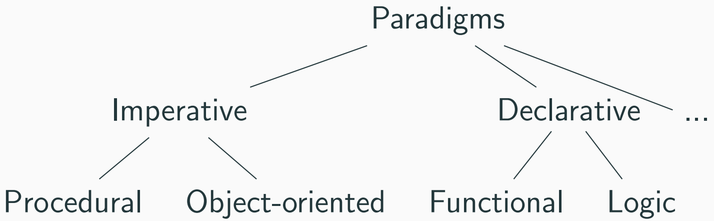
CS 314 Lecture 3

January 29, 2019

Paradigms

Paradigms


Paradigms
Imperative
Declarative
...
Procedural
Object-oriented
Functional
Logic

Paradigms

- Imperative, procedural: C, Pascal
- Imperative, object-oriented: C++, Java, C#, Python
- Functional: Haskell, OCaml, F#, Scheme
- Logic: Prolog

Typing

Typing can be done statically or dynamically.

	Static	Dynamic
Imperative	C, Java	Python
Functional	Haskell	Scheme

Imperative programming

Imperative programming

What is imperative programming?

- Program = series of statements that change state
- Assignment used to change values stored in memory

Closely matches execution of underlying hardware.

Imperative programming

Common features in imperative languages:

- Procedures
- Loops
- Blocks
- Conditional branches
- Unconditional branches

C

```c
#include <stdio.h>
#include <stdlib.h>

char cMessage[] = "Hello\n";

/* Execution will start here */
int main (int argc, char **argv)
{
    int i, count;

    count = atoi(argv[1]);
    for (i = 0; i < count; i++) {
        printf("Hello %d\n", i);
    }
}
```

Pointers

```c
int x = 42;
int* p = &x;
```

Pointers

```c
int i;
int* ptr;

i = 4;
ptr = &i;
*ptr = *ptr + 1;
```

Parameter passing

C uses “call by value” for parameter passing:

```c
void swap(int x, int y)
{
    int tmp = x;
    x = y;
    y = tmp;
}

int main(char* args[])
{
    int a = 10;
    int b = 20;
    swap(a, b);
    printf("%d, %d\n", a, b);
}
```

Parameter passing

To swap arguments we need pointers:

```c
void swap(int* x, int* y)
{
    int tmp = *x;
    *x = *y;
    *y = tmp;
}

int main(char* args[])
{
    int a = 10;
    int b = 20;
    swap(&a, &b);
    printf("%d, %d\n", a, b);
}
```

Memory management

Memory is manually managed:

```c
int* p = malloc(100 * sizeof(int));
...
free(p);
```

Challenges

- Pointer syntax
- Wild pointers
- Array index out of bounds
- Pointer arithmetic
- Memory leaks
- Use after free

Python

Python versions

Python 3 was released in 2008 but isn't backward compatible with Python 2. Both exist in parallel, but Python 2 is considered legacy. Some differences in Python 3:

- Generally cleaner
- print is a function
- Floating point division
- Better Unicode support
- Efficiency improvements

We'll use Python 3 (but note that many OS's default to Python 2).

Python

```python
print('Hello world!')
```

Python

Python can be run interactively in a read-eval-print loop (REPL):

```
$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license"
    for more information.
>>> 2+2
4
>>>
```

Python

Python doesn't require a main function, but it's common to use one:

```python
#!/usr/bin/env python3

def main():
    print('Hello world!')

if __name__ == '__main__':
    main()
```

Python

Python uses whitespace to delimit blocks:

```
def main():
    stmt1
    stmt2
    ...
    stmtN
```

Data types

- boolean (True, False)
- int
- float
- complex
- str
- bytes

If statements

```
if x > 50:
    print('x is large')
```

If statements

```python
if x > 50:
    print('x is large')
else:
    print('x is small')
```

If statements

```
if x > 50:
    print('x is large')
elif x > 10:
    print('x is medium')
else:
    print('x is small')
```

Input

To read values from the user, use the input function:

```python
name = input("What's your name? ")
print('Hi, ' + name + '!')
```

Note: value read will be a str.

Type conversion

- str(x)
- int(x)
- float(x)
- ...

Lists and tuples

```python
primes = [2, 3, 5, 7, 11, 13, 17, 19]
origin = (0, 0, 0)
```

But lists are mutable:

```python
primes[0] = 1
primes.append(23)
```

Loops

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

```python
for i in range(5, 7):
    print(i)
```

Loops

```python
for i in range(3):
    for j in range(4):
        print(i, j)
```

List comprehensions

```
[ i**2 for i in range(5) ]

[ i**2 for i in range(5) if i % 2 == 1]
```

Dictionaries

```
states = { 'New Jersey': 'NJ',
    'Pennsylvania': 'PA',
    'New York': 'NY' }

abbrev = states['New York']
```

References

Note that variables store references to objects, like Java:

```python
x = [1, 2, 3]
y = x
x.append(4)
print(y)
```

References

Note that variables store references to objects, like Java:

```python
x = [1, 2, 3]
y = x.copy()
x.append(4)
print(y)
```