# 基本命令

## 変数

　pythonに変数の宣言はありません。ただし、aに何も代入していない状態で、b = a と書くと、エラーが発生するので、変数のスペルミスによるバグは発生しません。

　pythonの変数には型はありません。「数値」「文字列」のどちらも入ります。

a = 12

b = "34"

print(a)

print(b)

　変数aには数値12, 変数bには文字列 "34" を入れています。pythonでは文字列は " で囲む、あるいは ' で囲みます。ですから、

b = '34'

と書いても同じです。どちらがポピュラーかというと " の方がポピュラーなようです[[1]](#footnote-1)。

　変数の値を表示するには、print() 関数を使います。 () の中に表示したい変数を書きます。変数の中身は「文字列」「数値」のどちらでも構いません。2個同時に表示したいときは

print(a, b)

のように , で区切ります。, で区切ると半角スペースが1個入ります。分かりやすく表示するために、

print("a =", a, " b =", b)

と書くのがよいでしょう。

　イコールの前後に半角スペースを入れたくない場合は、表示する文字列を自力で作成します。以下のように書きます。

print("a=" + str(a) + " b=" + b)

　str(a) は数値12を文字列 "12" に変換します。「+」は文字列を連結します。

　ここで登場した「+」の意味について考えてみましょう。

a + b

はaとbがどちらも数値のときは足し算、aとbがどちらも文字列のときは連結を表します。どちらか片方が数値、もう片方が文字列の場合は、エラーが発生します。ここでは数値aを文字列に変換してから連結しています。

　この例の場合、aは数値、bは文字列でした。両方とも数値として足し算する場合、文字列として連結する場合は、それぞれ以下のように書きます。

c = str(a) + b # 文字列として連結

d = a + float(b) # 数値として足し算

print("c =", c, "d =", d)

　str(a) は数値aを文字列に変換します。float(b) は文字列bを実数に変換します。

　文字列を数値に変換する関数はint() もあります。次の例を見てください。

moji1 = "10"

moji2 = "10.2"

num1 = int(moji1)

num2 = int(moji2)

print(num1)

print(num2)

　このプログラムはnum2 = int(moji2) の実行時にエラーが発生します。実数を表す "10.2" のような文字列を整数に変換しようとすると、エラーが発生します[[2]](#footnote-2)。

【課題2‑1】

　変数aに数値 -23.4, 変数bに文字列 "abc" を代入し、以下のように画面に表示するプログラムを作成しなさい。

<a = -23.4> <b = abc>

## for文

　pythonのforループはExcel VBAやC言語とは少し異なります。変数iを0, 1, 2, 3と変化させるには以下のように書きます。

for i in range(0, 4):

print(i)

　上記のプログラムを実行すると、画面には0 1 2 3 と表示されます。range(a, b) と指定するとaからb-1まで1ずつ変化します。b より1つ小さい数までしか変化しないことに注意して下さい。ここでpythonに関する重要なルールを学びます。

### ルール

　forループは「ループの範囲がどこまでか？」を表すためのしるしが必要です。

　Basic (Excel VBA) ではForの範囲を表すのにNextという文を用いました。Cではforの範囲を表すのに中括弧 { 　　 } を用いました。pythonではforのようにブロックを要求する命令には命令の末尾に：（コロン）がつきます。そしてブロックの範囲をインデント（字下げ）で表します。

　字下げ幅を何文字にするかは、何文字でもかまいません。通常は4文字が用いられます。そして、字下げ幅は1つのブロックにおいて、同じ下げ幅でないといけません。

　ループのインデントについて確認しておきましょう。以下の2つのプログラムを実行して結果を比べて下さい。

for i in range(0, 4):

print("a")

print("b")

for i in range(0, 4):

print("a")

print("b")

