Lecture 11

プログラミング演習 I　その11

本日の演習の流れ

ポインタの利用について

Jupyter Notebookの利用について

course page

http://amth.mind.meiji.ac.jp/courses/PE1/

ポインタ（pointer）とは

コンピュータのメモリは1バイト（byte）ごとに区切られ，各区画には一連番号がつけられている．これを**アドレス**（address）という．

C言語の特徴の一つは，変数と関数のアドレスを直接取り扱うことができることである．このために**ポインタ**が設けられている．

ポインタは，（よくいえば）C言語の醍醐味である．すべてのC言語経験者がポインタの習得で苦労している．逆に，楽勝だったという人は皆無のはずである．

Why?

・ポインタを使うことによって，プログラムが作りやすくなる．

・プログラムの実行速度が速くなるメリットがある．

ポインタ（pointer）とは

1 次のプログラムを実行せよ.

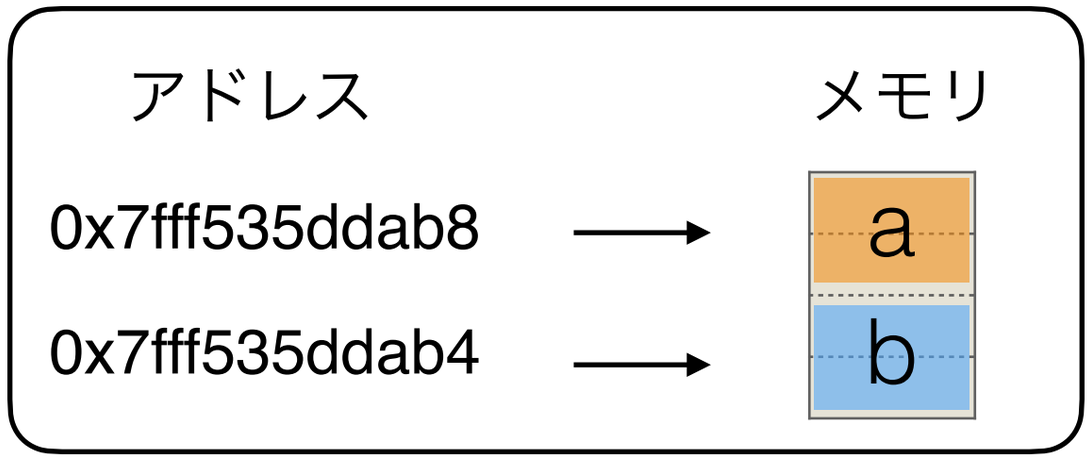
```c
#include <stdio.h>

int main(void){
  int a,b;
  printf("Input a and b ");
  scanf("%d %d",&a,&b);
  printf("a is %d.  Its address is %p\n",a,&a);
  printf("b is %d.  Its address is %p\n",b,&b);
  return 0;
}
```

ポインタ（pointer）とは

解説

```
1  #include <stdio.h>
2
3  int main(void){
4    int a,b;
5    printf("Input a and b ");
6    scanf("%d %d",&a,&b);
7    printf("a is %d.  Its address is %p\n",a,&a);
8    printf("b is %d.  Its address is %p\n",b,&b);
9    return 0;
10 }
```

アドレスを表示するときには書式 %p を使用する.



scanf関数を使って整数型変数に整数を入力するとき,

scanf(“%d %d”, &a, &b);

と変数 a に & が付いていた. &aは変数aのアドレスを示し, このアドレスの領域に数値を入力することになる. この「&」を**アドレス演算子**という.

次の printf 関数

printf(“a is %d. Its address is %p\n”, a, &a);

において a は変数 a に格納された値, &a は変数 a のアドレスを示す.

ポインタ変数

変数に「&」をつけるとその変数のアドレスを取り出すことができる.

C言語では，変数のアドレスを格納する変数も用意されている． これは**ポインタ変数(pointer variable)**という.

