Pythonプログラム機能部品カード　No.01

文字や数値を表示する

画面に文字や数値を表示します。

部品
01

具体例

画面に Hello と表示する。

```
print("Hello")
```

画面に 3 + 7の計算結果を表示する。

```
print(3+7)
```

変数Xに数を入れた後に
画面に 変数Xの内容を表示する。

```
x = 25
print(x)
```

裏面

Pythonプログラム機能部品カード　No.01

文字や数値を表示する

図式例

表示する(Hello)

```
print("Hello")
```

表示する(3+7)

```
print(3+7)
```

Pythonプログラム機能部品カード　No.02

四則演算と変数への代入

足し算、引き算、掛け算、割り算の基本的な使い方と、数値や計算結果の変数への代入方法です。

部品
02

具体例

変数iの内容に1を足した値

```
i + 1
```

変数xと変数iをかけた値

```
x * i
```

変数xの内容から2を引いた値

```
x - 2
```

変数aを10で割った値

```
a / 10
```

変数 a, 変数b, 変数cの内容を足す

```
a + b + c
```

変数aの内容を0にする。

```
a = 0
```

変数Xの内容を変数Iに1を足したものにする。

```
x = a + 1
```

裏面

Pythonプログラム機能部品カード　No.02

四則演算と変数への代入

図式例

a = 0

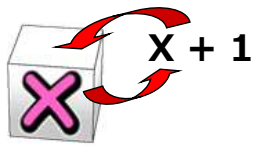
```
a = 0
```

x = x + 1

```
x = x + 1
```

x = 1

xと名前をつけた箱(変数)に1を入れる



x = x +1
(x ← x + 1)

初めにxの箱(変数)の中を取り出し+1する。計算結果をxの箱(変数)に入れなおす。

Pythonプログラム機能部品カード　No.03

キーボードから数値や文字の入力

キーボードから数値や文字を入力して変数に入れます。

部品 03

具体例

キーボードから入力した文字を変数xに入れる。

```
x = input()
```

キーボードから入力した数字を整数として変数aに入れる。

```
a = int(input())
```

Input()は文字として入力します。

例えば100とキーを打っても
100という数字として入力されます。

そこで、int( )を使用して数字→数値に変換して計算などに使用します。

裏面

Pythonプログラム機能部品カード　No.03

キーボードから数値や文字の入力

図式例

x = (入力)

```
x = input()
```

a = (数値入力)

```
a = int(input())
```

Pythonプログラム機能部品カード　No.04

条件を判断して、命令を実行

条件を判断して、条件が成り立つ時に命令を実行します。

具体例

部品 04

変数aの内容が70より大きければ、合格(Goukaku)と表示する。

```
if a > 70:
    print("Goukaku")
```

変数aの内容が、変数bの内容より小さければ、変数xに変数bの内容を入れる。

```
if a < b:
    x = b
```

x == y	x と y が等しい
x != y	x と y が等しくない
x > y	x は y よりも大きい
x < y	x は y よりも小さい
x >= y	x は y と等しいか大きい
x <= y	x は y と等しいか小さい

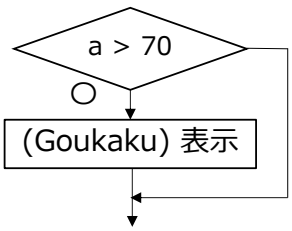
裏面

Pythonプログラム機能部品カード　No.03

条件を判断して、命令を実行

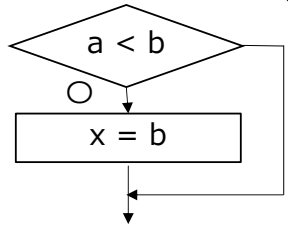
図式例

? a > 70:
○: 表示する(Goukaku)




```
if a > 70:
    print("Goukaku")
```

? a < b :
○: x = b




```
if a < b:
    x = b
```

Pythonプログラム機能部品カード　No.05

条件を判断して、〇と×で違う命令を実行

条件を判断して、条件が成り立つ時と成り立たない時に別々の命令を実行します。

部品 05

具体例

変数xの内容が19より大きいか判断して。大きければ成人(Seijin)と表示し、そうでなければ、未成年(Miseinen)と表示する。

```
if x > 19:
    print("Seijin")
else:
    print("Miseinen")
```

