

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勞している。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.

```
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 }
```

ポインタ (pointer) とは

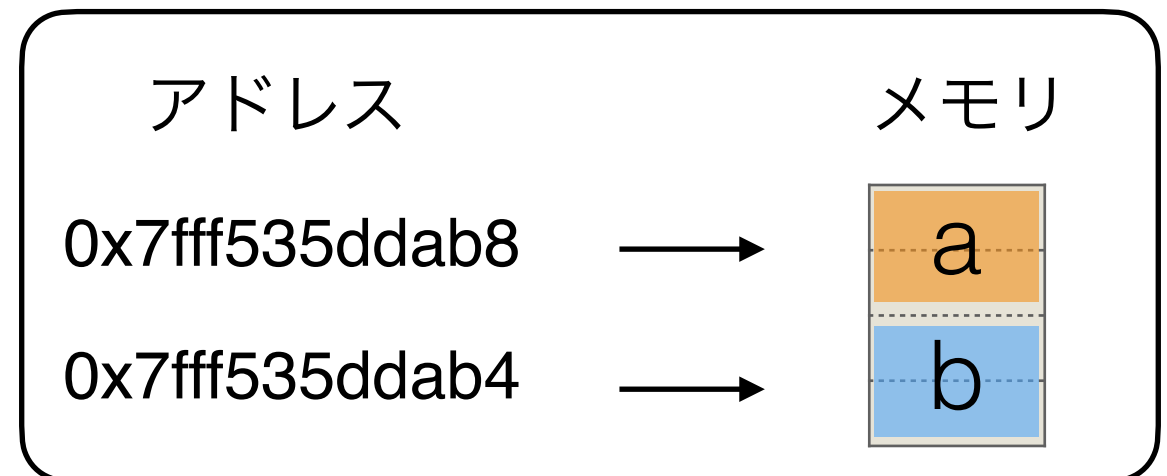
解説

```
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。

```
1 #include <stdio.h>
2
3 int main(void){
4     double a1 = 3.14159;
5     double b1 = 1.73205;
6     double *a2, *b2;
7     a2 = &a1;
8     b2 = &b1;
9     printf("a1 address = %p Its value is %lf\n",&a1,a1);
10    printf("a2 address = %p Its value is %lf\n",a2,*a2);
11    printf("b1 address = %p Its value is %lf\n",&b1,b1);
12    printf("b2 address = %p Its value is %lf\n",b2,*b2);
13    return 0;
14 }
```

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。

```
1 #include <stdio.h>
2
3 int main(void){
4     double a1 = 3.14159;
5     double b1 = 1.73205;
6     double *a2, *b2;
7     a2 = &a1;
8     b2 = &b1;
9     printf("a1 address = %p Its value is %lf\n",&a1,a1);
10    printf("a2 address = %p Its value is %lf\n",a2,*a2);
11    printf("b1 address = %p Its value is %lf\n",&b1,b1);
12    printf("b2 address = %p Its value is %lf\n",b2,*b2);
13    return 0;
14 }
```

→ //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, $a \leq b$ 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.

```
1 #include <stdio.h>
2 int main(void){
3     int a,b,c,d;
4     int *x,*y,*temp, *z;
5     scanf("%d %d",&a,&b);
6     printf("a= %d b = %d\n",a,b);
7     x = &a;
8     y = &b;
9     temp = &c;
10    z = &d;
11    if(*x > *y){
12        *temp = *x; // c = a;
13        *x = *y; // a = b;
14        *y = *temp; // b = c;
15        printf("a=%d b = %d\n",*x,*y);
16        printf("a=%d b = %d\n",a,b);
17    }else{
18        *z = *x+*y; //d = a+b;
19        printf("%d + %d = %d\n",*x,*y,*z);
20    }
21    printf("but the value of d is: %d\n",d);
22    return 0;
23 }
```

出力

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

← //d = *x + *y; と同じ

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.

```
1 #include <stdio.h>
2
3 int main(void){
4     int a = 100;
5     int b = 200;
6     int c;
7     int *pa, *pb, *pc;
8     pa = &a;
9     pb = &b;
10    *pc = *pa + *pb;
11    c = *pc;
12    printf("c: %d\n",c);
13    return 0;
14 }
```

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.

```
1 #include <stdio.h>
2
3 void saidai(int *a, int *b, int *c, int *maxm){
4     if(*a >= *b && *a >= *c){
5         *maxm = *a;
6     }else if(*b >= *c){
7         *maxm = *b;
8     }else{
9         *maxm = *c;
10    }
11 }
12
13 int main(void){
14     int x,y,z,result;
15     printf("Enter x,y, and z.\n");
16     scanf("%d %d %d",&x,&y,&z);
17     saidai(&x,&y,&z,&result);
18     printf("maximum value: %d\n",result);
19     return 0;
20 }
```

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.