2 次のプログラムを実行せよ.

```c
#include <stdio.h>

int main(void){
  double a1 = 3.14159;
  double b1 = 1.73205;
  double *a2, *b2;
  a2 = &a1;
  b2 = &b1;
  printf("a1 address = %p Its value is %lf\n",&a1,a1);
  printf("a2 address = %p Its value is %lf\n",a2,*a2);
  printf("b1 address = %p Its value is %lf\n",&b1,b1);
  printf("b2 address = %p Its value is %lf\n",b2,*b2);
  return 0;
}
```

ポインタ変数

ポインタ変数

解説

ポインタ変数 b2 に * をつけた *b2 を出力すると，ポインタ変数 b2 で指示されたアドレスに格納されたデータが表示された．すなわち，ポインタ変数に「*」をつけると，ポインタ変数が指し示すアドレスに格納されたデータ (値) を間接的に参照できる．この「*」を間接参照演算子 (indirection or *dereference* operator) という．

```c
#include <stdio.h>

int main(void){
  double a1 = 3.14159;
  double b1 = 1.73205;
  double *a2, *b2;
  a2 = &a1;
  b2 = &b1;
  printf("a1 address = %p Its value is %lf\n",&a1,a1);
  printf("a2 address = %p Its value is %lf\n",a2,*a2);
  printf("b1 address = %p Its value is %lf\n",&b1,b1);
  printf("b2 address = %p Its value is %lf\n",b2,*b2);
  return 0;
}
```

//printf(“%p %lf\n”,a2,*&a1); と同じ

//printf(“%p %lf\n”,b2,b1); と同じ

出力

```
a1 address = 0x7ffee73a9ab0 Its value is 3.141590
a2 address = 0x7ffee73a9ab0 Its value is 3.141590
b1 address = 0x7ffee73a9aa8 Its value is 1.732050
b2 address = 0x7ffee73a9aa8 Its value is 1.732050
```

ポインタ変数

3 次のプログラムは2つの整数a, b を入力し，a > bのときにはaとbを入れ替え，a ≦ bのときには a + b の計算を行うプログラムである．　まずはaを30とbを20とする．

```c
#include <stdio.h>
int main(void){
    int a,b,c,d;
    int *x,*y,*temp, *z;
    scanf("%d %d",&a,&b);
    printf("a= %d b = %d\n",a,b);
    x = &a;
    y = &b;
    temp = &c;
    z = &d;
    if(*x > *y){
        *temp = *x;  //  c = a;
        *x = *y;    // a = b;
        *y = *temp; // b = c;
        printf("a=%d b = %d\n",*x,*y);
        printf("a=%d b = %d\n",a,b);
    }else{
        *z = *x+*y; //d = a+b;
        printf("%d + %d = %d\n",*x,*y,*z);
    }
    printf("but the value of d is:   %d\n",d);
    return 0;
}
```

//d = *x + *y; と同じ

出力

```
10
20
a= 10 b = 20
10 + 20 = 30
but the value of d is:   30
```

C言語の醍醐味

4 次のプログラムの出力はどうなるか確認せよ.

```c
#include <stdio.h>

int main(void){
  int a = 100;
  int b = 200;
  int c;
  int *pa, *pb, *pc;
  pa = &a;
  pb = &b;
  *pc = *pa + *pb;
  c = *pc;
  printf("c: %d\n",c);
  return 0;
}
```

ヒント プログラムにはバグを意図的に混入してある.

ポインタによる引数の引き渡し

5 以下のプログラムを実行してみよう.

```c
#include <stdio.h>

void saidai(int *a, int *b, int *c, int *maxm){
  if(*a >= *b && *a >= *c){
    *maxm = *a;
  }else if(*b >= *c){
    *maxm = *b;
  }else{
    *maxm = *c;
  }
}

int main(void){
  int x,y,z,result;
  printf("Enter x,y, and z.\n");
  scanf("%d %d %d",&x,&y,&z);
  saidai(&x,&y,&z,&result);
  printf("maximum value: %d\n",result);
  return 0;
}
```

出力

```
Enter x,y, and z.
8
7
21
maximum value: 21
```

解説

このプログラムでは，&x,&y,&zから*a,*b,*cに値を引き渡している.

&x は，変数 x のアドレスを示す．仮引数 *a はポインタ変数で，&x が示すアドレスの内容を示す．したがって，データの引き渡しは，データが格納されている領域のアドレスで引き渡されている.