変数aの内容が変数bの内容より大きいか判断して。大きければ変数xに変数aの内容を入れ、そうでなければ、変数xに変数bの内容を入れる

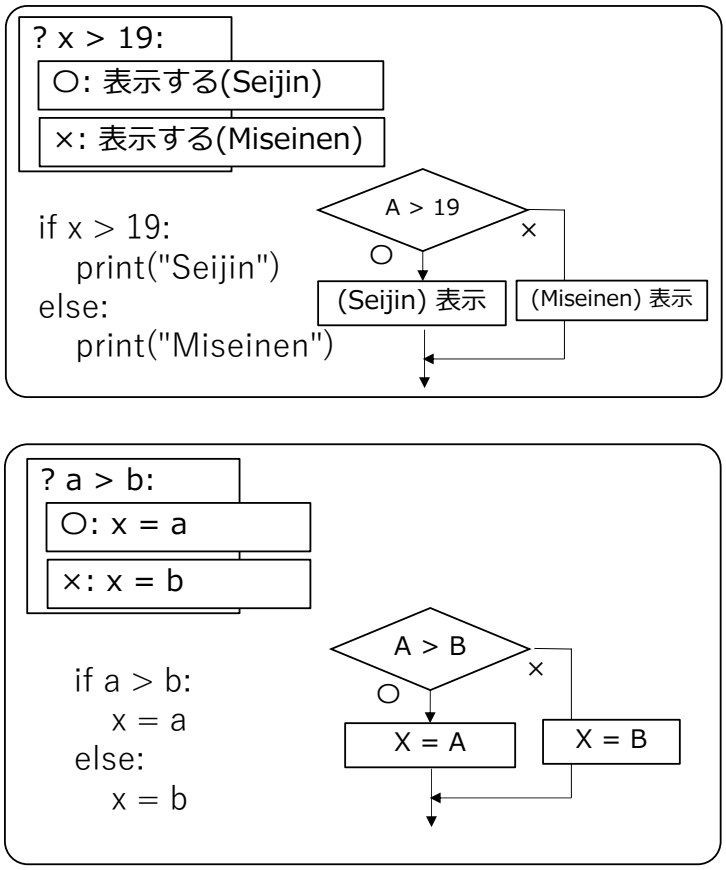
```
if a > b:
    x = a
else:
    x = b
```

裏面

Pythonプログラム機能部品カード　No.05

条件を判断して、〇と×で違う命令を実行

図式例



Pythonプログラム機能部品カード　No.06

複数の条件を判断して違う命令を実行

複数の条件を判断して、違う命令を実行します。
No.04やNo.05で、実行する命令として、No.04やNo.05自体を使用します。

部品 06

具体例

```
if a < 6:
    print("Youji")
else:
```

```
if a < 12:
    print("Kodomo")
else:
    print("Otona")
```

条件を判断する二つの部品を組み合わせます。

```
if a < 6:
    print("Youji")
else:
    if a < 12:
        print("Kodomo")
    else:
        print("Otona")
```

まず、変数aの内容が6未満であるか判断して、未満であれば、幼児(Youji)と表示します。

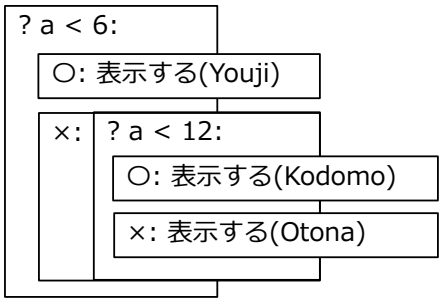
もし、そうでなければ、今度は、変数Aの内容が12未満であるか判断して、未満であれば、子供(Kodomo)と表示します。そうでなければ、大人(Otona)と表示します。