```
1 #include <stdio.h>
2
3 int main(void){
4     int x[6]={10, 20, 30, 40, 50, 0};
5     int *p;
6     for(p=x; *p != 0; p++){
7         printf("%d ",*p);
8     }
9     printf("\n");
10    return 0;
11 }
```

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

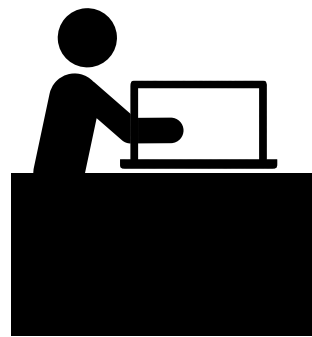
ヒント

- `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。

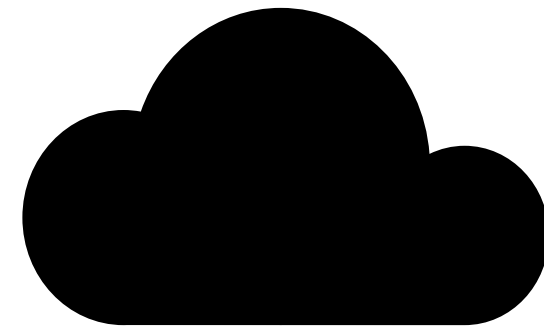
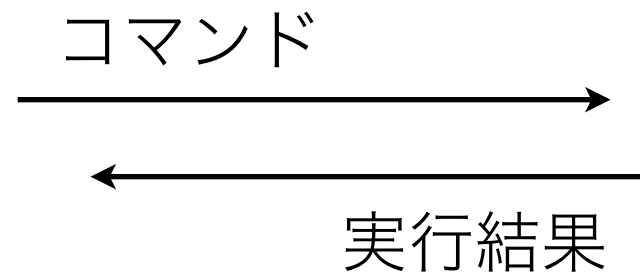
Jupyter Notebookの利用について

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.



利用者



計算機

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

ここ！

A screenshot of the CoCalc website. The header includes the CoCalc logo and navigation links: Features, Software, Pricing, Policies, Shared Files, Doc, and Sign In. Below the header, there's a main section with the text "Your best choice for teaching remote scientific courses!" and "Save weeks of class time troubleshooting software and make your TA's more effective." At the bottom of this section, there are two buttons: "Run CoCalc Now" (highlighted with a red rectangle and an arrow from the text "ここ！") and "Sign In". To the right of the main text, there's a preview of a Jupyter Notebook interface showing code and a plot.

Jupyter Notebookの利用について

