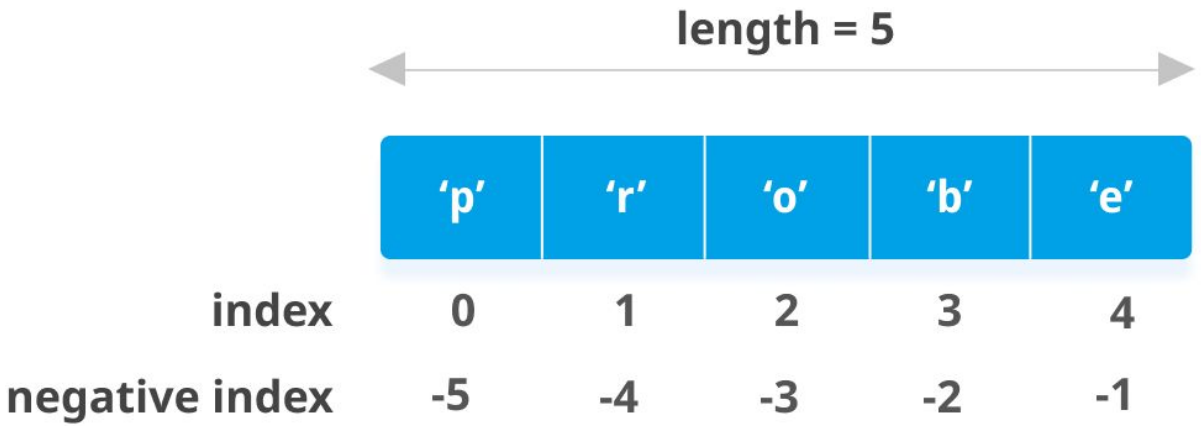


Figure 2.4 Flow chart for iteration

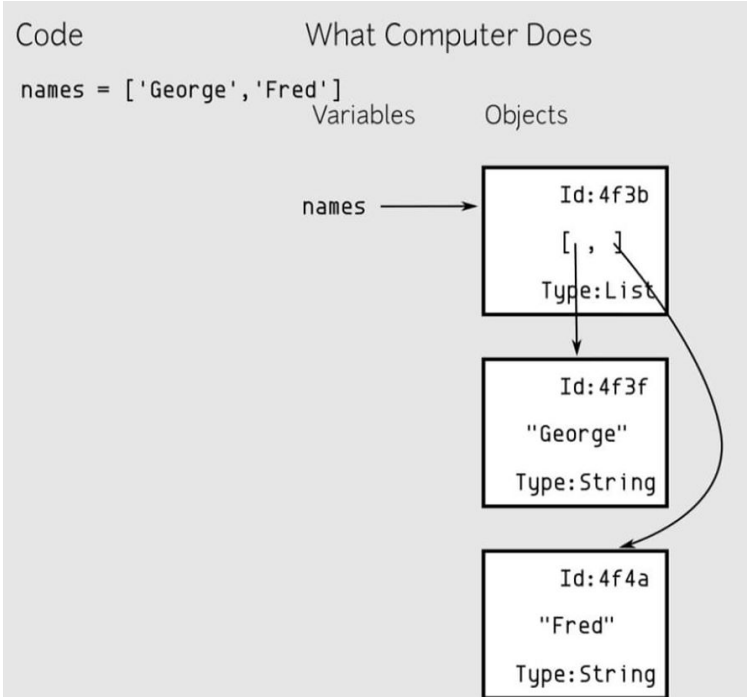
Non-Scalar Objects ထပ်မံခွဲထုတ်လို့ရ

- Strings
- Lists
- Tuples
- Dictionaries



Lists

- Ordered sequences of informations
- usually homogeneous (but can also be mixed)
- mutable (create ပြီးမှ modify ပြန်လုပ်လို့ရ)
- သတိပြုရန် - တခြား type တွေ အကုန် immutable
- indexing and slicing
- list တွေကို data တွေ အပြောင်းအလဲ လုပ်ချင်ရင် သုံးတယ်။ မပြောင်းလဲချင် safe ဖြစ်ချင်ရင် tuple သုံးတယ်။




```
>>> list_L = [ "H","E","L","L","O" ]

>>> list_L1 = list_L # list_L1 is not new list

>>> list_L1 = list_L [ : ] # list_L1 is new list

>>> list_L [ 0 ] = "Y"
```

