파이썬의 메모리 구조

안태진(taejin@codecure.smuc.ac.kr)

상명대학교 보안동아리 CodeCure

목차

Python 개요

- 파이썬 (Python)
  - 귀도 반 로섬 (Guido Van Rossum)에 의해 만들어진 프로그래밍 언어
  - 특징
    - 객체 지향 프로그래밍
    - 인터프리터 방식
    - 동적 타이핑 방식
      - 프로그램 실행 시 변수의 데이터 타입 결정
  - 실행 가능한 의사코드라는 별명이 있을 정도로 문법 간단

Python 개요

- 종류
  - CPython
    - C언어로 짜여진 파이썬, 우리가 평소에 쓰는 파이썬
  - Cython
    - 파이썬 코드를 C언어로 변환하여 컴파일 하는 파이썬
  - Jython
    - Java로 구현되어 JVM에서 돌아가는 파이썬
  - Pypy
    - 파이썬으로 짜여진 파이썬
  - IronPython
    - C#으로 짜여진 .NET 플랫폼 용 파이썬

목차

Python 메모리

- 메모리 특징
  - 객체
    - 파이썬에서는 모든 것이 객체
    - 모든 객체들을 힙 영역에 저장하고 다른 변수들이 참조 함
    - 모든 객체는 refrence count를 가지고 있고 이것이 0이 되면 GC(Garbage Collection)이 객체 제거

```
class People():
    pass
```

```
print(' ', type(x))
print(' ', type(int), end = '\n\n')

p1 = People()
print(' ', type(p1))
print(' ', type(People), end = '\n\n')

print(' ', type(sys))
print(' ', type(type(sys)), end = '\n\n')

print(' ', type(type))
```

```
 <class 'int'>
 <class 'type'>

 <class '__main__.People'>
 <class 'type'>

 <class 'module'>
 <class 'type'>

 <class 'type'>
```

Python 메모리

- 자주 쓰일 것 같은 숫자들 (int: 0 ~ 256) 메모리에 한번 올려두고 모두 같은 숫자를 가리키도록 함

```
x = 0
y = 0
hex(id(x)), hex(id(y))
```

```
('0x70756c20', '0x70756c20')
```

```
x = 256
y = 256
hex(id(x)), hex(id(y))
```

```
('0x70758c20', '0x70758c20')
```

```
x = 10000
y = 10000
hex(id(x)), hex(id(y))
```

```
('0x1170a4a3870', '0x1170a4a3c90')
```

```
x = 1.0
y = 1.0
hex(id(x)), hex(id(y))
```

```
('0x11709a55a20', '0x11709a55b70')
```

- 변수에 값을 할당할 때 변수 생성됨

목차

- Python 개요
- Python 메모리
  - 메모리 특징
  - 메모리 구조
  - 메모리 관리

Python 메모리

- 메모리 구조
  - Global Space, Heap Space, Call Frame으로 구분
  - Global Space
    - 전역 변수들이 저장
    - 파이썬이 종료될 때 까지 지속
  - Heap Space
    - 폴더들이 저장
    - 간접 접근
  - Call Frame
    - 지역 변수
    - 호출 종료시 삭제

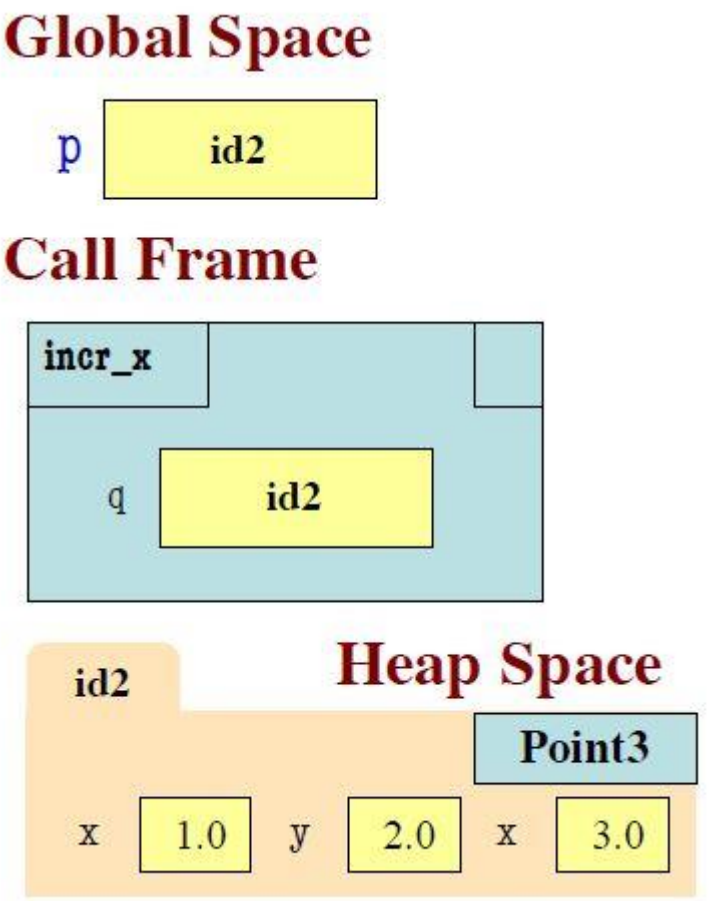
Python 메모리

```
class Point():
    def __init__(self, xPos, yPos, zPos):
        self.x = xPos
        self.y = yPos
        self.z = zPos

def incr_x(q):
    q.x = q.x + 1
```