ポインタと配列

6 以下のプログラムを実行してみよう.

```c
#include <stdio.h>

int main(void){
  int x[6]={10, 20, 30, 40, 50, 0};
  int *p;
  for(p=x; *p != 0; p++){
    printf("%d ",*p);
  }
  printf("\n");
  return 0;
}
```

出力

```
10 20 30 40 50
```

解説

- p=x：ポインタ変数 p に配列 x の先頭アドレスを代入している. すなわち, 配列名 x は配列 x の先頭アドレスを表している. ここでp=&x[0]としてもよい.
- *p!=0：ポインタ変数 p で指示される配列の要素が 0 でない間, 繰り返す.
- p++：ポインタ変数 p を 1 ずつ増やす. すなわち, アドレスを進める.

x[0]を*pで表したとき, 次の要素x[1]は, *(p+1)となる. ここで p+1 は, pのアドレスに要素1つ分のアドレスを加えるということである. （pの値に1を足すということではない）

ポインタと配列

7 整数型配列dに次の10個の整数(1,2,3,4,5,6,7,8,9,10)が格納されている．これを次のように表示するプログラムを作成しなさい．ただし，ポインタを使用せよ．

```
10 9 8 7 6 5 4 3 2 1
```

ヒント

・ p = &d[9];

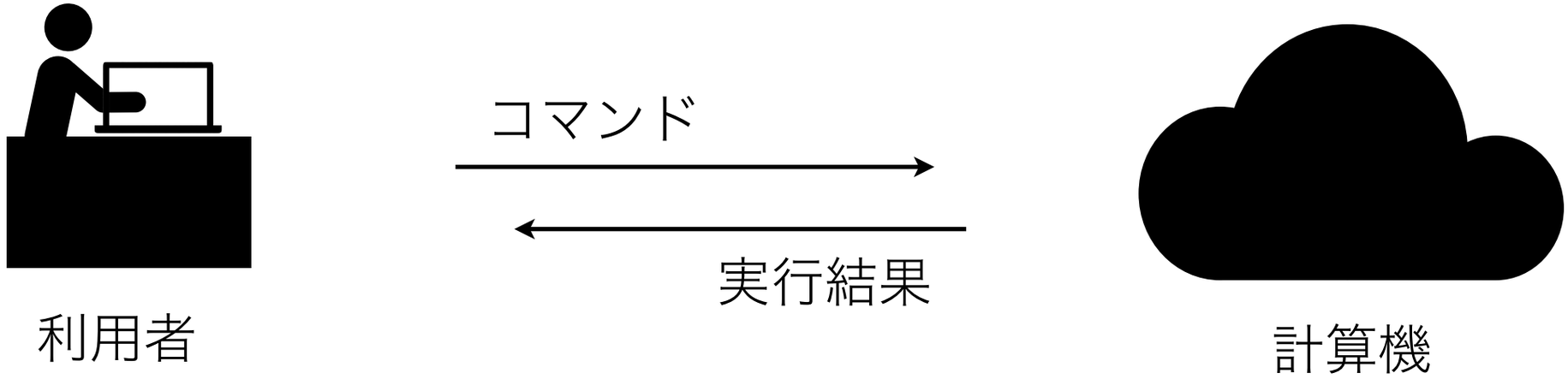
・ 配列 d の後尾の要素のアドレスを ポインタ変数 p に代入し，*(p-i)で配列の要素を指定する.

Jupyter Notebookの利用について

Jupyter Notebookとは

・Jupyter Notebookは, Webブラウザを利用した対話的コンピューティング環境である.

・Jupyter Notebookでは, Pythonで書かれたプログラムの呼び出しや, データの操作により対話的コンピューティングを行う.