裏面

Pythonプログラム機能部品カード　No.05

複数の条件を判断して違う命令を実行

図式例




```
if a < 6:
    print("Youji")
else:
    if a < 12:
        print("Kodomo")
    else:
        print("Otona")
```

字下げしていることで、elseの中で実行する命令となっている。

Pythonプログラム機能部品カード　No.07

0からnまで繰り返す

ある変数の内容を0からnまでカウントアップしながら、繰り返して処理します。

部品 07

具体例

変数iの内容を0, 1, 2, 3 …と10までカウントアップして、処理を繰り返しています。

```
for i in range(11):
    print(i)
```

<- 変数iの内容を0,1,2,3, ・・,10まで変えて繰り返しで実行します。

繰り返しの中で実行する命令としてprint(i)で、iの内容を表示しています。

range(n)の意味
　0から始まってn以下の数まで、1づつカウントアップしたデータを作ります。

0,1,2,3,4,5,6,7,8,9,10

裏面

Pythonプログラム機能部品カード　No.07

0からnまで繰り返す

図式例

```
for i in range(11):
    print(i)
    print(i*2)
    print(i*3)
```

for i in range(11):
　繰り返しで実行する命令の集まり(変数iの利用含む)

i = [1,2,…,10]
繰り返す
　繰り返しで実行する命令の集まり(変iの利用含む)

補足

↑ 字下げしていることで、for命令の中で3個のprint命令が実行される。

Pythonプログラム機能部品カード　No.08

指定回数繰り返す:range()

ある変数の内容を初めの数から、終わりの数まで、指定した数だけでカウントアップしながら、繰り返して処理します。

部品 08

具体例

変数iの内容を1から10未満の数まで2づつカウントアップして、処理を繰り返しています。

```
for i in range(1,10,2):
    print(i)
```

<- 変数iの内容を1から10未満まで2づつ変えて繰り返しで実行します。

range(1,10,2):[1,3,5,7,9]

変数iの内容を10から100未満の数まで10づつカウントアップして、処理を繰り返しています。

```
for i in range(10,100,10):
    print(i)
```

<- 変数iの内容を10から100未満まで10づつ変えて繰り返しで実行します。

range(10,100,10):[10,20,30,・・,80,90]

裏面

Pythonプログラム機能部品カード　No.08

指定回数繰り返す:range()

Range()のパラメータ

range(終了値)

range(5):[0,1,2,3,4]

range(開始値, 終了値, 増分)

range(1,10,2):[1,3,5,7,9]

range(10,100,10):[10,20,30,・・,80,90]

Pythonプログラム機能部品カード　No.09

リストの処理(取り出し、入れ替え)

リストの中の個々の要素の取り出し、入れ替え、追加、クリアをします。

部品 09

具体例

d[3]	リストdの4番目の内容です。
d[i]	リストdの(iの変数の内容) の内容です。 例えばiに4が入っていたら5番目の内容です。

```
d[2] = 2
```

リストdの3番目の内容を2にします。

```
d[i] = x
```

リストdの(iの変数の内容) の内容を変数Xの内容にします。
例えばiに4、xに100か入っていたら5番目の内容を100にです。

裏面

Pythonプログラム機能部品カード　No.09

リストの処理(取り出し、入れ替え)

図式例

```
x = d[3]
```

```
x = d[3]
```

```
? d[i] < 10 :
  ○: (処理)
```

```
if d[i] < 10:
    (処理)
```

```
d[i] = 2
```

```
d[i] = 2
```

Pythonプログラム機能部品カード　No.10

二つの変数の内容を入れ替える

二つの変数の内容を入れ替えます。入れ替える時、片一方の変数の内容を一時的に他の変数に入れるのがポイントです。

部品 10

具体例

```
t = a
a = b
b = t
```

変数aと変数bの内容を入れ替えます。(変数aにbの内容を入れる前に、変数aの内容をtに入れて退避させています)