　インデントの有無でループの範囲が決定されることが分かります。以下のプログラムはどうなるでしょうか。

print("1")

for i in range(0, 4):

print("a")

print("b")

　print("a") はインデントされているので、forの範囲内です。print("b") はループの内側、外側（print("1") やforの行）のどちらともインデントが一致していないので、エラーとなります。

### 注意

　pythonでインデントを表現するとき「半角空白を用いる」と「Tab文字を用いる」の2つの方法があります。両者を混在させるとエラーになります[[3]](#footnote-3)。半角空白を用いることが推奨されています。IDLEではtabキーを押すと、半角空白を4個挿入します。

　ブラウザからプログラムをコピペするとき、元々のプログラムにTab文字が含まれている場合、Tabから半角空白への自動変換が発生することがあります。ブラウザからコピペしたプログラムに、自分が付け足す場合は、要注意です。

　tab文字を空白文字に変換したいときは以下のサイトを使うとよいでしょう。

<https://so-zou.jp/web-app/text/tab-space/>

　変数iを0, 2, 4, ... のように、1以外の等間隔で変化させたい場合は、以下のように書きます。

for i in range(0, 10, 2):

print(i)

【例題】

　1から10までの和を求めるプログラムを作成しなさい。

sum = 0

for i in range(1,11):

sum = sum + i

print("sum = " + str(sum))

【課題2‑2】

　6!（1×2×3×.....×6）を計算するプログラムを作りなさい。

## if文

　if文は次のように書きます。% は余りをとる演算です。

a = 4

if a % 2 == 0: # % は余りをとる演算 C と同じ

print("偶数")

else:

print("奇数")

　ifもforと同様に、ブロックを要求します。ゆえに、if文とelse文の後には :（コロン）が必要です。

　ifで等しいかどうかを判定する演算子は == です。Basicは「=」、C言語は「==」なので、C言語と同じです。else: 以下が不要な場合は、省略可能です。しかし、以下のように条件が成立するときの処理を省略すると、エラーが発生します。

a = 4

if a % 2 == 0:

else:

print("奇数")

　BasicやC言語では上記と等価な表現が許されましたが、pythonではダメです。ifの次に来る文はインデントが必要であるというエラーが出ます。以下のようにpassという文が必要です。

if a % 2 == 0:

pass

else:

print("奇数")

　Ifで数値を比較する場合の書き方は、以下のようにC言語と同じです。

a == b 等しい

a != b 等しくない

a < b

a > b

a <= b # a =< b はエラー

a >= b # a => b はエラー

　Else Ifについて復習しておきましょう。変数aの値によって、以下のように表示する場合を考えます。

* 10以上の場合「10 <= a」と表示する
* 5以上10未満の場合「5 <= a < 10」と表示する
* 0以上5未満の場合「0 <= a < 5」と表示する
* 0未満の場合「a < 0」と表示する

　ifは「成立する」か「成立しない」かの二者択一のどちらかを判断する文です。素直に考えると、以下のアルゴリズムが考えられます。



　これをプログラムにすると以下のようになります。

a = 8 # a の値は色々と変えてみて、プログラムの動作をチェックする

if a >= 10:

print("10 <= a")

else:

if a >= 5:

print("5 <= a < 10")

else:

if a >= 0:

print("0 <= a < 5")

else:

print("a < 0")

　このプログラムは、elseの中にifが入っており、少し見づらいです。elseとifを一体化したelifという命令があり、以下のように書くことができます。

if a >= 10:

print("10 <= a")

elif a >= 5:

print("5 <= a < 10")

elif a >= 0:

print("0 <= a < 5")

else:

print("a < 0")

　フローチャートは以下のように書いた方が、分かりやすいかも知れません。



【例題】

　1～50までの数値を連続して書き出すプログラムを作成しなさい。ただし、数値を1個出力するたびに、改行します。そして「3の倍数」「3を含む（ex. 13, 32）」の場合は、「13 V」のように、数値の後に半角空白を置き、その後に「V」を書きなさい。