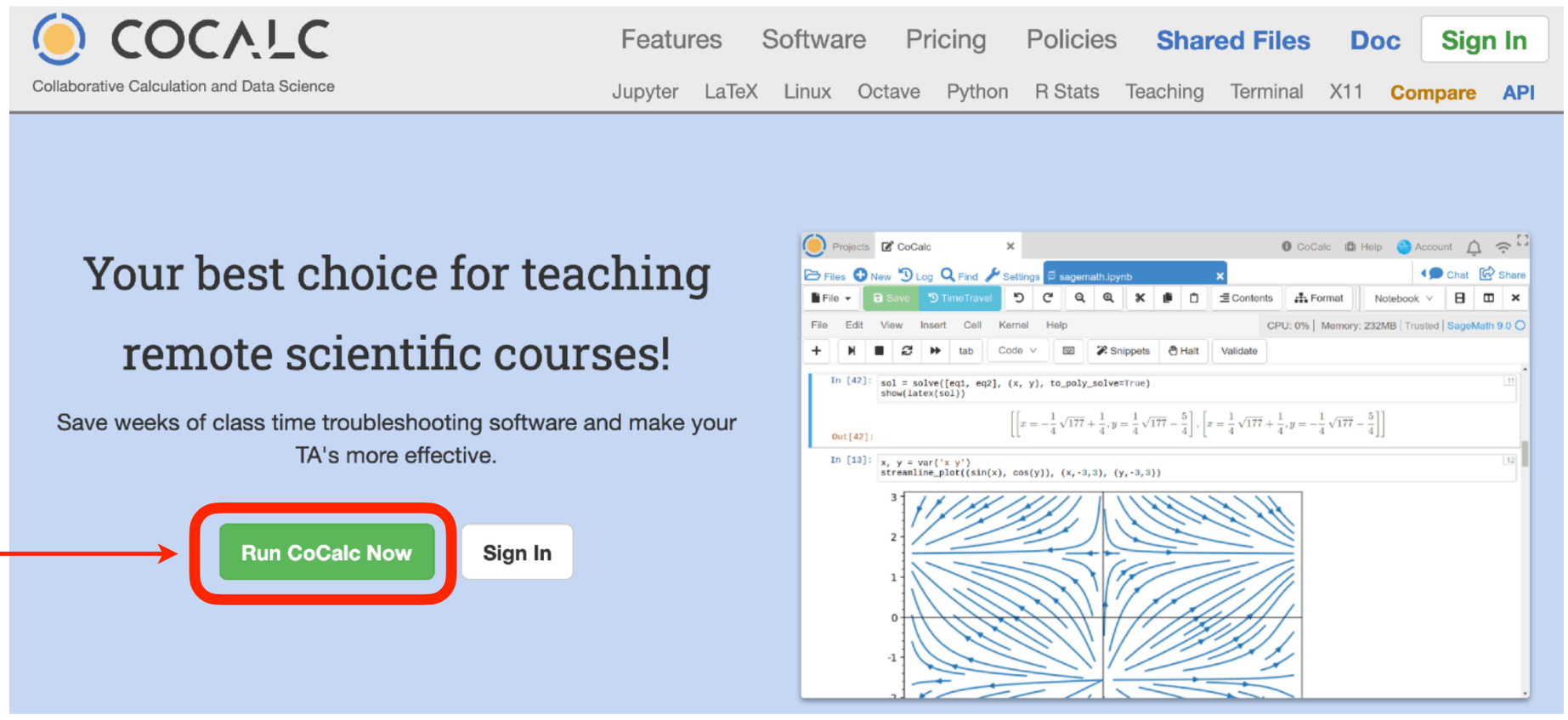
Jupyter Notebookの利用について

Jupyter Notebookの使い方

1．以下のリンク(cocalc.com)を開く.

