# f-string

**self** is a reference to object methods

**cls** is a reference to class methods

BaseException  
 +-- KeyboardInterrupt  
 +-- Exception  
 +-- ArithmeticError  
 | +-- ZeroDivisionError  
 +-- AssertionError  
 +-- AttributeError  
 +-- EOFError  
 +-- ImportError  
 | +-- ModuleNotFoundError  
 +-- LookupError  
 | +-- KeyError  
 +-- NameError  
 +-- SyntaxError  
 | +-- IndentationError  
 +-- ValueError

|  |  |
| --- | --- |
| {num:,} | separate every 3 digits using comma |
| {num:.2f} | round the number by two digits |
| {num:.2%} | Two decimal digits |
| {num:x} | cause that field to be a minimum number of x characters wide |
| {str!a} | Ascii() |
| {str!r} | Repr() |
| {str!s} | Str()  To represent sets of sets, the inner sets must be frozenset |

# Data Structures

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Type** | **Indexing** | **Ordered** | **Mutable** | **Duplicate** | **How To** |
| List [] | ✓ | ✓ | ✓ | ✓ | [] / list() |
| Tuple () | ✓ | ✓ | ✗ | ✓ | () / tuple() |
| Set {} | ✗ | ✗ | ✓ | ✗ | set() |
| Dictionary{K:V} | ✗ | ✓ | ✓ | ✗ | {} / dict() |

# Built-in Funcs

|  |  |
| --- | --- |
| **Syntax** | **Description** |
| bool([x]) |  |
| callable() | If this returns True, it is still possible that a call fails, but if it is False, calling obj will never succeed. |
| enumerate() | will give you the value and its index in an iterable object as an enumerate obj. |
| round(num [, decimal place]) |  |
| map(func, iterable) | map a function over an iterable obj and returns a generator |
| filter(func, iterable) | map a conditional func |
| sorted(iterable,[key=\*,reverse=False]) |  |
| zip(\*iterables, strict=False) | turns rows into columns, and columns into rows |
| getattr(object, name[, default]) | = object.name | AttributeError if no default |
| hasattr(object, name) |  |
| setattr(object, name, value) | create or update value of an attribute |
| delattr(object, name) | = del object.name |

**Closure**: local, global, unlocal

✗ Change global variable in func → use global

✓ Read global variable in func

Return a function inside a function

def Quadratic(a,b,c):  
        def inner(x):  
                return a \* (x\*\*2) + b \* x + c  
        return inner

q = Quadratic(3,4,1)  
print(q(3)) #3 \* (5\*\*2) + 4 \* 5 + 1

# Dunders

|  |  |
| --- | --- |
| **Dunder** | **Description** |
| \_\_init\_\_ | *Always* runs when instantiating |
| \_\_repr\_\_ | <\_\_main\_\_.class object at 0x104aada60> |
| \_\_str\_\_ | print(instance) |
| \_\_enter\_\_ | =open | Context manager |
| \_\_exit\_\_ | =close | return True: never catch exception ; False: catch probable exception |
| \_\_call\_\_ | Define what happens when the user calls your instance or method |
|  |  |

# Arguments

def info(name, city, age): …

✓ def info(name, city, age=24)

✗ def info(name, city="New York", age)

No positional is allowed after a kwarg

def info(name, \*args): …

unknown number of positional arguments

def info(a,b,/,d):…

only 3 args. a, b pos-only. d kw-only

def info(a,b,\*,d):…

only 3 args. a, b pos-kw. d kw-only

def info(a,b,\*args, d):…

after \*args, arguments are kw-only

def info(a,b,/): …

prevent from kw-calling by **/** as final arg

def info(name, \*\*kwargs): …

unknown number of keyword arguments

info(name, city, age): #positional

info(name="amin", city="NYC", age=21) #keyword

info(age=21, name="amin", city="NYC") #keyword

3 ways of calling a func

Order is not important in kw calling

While is slower than for

|  |  |  |
| --- | --- | --- |
|  | increment | condition |
| while | Python | Python |
| for | C | Python |

**finally**: will be always executed

**else**: only executes if no exceptions happen

**Errors**:

1. Syntax Error: typo
2. Logic Error: zero division, …
3. Semantic Error: unexpected output

**lambda** <variable names>: <returning value>

SWAP

print(id(x), id(y))

x , y = y ,x

print(id(x), id(y))

>>>

4335495504 4335495536

4335495536 4335495504

Str Methods:

* Str.split([thing to split by]) returns list
* Str.strip(characters) removes chars

[start:stop:step]

Operators

|  |  |
| --- | --- |
| + | Add |
| - | Subtract |
| \* | Multiply |
| / | Division |
| // | Floor |
| \*\* | power |
| % | mod |

a = ([5,4,6],2,3)

a[0] = 1 **ERROR**

a[0][0] = 1 **([1,4,6],2,3)**

x = 5

x = 3

will have different id

Defined functions are **mutable**

Variables with the same value may or may not have the same memory address