リストd[0]の箱の内容とd[i]の内容を入れ替えています。

```
t = d[0]
d[0] = d[i]
D[i] = d[0]
```

裏面

Pythonプログラム機能部品カード　No.010

二つの変数の内容を入れ替える

図式例

```
A <-> b
```

```
t = a
a = b
b = t
```

又は

```
t = a
a = b
b = t
```

```
d[0] <-> d[i]
```

```
t = d[0]
d[0] = d[i]
D[i] = t
```

又は

```
t = d[0]
d[0] = d[i]
d[i] = t
```

補足

Pythonの場合は、下記のようにしても入れ替えできます。

```
a, b = b, a
```

```
d[0], d[i] = d[i], d[0]
```

Pythonプログラム機能部品カード　No.11

二重の繰り返し

ある変数の内容をカウントアップする繰り返しを二つ組み合わせて使用します。

部品 11

具体例

```
for i in range(5):
    for j in range(5):
        print(d[i] * d[j])
```

繰り返しで実行する命令の集まり(変数iの利用含む)

変数iでカウントアップ　　変数Iでカウントアップ

繰り返しの二つの部品を組み合わせます。

```
for i in range(5):
    for j in range(5):
        print(d[i] * d[j])
```

変数iの内容が0,1,・・4とカウントアップして下記の変数jを使った繰り返しを5回繰り返します。

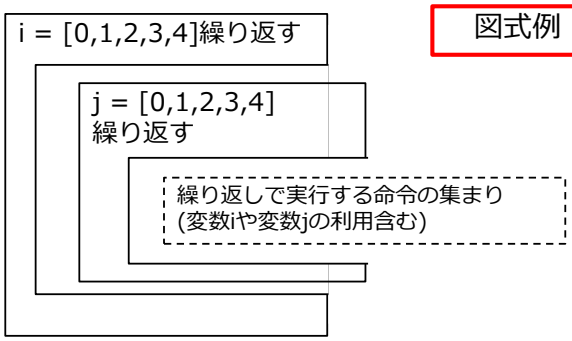
変数jの内容が0,1,・・4とカウントアップして表示します。

裏面

Pythonプログラム機能部品カード　No.11

二重の繰り返し

図式例




```
for i in range(5):
    for j in range(5):
```

繰り返しで実行する命令の集まり
(変数iや変数jの利用含む)

Pythonプログラム機能部品カード　No.12

二重の繰り返し(内の繰り返しの開始を変更)

ある変数の内容をカウントアップする繰り返しを二つ組み合わせて使用します。但し、内側のカウントアップする数の初めを外側の繰り返しの変数の内容にしています。

部品 12

具体例

```
for i in range(5):
    for j in range(i,5):
        print(d[i] * d[j])
```

繰り返しで実行する命令の集まり(変数iの利用含む)

変数iでカウントアップ　　変数iでカウントアップ
初めにiを設定(処理内はiから)

繰り返しの二つの部品を組み合わせます。

```
for i in range(5):
    for j in range(i,5):
        print(j)
```

変数iの内容が0,1, 2, 3, 4とカウントアップして下記の変数Jを使った繰り返しを5回繰り返します。

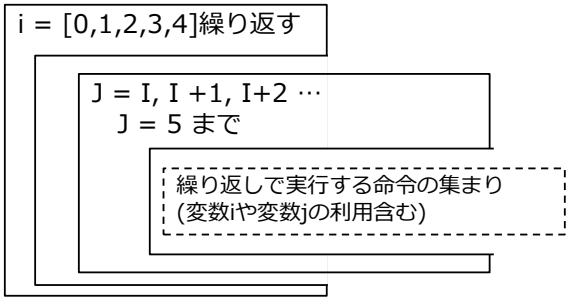
変数jの内容が、個々の繰り返しが始まった時のiの内容から始めるので
0, 1, 2, 3, 4,
1, 2, 3. 4
2, 3, 4
3, 4
4 とカウントアップして表示します。

裏面

Pythonプログラム機能部品カード　No.12

二重の繰り返し(中の繰り返しの開始を変更)

図式例



Range()の開始をiにするのがポイント

```
for i in range(5):
    for j in range(i,5):
```

繰り返しで実行する命令の集まり
(変数Iや変数Jの利用含む)