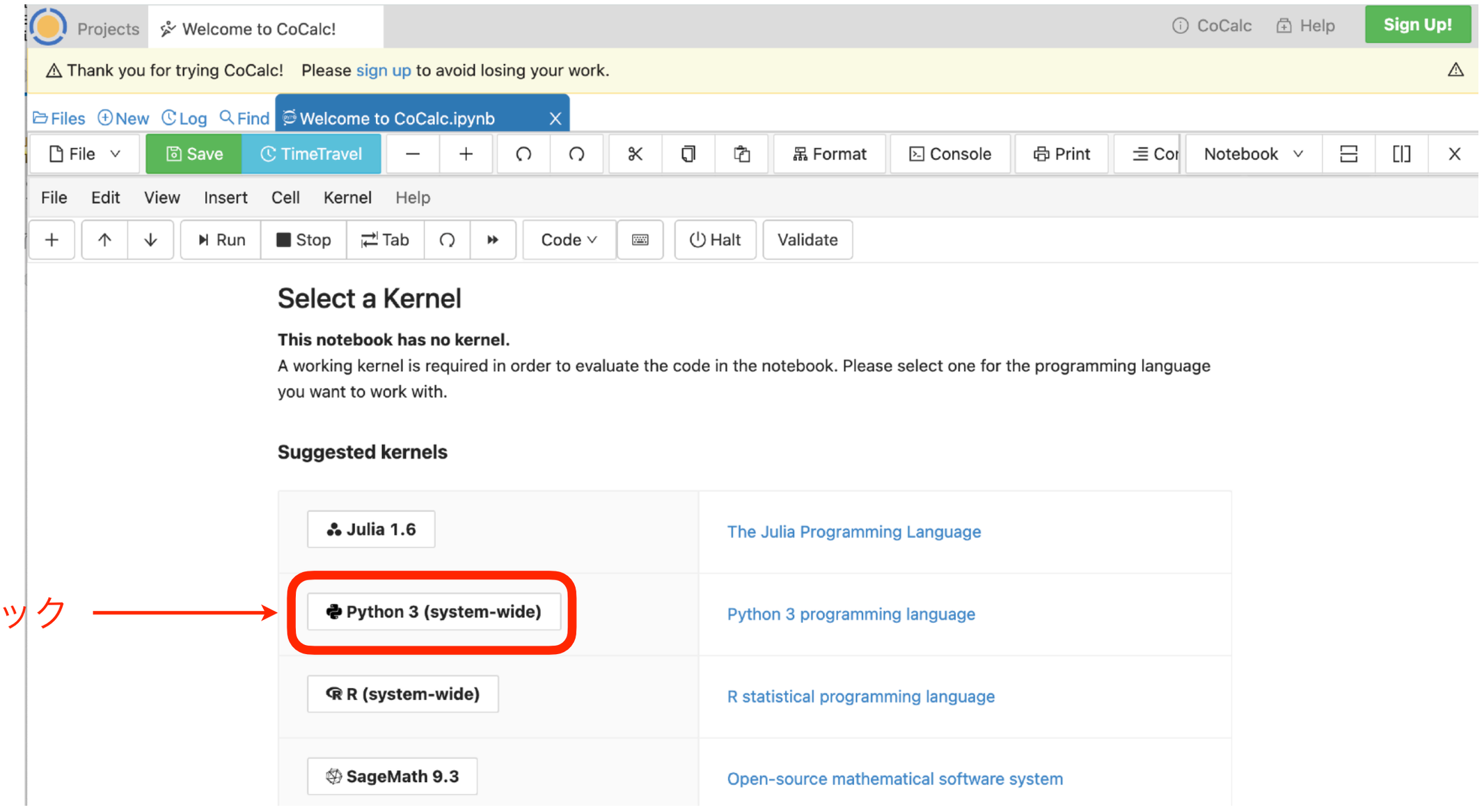
https://cocalc.com/

2. 「Run CoCalc Now」をクリック



Jupyter Notebookの利用について

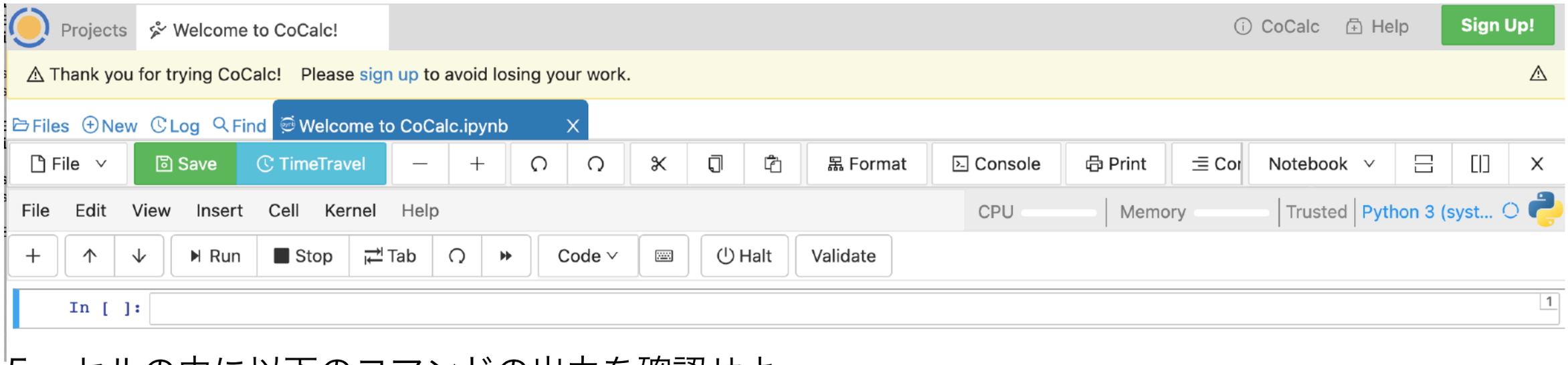
3. 「Python」を選択する



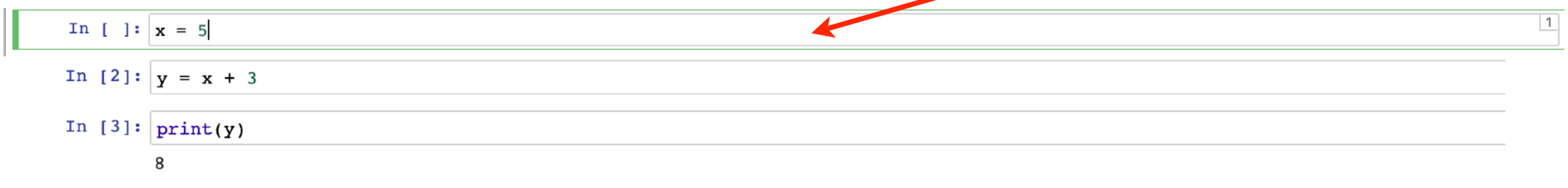
クリック →

