double型 64bit = 8byteで小数点数

$$\zeta(s) = \sum_{k=1}^{n} \frac{1}{k^s}$$

$s=3$. $\zeta(3)$ Apery定数.

C言語

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

double myzeta(double s, int n){
    double z;
    int k;
    z = 0;
    for (k=1; k<n; k++){
        z = z + 1/pow(k,s);
    }
    return z;
}
int main(){
    printf("%lf\n", myzeta(3,100));
}
```

- pow で乗数を使う (sin, cos, …)
- 近似の度合 (int n)
- 局所変数 (double z; int k;)
- $k = 1, 2, 3, \cdots, n-1$
- ① 右辺を計算 ② 左辺に代入
- pow(k,s) → k^s
- 関数定義
- for文でうごかす (100)
- %0.10lf ・上から10ケタ
- %lf double型

C言語 1972年
C90 1990年 むかし

完全に理解するのがやさしい。
byte, アドレス, … メモリ 1byte.

・ライブラリたくさん。 → C言語による。

・どうコンピューター内部でうごいてるかわからなくても簡単に書ける。

python版

```
def myzeta(s,n):
    z=0;
    for k in range(1,n):
        z=z+1/k**s;
    return z;

print(myzeta(3,100));
```

インデント、字下げで { } のかわり。

・変数の型なし

・宣言しなくても自動的に局所変数。

1991 python
2000 python 2 (version 2) ← つい最近まで、こっちがメジャー。
2008 python 3 (〃 3)

R版 (統計用)

引数

myzeta(3) nを省略すると、10になる。

```
myzeta <- funtion(s, n=10) {
    z <- 0;
    for k in seq(1:n-1) {
        z <- z + 1/k^s;
    }
    return (z);
}
print(myzeta(3,100));
```

return (z); → zを戻す。

基本の考え方

① 関数のくみあわせでプログラム

② 局所変数

1980年代 → ③ Object 指向

◎ myzetaのweb pageをざっとみる。

◎ movie.

コンパイラ ← プログラム言語 → マシン語? (コンパイラ) ↓ 実行

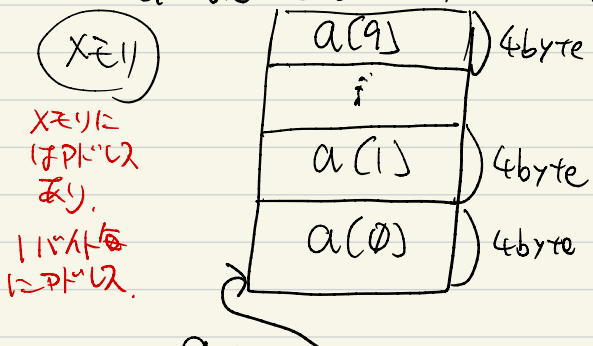
インタプリタ ← プログラム言語をよみながら、実行するプログラム。

notebook interface.

配列 (array)　　ベクトルみたいなもの。

int a[10];　　サイズ10の配列。
a[0], a[1], …, a[9]　変数として使える。

double b[20];　　サイズ20のdoubleの配列



int a[10]　40byte
メモリを使う。

練習　ベクトルの外積。

Vsum(x, y, z, 3)　(くわしくは、C言語の本。)

基礎がイケる。③　中級編。　Object 指向。

Object,　class method　←"関数みたいなもの"
instance method

Java.
2D点　$\begin{pmatrix} x \\ y \end{pmatrix}$

色情報付きの2D点。　$\begin{pmatrix} x \\ y \end{pmatrix}$ color.

classには、おやこ関係あり。
こみこみ。すべての親。

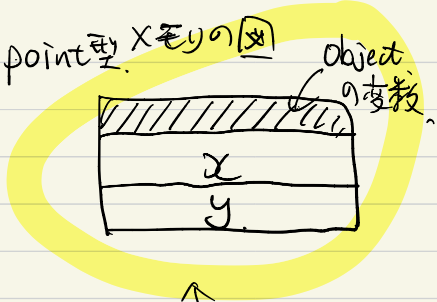
```
public class point extends Object {
    int x;
    int y;
    public static void main(...) {
        point p = new point();
    }
}
```

point → 親 (Object)

static ← クラスmethod

new point() ← point型の instance objectを生成!（C言語のmalloc相当をやってる）

point型 X 型の図



↑ このかたまりを instance.object とよぶ。

①
```
public void output() {
    int p,q;
    p = this.x; q = this.y;
    System.out.print("x=" + p + "y=" + q + "\n");
}
```
インスタンス method

②
```
p.output();
```
インスタンスpに、method outputをてきよう!

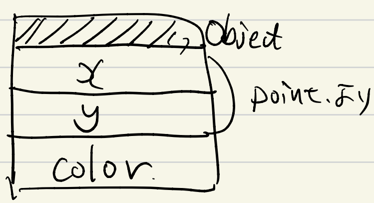
this は p.

pのclassにそのmethodなければ親classでさがして実行。

```
public class coloredPoint extends point {
    int color;
    public void output() {
        super.output();
        System.out.print("color=" + this.color);
    }
}
```

super.output(); ← 自分の親の outputをよぶ。

coloredPointのインスタンスobjectは、



- いろんな意味あり

(インスタンス). (インスタンスmethod)

(インスタンス). (メンバ変数, x, y, colorなど)

(クラス名). (クラスmethod) ← static宣言

⋮

javac point.java

生成 → a.out 相当

point.class

(Java仮想マシンのマシン語)

java point