　素直に考えると以下のプログラムとなります。

for i in range(1, 51):

if i % 3 == 0: # 3で割り切れるとき

print(str(i) + " V")

elif i % 10 == 3: # 1の位が3のとき

print(str(i) + " V")

elif int(i / 10) == 3: # 10の位が3のとき

print(str(i) + " V")

else:

print(i)

　elifを何回も使わずに、以下のようにorを使う方法もあります。

for i in range(1, 51):

if i % 3 == 0 or i % 10 == 3 or int(i / 10) == 3:

print(str(i) + " V")

else:

print(i)

　orは複数の条件を列挙し、1つ以上が成立したときに「条件成立」とみなす場合に使います。複数の条件を列挙し、全て真のときに「条件成立」とみなす場合はandを使います。aが4でも7でも割り切れる場合に「a は28の倍数です」と表示するには以下のように書きます。

if a % 4 == 0 and a % 7 == 0:

print("a は28の倍数です")

（発展）

　この例では「aが3で割りきれるか否か」を「3で割った余りが0なら割り切れる」という判定の方法をとりました。余りをとる演算を持たない処理系の場合（いまどきそんな処理系はないと思いますが、昔はありました）は以下のように書きます。

if int(a/3) == a/3:

　ここでint(a/3) という表現が出てきました。関数intは2つの使い方があります。

　引数[[4]](#footnote-4)として文字列を与えると、その文字列を数値にします。"11.2" のような小数を与えるとエラーが発生します。

　引数として数値を与えると、小数点以下を切り捨てて整数にします。たとえば int(11.2) は11を返します[[5]](#footnote-5)。上の例でa=4のとき、int(a/3) は1.3333…… を切り捨てて1になり、右辺は1.3333…… なので == は成立しません。a=3のときは int(a/3) と a/3 はどちらも1になるので、== が成立します。

（発展）

　「3を含む」という条件を「文字 "3" を含む」と考えて、以下のようなプログラムも考えられます。ここでの "3" in str(i) は文字列iの中に "3" が含まれているならifが成立します。

for i in range(1, 51):

if i % 3 == 0 or "3" in str(i):

print(str(i) + " V")

else:

print(i)

## キーボードからの入力

　キーボードから入力を取得するとき以下のように書きます。

a = input("何か入力して下さい　")

b = a

if b == 1:

print("b = 1")

else:

print("b != 1")

　実行すると、input文のところで「何か入力してください　」と表示し、キーボードからの入力を待ちます。何か入力してenterキーを押すと、入力した文字列をaに代入して、次のb = aの文を実行します。b = a は無駄なように見えますが、気にせず先を読み進めてください。

　キーボードから「1」「enter」と入力すると、b != 1 と画面に表示されます。1を入れたはずなのに、なぜか？

　それは、数値の 1 と文字列の "1" は異なるとpythonが判定するからです。キーボードから入力したものは、文字列としてaに代入されます。b = a の部分を以下のように書き換えて下さい。

b = float(a)

　float(a) は文字列aを実数に変換します。a = "ab" のように数値に変換できない文字列だった場合は、エラーが発生します。

b = int(a)

とすると、文字列aを整数に変換します。a = "1.5" のように、整数でない文字列が入っていた場合は、エラーが発生します。

　今回のサンプルプログラムの場合は、float(), int() どちらでも構いません。

【課題2‑3】

　キーボードから入力した数値が偶数か奇数かを表示するプログラムを作りなさい。ただし、a÷bの余りをrとするとき、r = a % b と表記します。

【課題2‑4】

　キーボードから入力した数値が80以上のとき「A」、60以上80未満のとき「B」、60未満のとき「C」と表示するプログラムを作成しなさい。

## while文

　whileループは以下のように組みます。

i = 1

print(i)

while i < 5:

print("while top")

i = i + 1

print(i)

print("while end")

print("loop exit")

　whileもブロックを要求するので、右端に :（コロン）が必要です。無限ループは以下のように組みます。trueと書くとエラーになります。Trueと書いてください。

i = 1

print(i)

while True:

print("while top")

i = i + 1

print(i)

if i == 5:

break

print("while end")

print("loop exit")