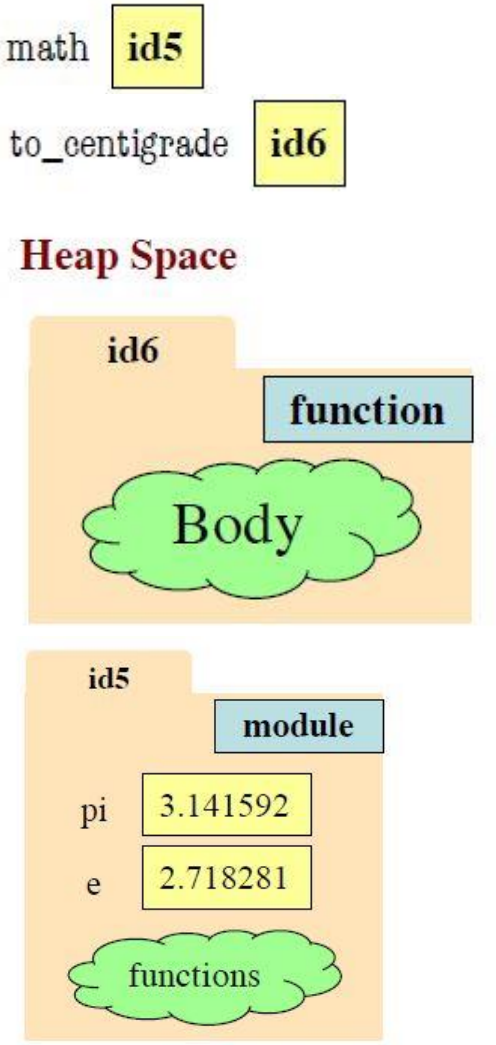
```
p = Point(1, 2, 3)
incr_x(p)
p.x
```

```
2
```



Python 메모리

- 함수 정의시 메모리 구조
  - 함수 이름과 똑같은 전역 변수 생성
  - 함수 몸체의 폴더 힙에 생성
  - 폴더의 id, 전역 변수에 저장



- 모듈 임포트시 메모리 구조
  - 모듈 이름과 같은 전역 변수 생성
  - 내용(변수, 함수) 폴더에 저장
  - 폴더의 id, 전역 변수에 저장
  - from 키워드는 Global Space에 저장

Python 메모리

- 함수 호출 시마다 Call Frame 생성
- Call Frame 생성 순서
  - 프레임 생성
  - 인자 값 매개변수에 할당
  - 함수 몸체 실행
  - 호출 프레임 제거

```
def to_centigrade(x):
    return 5 * (x - 32) / 9
```

```
print(' ', to_centigrade)
print(' ', hex(id(to_centigrade)))
print(' ', to_centigrade(32))
```