[i for i in seq]

[i for i in seq if (condition)]

[i*j for i in seq1 for j in seq2]

Tuples

- Ordered sequences of elements

- heterogeneous ရောထည့်လို့ရ

- ( ) နဲ့ သတ်မှတ်

သတိပြုရန် - function နဲ့ မမှားဖို့ တလုံးထဲဆို ("H",) လို့ရေးပါ။ comma ပါရပါတယ်။

- like strings, can concatenate, repeat and slice

```
>>> t1 = ( 1,2,3 )
>>> t2 = ( "Hello", )
>>> print( t1 + t2 )
```

Dictionaries

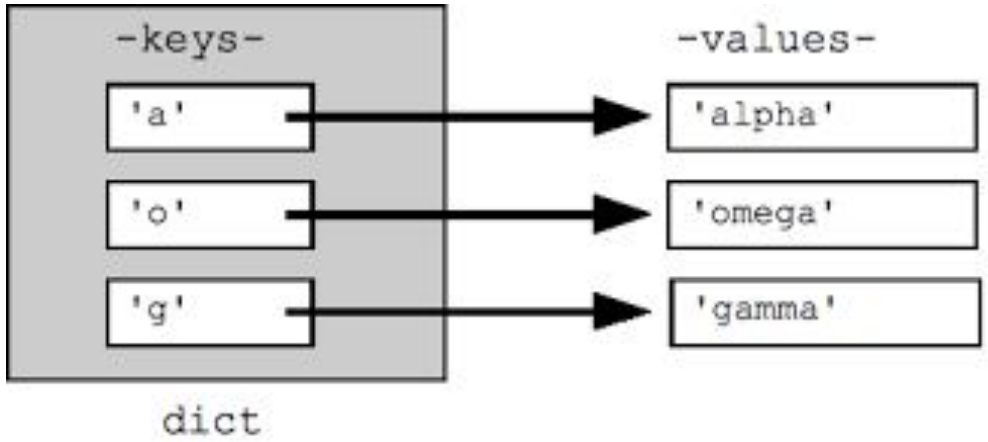
- unordered sequence ဖြစ်တယ်။
- { key : value } အတွဲတွေအဖြစ် သုံးတယ်။
- Hashing နည်းကို သုံးထားတယ်။
- a key must be unique and unchanged
- value လိုချင်ရင် dict_name[key]
- တခုခုရဲ့ Data တွေကို စုထားချင်တဲ့အခါ သုံးတယ်။
- ဖျက်ချင်ရင် keyword - del dict_name[key]

ဥပမာ **student1 {**

"name" : "Thura"

"age" : 18

}



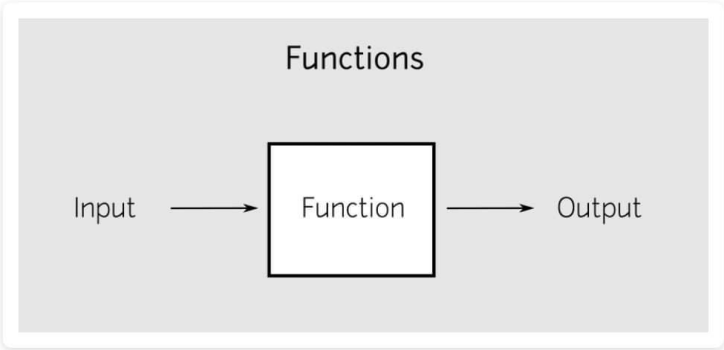
Functions in Python

လုပ်ဆောင်ချက်များ

Two types of Functions

- Built-in function

- defined function



A function is like a box. It can take input and return output. It can be passed around and reused.

```
def function_name(parameter):
        < expression >
        < expression >
        # local scope
        # in a defined function
        return < object >
# Out of a defined function
# Global Scope
```

- Function တခုဟာ parameter တခု သို့မဟုတ် တခုထက်ပို ကို လက်ခံရယူ တွက်ချက်ပြီး return အမြဲပြန်တယ်။ Python မှာ return ဘာမှ ရေးမထားရင်တောင် None ဆိုတာ ပြန်ပေးတယ်။
- parameter နေရာမှာ တခြား Function ကိုလည်း ထည့်လို့ရတယ်။ Function ထဲက function ပေါ့။ ဒါကို Higher order function လို့ ခေါ်တယ်။
- Recursive Function (aka Dynamic Programming) ထပ်ခါတလဲလဲ ကျော့သော ဖန်ရှင်များ
    - Base Case
    - Inductive Case

Why we should ?