3. 「Python」を選択する

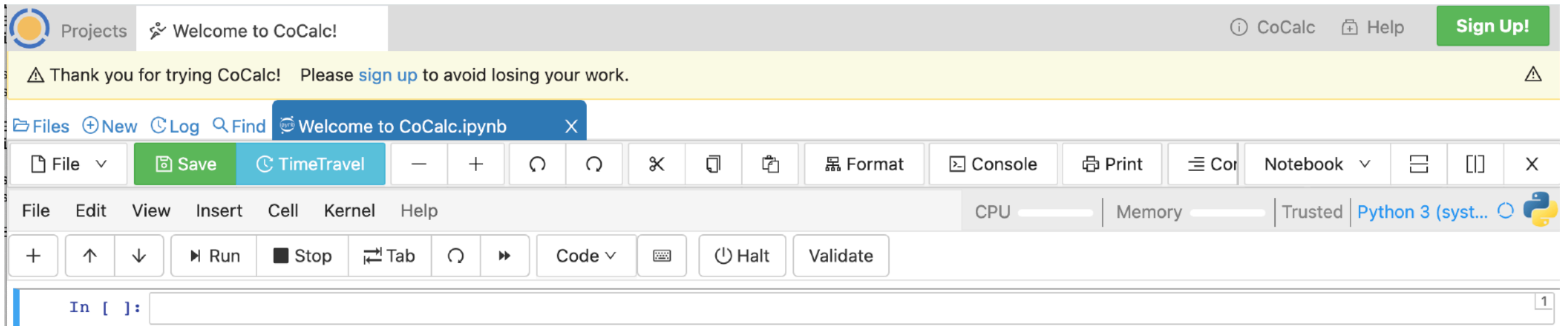
The screenshot shows the CoCalc Jupyter Notebook interface. At the top, there's a header with 'Projects', 'Welcome to CoCalc!', and a 'Sign Up!' button. Below the header is a yellow banner with a warning: 'Thank you for trying CoCalc! Please sign up to avoid losing your work.' The main toolbar includes buttons for 'File', 'Save', 'TimeTravel', and various editing tools. The 'Kernel' menu is open, showing the 'Select a Kernel' dialog. The dialog text says: 'This notebook has no kernel. A working kernel is required in order to evaluate the code in the notebook. Please select one for the programming language you want to work with.' Below this, there's a section titled 'Suggested kernels' with a table of options:

Julia 1.6	The Julia Programming Language
Python 3 (system-wide)	Python 3 programming language
R (system-wide)	R statistical programming language
SageMath 9.3	Open-source mathematical software system

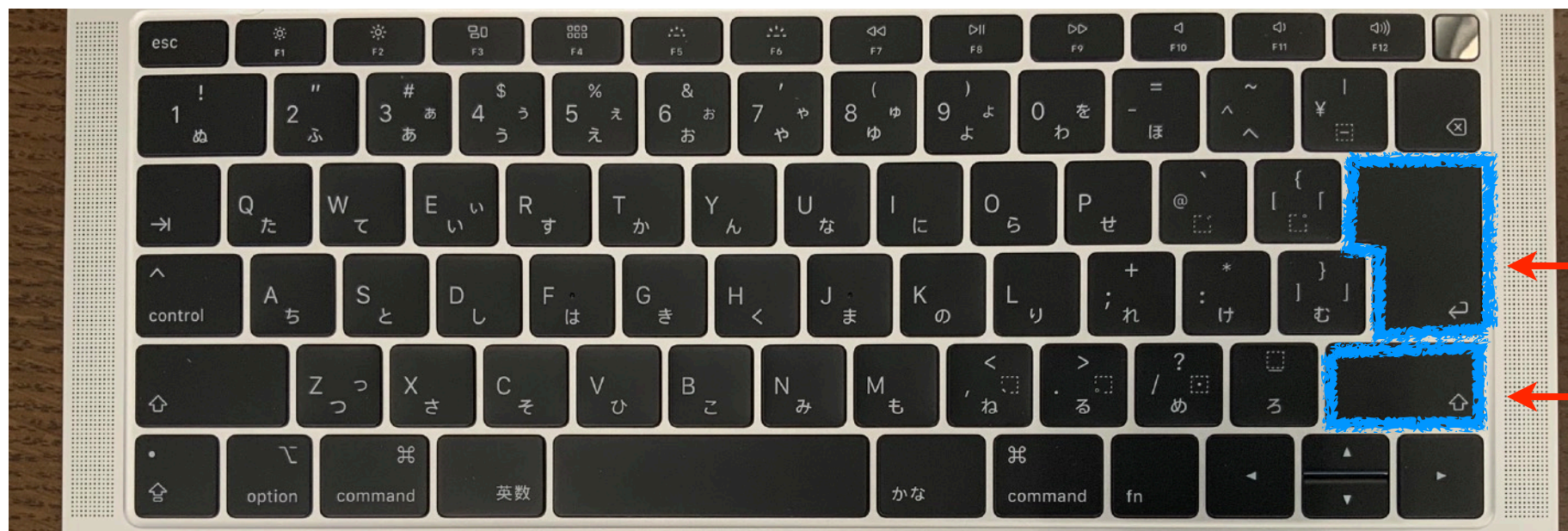
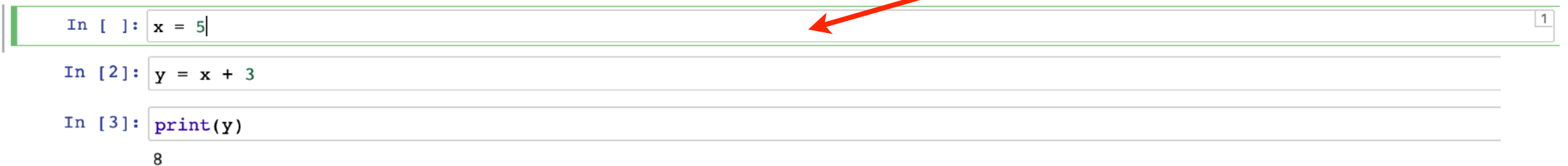
A red box highlights the 'Python 3 (system-wide)' option, and a red arrow points to it from the word 'クリック' (Click) written in red text.

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)
```