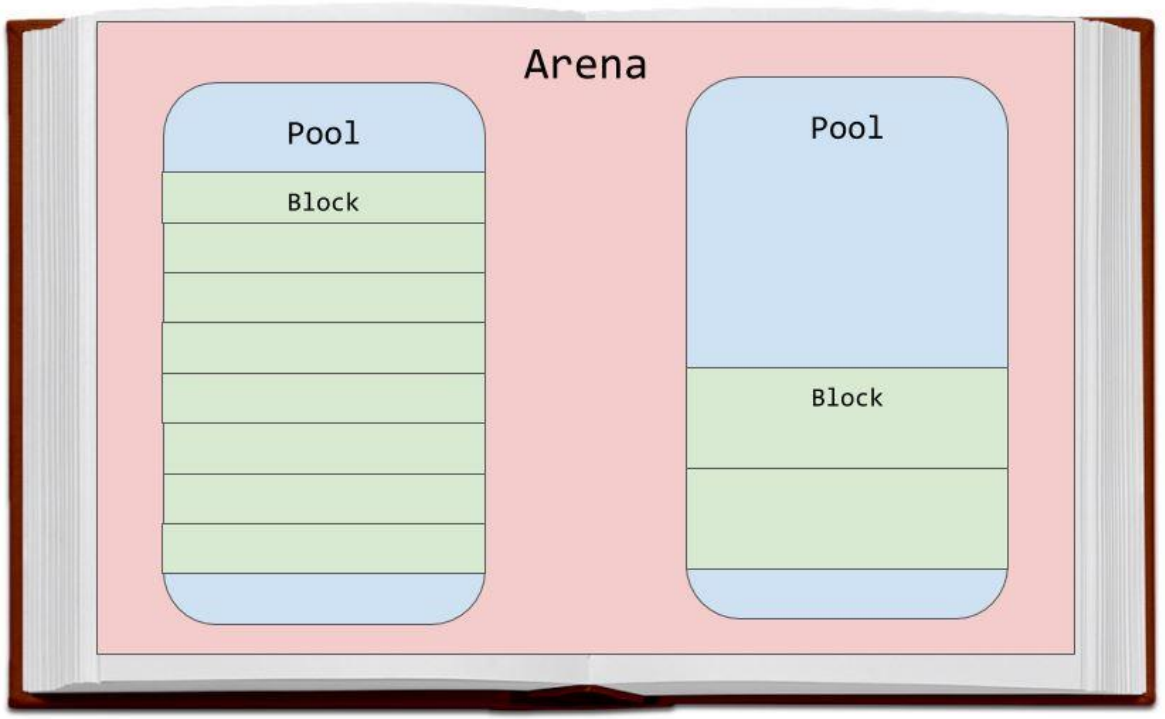
```
 <function to_centigrade at 0x000001170A4C9510>
 0x1170a4c9510
 0.0
```

목차

- Python 개요
- Python 메모리
  - 메모리 특징
  - 메모리 구조
  - 메모리 관리

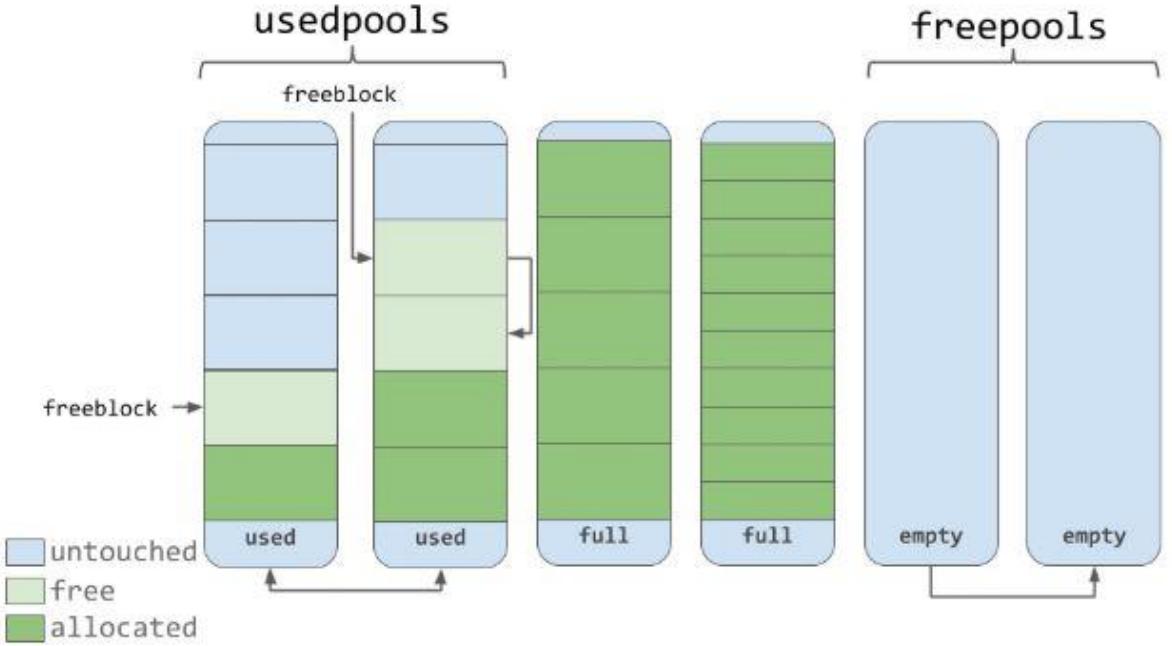
Python 메모리

- 메모리 관리
  - 파이썬은 메모리를 3 조각으로 나눠 관리
    - Arena, Pool, Block



Python 메모리

- Pool
  - 4KB의 Arena안에 있는 가상 메모리 페이지
  - used, full, empty의 3가지 상태로 존재
  - used pool에 할당할 block이 존재하면 used pool에 할당, 모든 공간에도 없다면 empty pool에 할당