- Decompose ခွဲထား

- Abstract ကုတ်လျော့

Lambda Functions
အမည်မဲ့ ဖန်ရှင်များ

သင်္ချာတွက်ထုတ်ခြင်းများအတွက် ဒါမှမဟုတ် တခဏတာ လိုအပ်ချက်အတွက် function ရေးဖို့ လိုအပ်လာရင် တိုတို တုတ်တုတ် ထိထိရောက်ရောက် ရေးနိုင်ရန် သုံးသည်။

```
In [1]: #lambda funct
        lambda a: a*2
Out[1]: <function __main__.<lambda>(a)>

In [2]: (lambda a: a*2)(10) # 10 is argument
Out[2]: 20

In [3]: f = lambda a : a**2

In [4]: f(10)
Out[4]: 100

In [5]: g = lambda x,y : x + y

In [6]: g(10,5)
Out[6]: 15
```

map( ) နဲ့ filter( ) တွေဟာ range( ) လိုပဲ Python မှာ ပါတဲ့ generators တွေ ဖြစ်ပါတယ်။ ကိုယ်ပိုင် generators တွေကိုတော့ yield နဲ့ ရေးရပါတယ်။

map(,) ဟာ ရှေ့က function နဲ့ နောက်က sequence ကို အလုပ်လုပ်ဖို့ ချိတ်ပေးတာ ဖြစ်ပါတယ်။

filter(,) ကိုတော့ အခြေအနေ စစ်ထုတ်ပစ်ပြီး တခုခုလုပ်ချင်တဲ့အခါမျိုးမှာ သုံးရင် ကောင်းမွန်ပါတယ်။

In [1]:
```
# functional tools
# map(function,iterable)
list(map(lambda x: 2*x,[0,1,2,3,4]))
```

Out[1]:
```
[0, 2, 4, 6, 8]
```

In [2]:
```
g = lambda num: num%2 == 0
```

In [3]:
```
# filter
list(filter(g,[0,1,2,3,4]))
```

Out[3]:
```
[0, 2, 4]
```

In [4]:
```
list_L = list(range(-10,10))
```

In [5]:
```
g = lambda x : x >= 0
```

In [6]:
```
list(filter(g,list_L))
```

Out[6]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Error Handling in Python

User နဲ့ ပိုသက်ဆိုင်တဲ့ Error တက်နိုင်တဲ့ နေရာတွေမှာ error တက်လည်း program ဆက် အလုပ်လုပ်နေအောင် သုံးတယ်။

Common Types of Errors

- SyntaxError
- NameError
- AttributeError
- TypeError
- ValueError
- IOError

```
try :
    ရေးလိုက်တဲ့ ကုတ်
except :
    Error တက်နေရင် အလုပ်လုပ်မယ့် ကုတ်
```

Defensive Programming

Try except နဲ့ ပြမယ့်အစား တခါတည်း ကုတ်ကို အလုပ်မလုပ်စေမယ့် လုပ်ဆောင်ချက်ကို "မှလွဲ၍" သတ်မှတ်လိုက်တယ်။

