C#スクリプト

変数の宣言　型名　変数名;

変数の代入法　変数名　＝　代入する値;

変数の初期化　型名　変数名　＝　代入する値;

変数の代入　変数名　＝　代入する変数名;

文字列の代入　変数名　＝　“代入する文字列”;

計算結果の代入　変数名　＝　数値　＋　数値;

float height = 160.5f

浮動小数点型

変数名　＋＝　数値

変数名　＋＋

変数名　＝　変数名　＋　数値

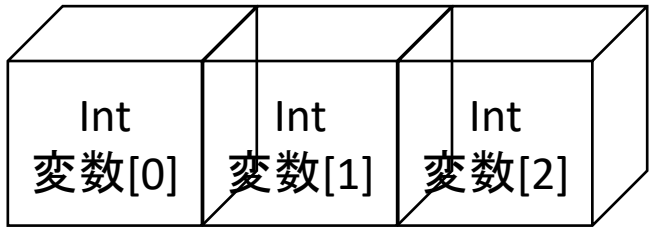
変数名　＝　変数名　＋　1

整数型の配列

new　作るという意味

int[ ] 変数　＝　new int[数値]

int[ ] 変数　＝　new int[2]のとき



int[ ]　array　＝ {数値,数値,数値,数値,数値}

int[ ]　array　＝ {3,5,2,7,4}のような場合はnewは書かない

int sum　→整数型

int points.Length　→整数型

浮動小数点型

float average　=　sum / points.Length　→整数型

float average　=　1.0f * sum / points.Length　→小数

メソッド　　意味のある処理ブロック

引数→　　メソッド　　→返り値

ひきすう
複数　　1つ

値を返さないメソッド　　空白（引数なし）

返り値の型　メソッド名（型　引数、型　引数）

```
void Start()
{
        int answer ;
        answer  =  Add(3, 4) ;
}
```

```
int  Add(int a , int b)
{
        int  c = a + b ;
        return c ;
}
```

←メソッドの処理

返り値

値を返さないメソッド　空白（引数なし）

```
void Start()
{
    Sayhello();
}
```

メソッドの呼び出し

引数なし

返り値なし

メソッド名 ( )

```
void Sayhello()
{
    Debug.Log(“Hello”);
}
```

←メソッドの処理

値を返さないメソッド　空白（引数なし）

```
void Start()
{
    Callname(“Tom”);
}
```

メソッドの呼び出し

引数1つ

返り値なし

メソッド名（string型　引数）

```
void Callname(string name)
{
    Debug.Log(“Hello” + name);
}
```

↑ Printのコード

↑ メソッドの処理

クラス　　メソッドと変数をまとめたもの

Hp：　ヒットポイント

```
class　クラス名
{
        メンバ変数の宣言;
        メンバメソッドの実装;
}
```

変数
Hp、　Power

メソッド
攻撃、防御、・・

プレイヤ

インスタンス（実体）

Player
my Player;

数字

Int
Num;

my Player. Attack() ;

my Player. Damage() ;

○○.△△

○○ クラスが持つ△△メソッド（または変数）を使っている

他のクラスから呼び出せる

```
public class Player  ←Player クラスの宣言
{
    他のクラスから呼び出せない
    private int hp = 100 ;
    private int power = 50 ;      } メンバ変数宣言

    public void Attack()   Attackメンバメソッド
    {
        debug.Log(this.power + “のダメージを与えた”) ;
    }

    public void Damage(int damage)
    {
        this.hp −= damage ;
        debug.Log(damage + “のダメージを受けた”) ;
    }
}
```

呼び出せる

Damageメンバメソッド

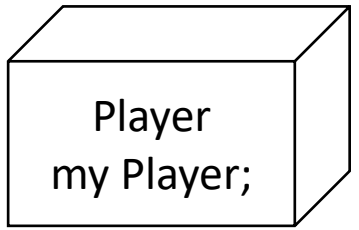
自分自身のインスタンス

継承

```
public class Test : Monobehaviour
{
    void Start()
    {
        Player myPlayer = new Player();
        myPlayer.Attack() ;
        myPlayer.Damage(30) ;
    }
}
```

インスタンス生成

Player型のmyPlayer変数宣言



箱生成

実行結果

50のダメージを与えた
30のダメージを受けた