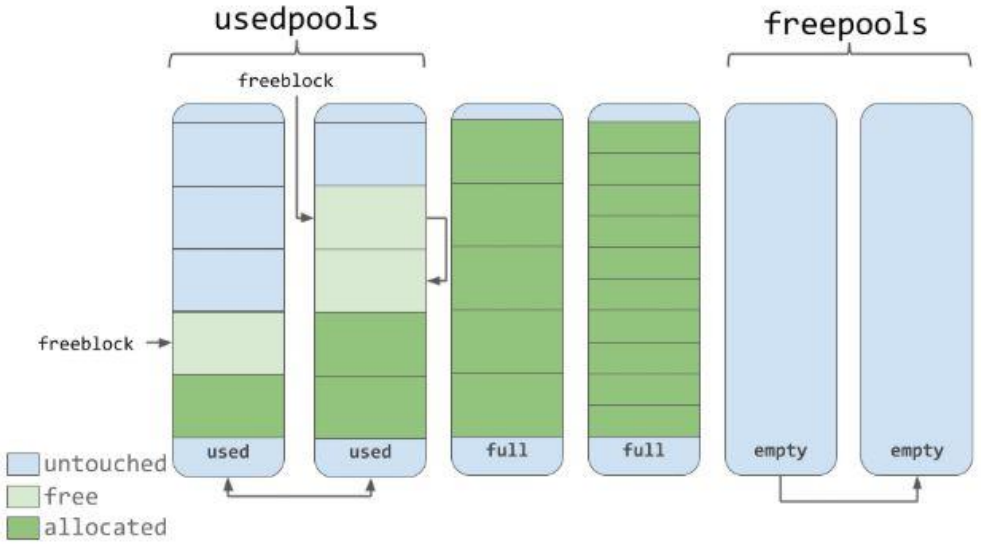
Python 메모리

- Block
  - Pool안에 있는 8bytes의 메모리
  - 요구되는 사이즈를 8단위 배수로 끊어 메모리 제공

Request in bytes	Size of allocated block
1-8	8
9-16	16
17-24	24
25-32	32
33-40	40
41-48	48
49-56	56
57-64	64
…	…
497-504	504
505-512	512

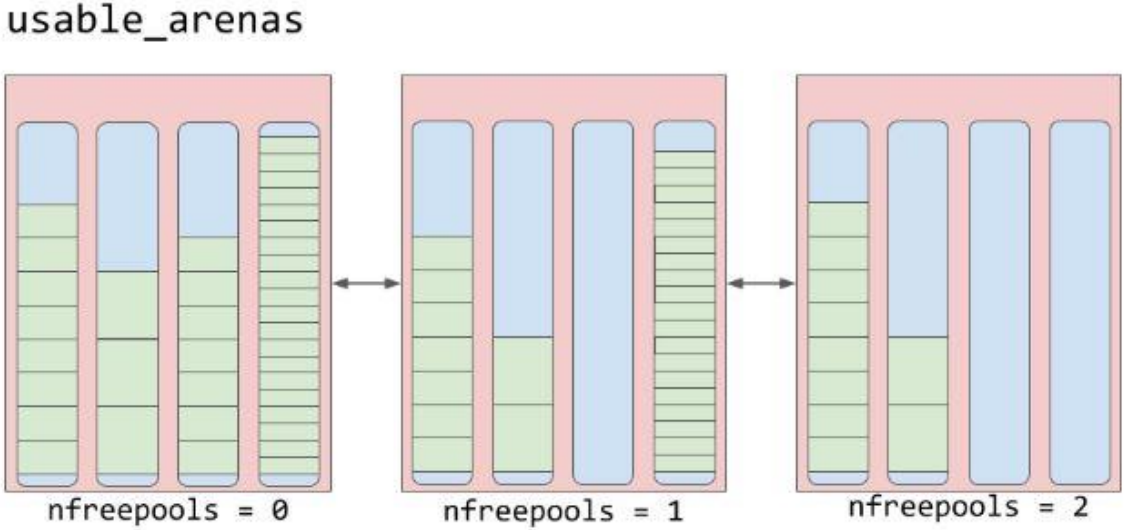
Python 메모리

- Block
  - untouched, free, allocated의 3가지 상태로 존재
  - free block은 완전히 free 되어서 OS에게 돌아간것이 아니라, 다음 데이터가 사용할 때까지 할당 되어있음
  - 사용 가능한 free block보다 요구되는 block이 더 많을 때 untouched block 사용



Python 메모리

- Arena
  - 256KB의 가장 큰 메모리 조각
  - Arena는 free pool이 적은 순서대로 연결 리스트의 앞에 위치함
  - Arena만 유일하게 완전히 free될 수 있으므로, 프로그램의 메모리 사용량을 줄이기 위해 이 방법 사용



감사합니다!