assert မှ လွဲ၍

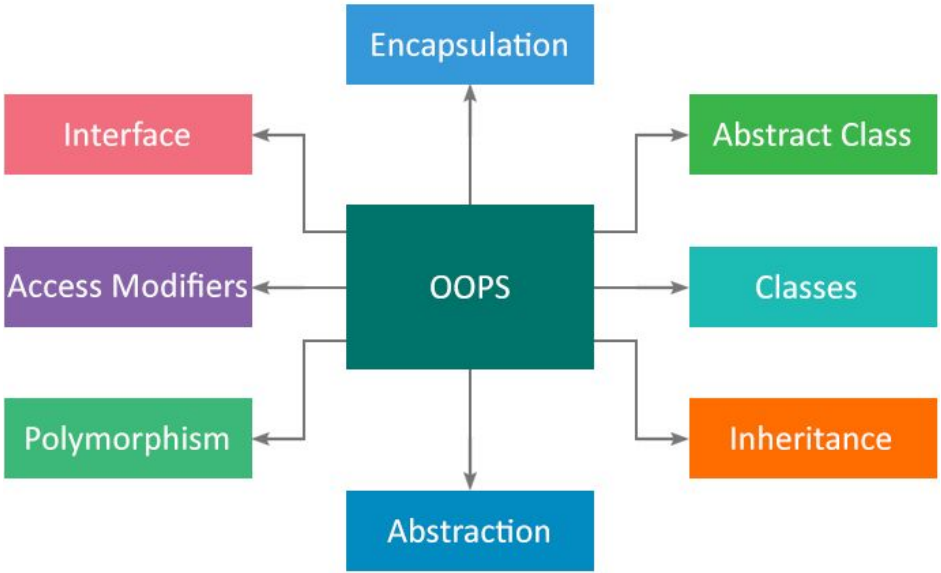
```
assert len(grades) != 0 :
    print("there is no grade")
```

OOP in Python

Everything is an object.

Literals တွေကို Object တွေကို အဖြစ် အမျိုးအစား ခွဲထားတယ်။ အမျိုးအစား ခွဲထားတယ်ဆိုတာ အုပ်စု ဖွဲ့ထားတာ ဖြစ်တယ်။ အဲ့ဒီ အုပ်စုကြီးကို class လို့ခေါ်တယ်။

>>> type( )



Robot

- name:
- color:
- weight:

+ introduceSelf()

r1 = 

- name: "Tom"
- color: "red"
- weight: 30

r2 = 

- name: "Jerry"
- color: "blue"
- weight: 40

Person

- name:
- personality:
- isSitting:
- robotOwned:

+ sitDown()
+ standUp()

p1 = 

- name: "Alice"
- personality: "aggressive"
- isSitting: false
- robotOwned: r2

p2 = 

- name: "Becky"
- personality: "talkative"
- isSitting: true
- robotOwned: r1

special functions

- __init__(self,var_a) var_a ကို initialize လုပ်တာ
- __str__( ) ရလဒ်ကို print(self) ထုတ်ပေးတာ
- __len__( ) len(self)
- __eq__(self,other) self == other
- __lt__(self,other) self < other
- __add__(self,other) self + other
- __sub__(self,other) self - other

Custom Classes

Code | What Computer Does

Variables | Objects

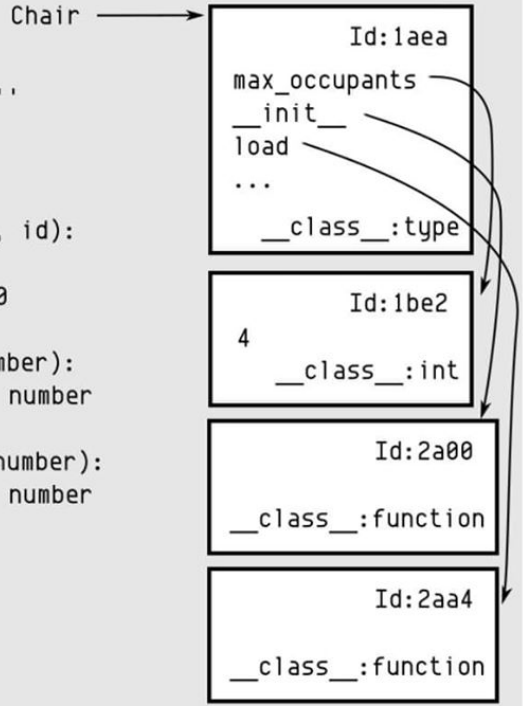
```
class Chair:
    '''Another Chair'''

    max_occupants = 4

    def __init__(self, id):
        self.id = id
        self.count = 0

    def load(self, number):
        self.count += number

    def unload(self, number):
        self.count -= number
```



Subclasses

Code | What Computer Does

```
sixer = Chair6(76)
```