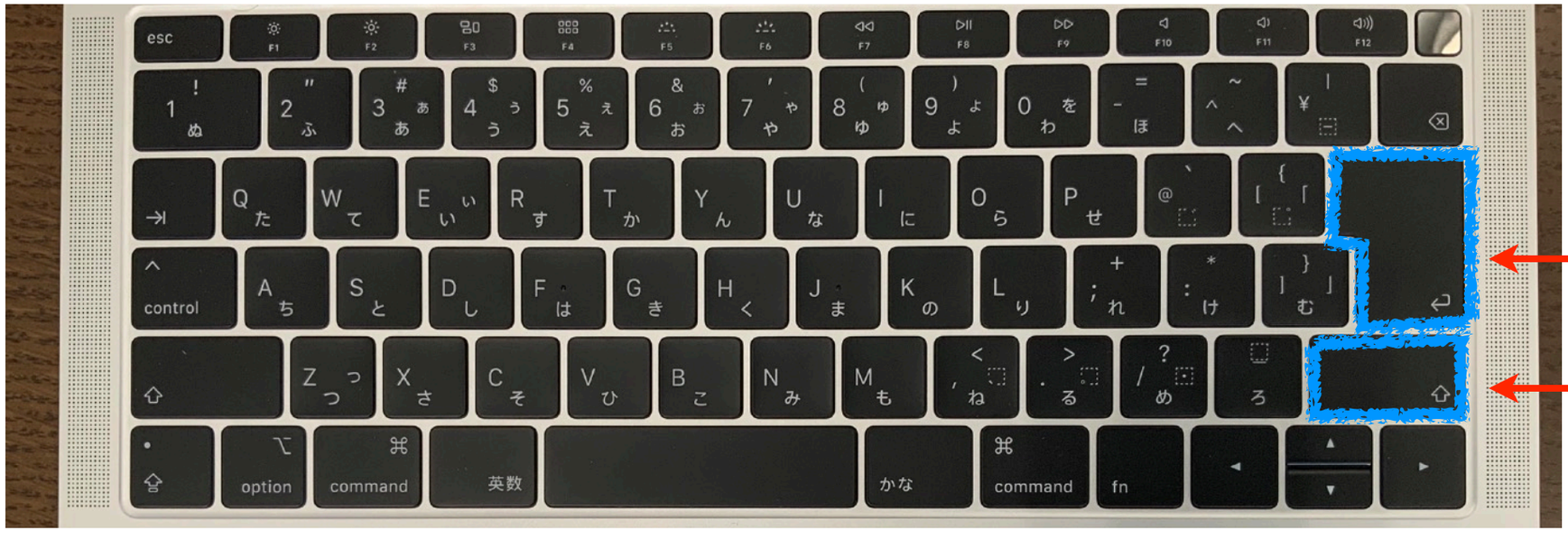
Jupyter Notebookの利用について

4．以下の画面が表示される．



5．セルの中に以下のコマンドの出力を確認せよ．





Jupyter Notebookの利用について

以下のコマンドの出力を確認せよ.

8

In [4]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
x = np.random.rand(100)
plt.plot(x)
```