　ループからの脱出はbreakです。これはCと同じです。ループの先頭に戻るのはcontinueで、これもCと同じです。例を示します。

i = 1

print(i)

while True:

print("while top")

i = i + 1

print(i)

if i == 3:

continue

if i == 5:

break

print("while end")

print("loop exit")

【課題2‑5】

　1, 2, 4, 8, ... という数列を表示するプログラムを作成しなさい。ただし2000未満の範囲で表示しなさい。

## print関数

　print関数についてもう少し詳しく学びましょう。まずは、下の例を見て下さい。

line = "abc"

line2 = "def"

line3 = "xyz"

print(line) # 改行あり

print(line2, end = "") # 改行なし

print(line3) # 改行あり

　print関数は1回実行すると改行します。改行を防ぐには、コンマで区切った後、

end = ""

と記述します。

### 整形して出力する

　まずは、次のプログラムを実行してみてください。

i = 10

j = -3

f = 12.345

f2 = -12.345

moji = "abcd"

print("i={} j={} f={} f2={} moji={}".format(i,j,f,f2,moji))

　画面に表示する文字列を□□□□□□とするとき、

"□□□□□□".format( , ,... )

という形式で書かれています。□□□□□□□の中に {} が5個あります。{} の中にはformat(i,j,f,f2,moji) で指定した変数の値が順番に入ります。{} の個数とformat() の中の変数の個数は同一にする必要があります（format() の中の変数の個数が多い場合は、{} の個数分を先頭から使用します）。

　{} の中に詳しい指定を書くことができます。整数の場合、桁数を指定したり、空白の桁を0で埋めたりすることができます。以下に例を示します。

print("i = <{:3}> i = <{:03}>".format(i,i))

print("j = <{:3}> i = <{:03}>".format(j,j))

print("i = <{:1}> i = <{:01}>".format(i,i))

print("j = <{:1}> j = <{:01}>".format(j,j))

　{:3} と書くと、3桁で表示します。{:03} と書くと、3桁で表示し、かつ空白の桁に0を書きます。

　{:1} のように、表示する数値が2桁であるにもかかわらず、1桁で表示するよう指定したときは、最低限の桁数で表示します。

　実数の場合は、以下のような指定が可能です。

print("f = <{:8.3f}> f = <{:.3f}>".format(f,f))

print("f = <{:10.3e}> f = <{:.3e}>".format(f,f))

print("f2 = <{:8.3f}> f2 = <{:.3f}>".format(f2,f2))

print("f2 = <{:12.3e}> f2 = <{:.3e}>".format(f2,f2))

　{:8.3f} と書くと、全体を8桁で表示し、小数点以下に3桁確保します。{.3f} と書くと小数点以下に3桁確保します。

　{:8.3e} と書くと、指数形式で表示し、仮数部の小数点以下の桁数のために3桁を確保します。小数点以下4桁目を四捨五入して、小数点以下3桁にします。

　それ以外にも、非常に多くの表示形式の与え方がありますが、ここで書いてあることが使えれば、必要十分であると思われます。ここで学習した方法では対応できないケースに遭遇したときは「python 文字列 フォーマット」などのキーワードでネット検索して下さい。

【課題2‑6】

　i = 30 j = -30 f = 12.3456 のとき、以下のように表示するプログラムを作成しなさい。

i = < 30> j = < -30> f = < 12.35> f = < 1.23e+01

## コメント文と継続文

　# 以降はコメントとみなされます。複数の行にまたがるときは、\ を使います。以下に例を示します。

# 以下は1から10までの和を求める

# プログラムです

sum = 0

for i in range(1, 11):

sum = sum + i # 途中に # を置くこともできます。

print("step ", i,\

"sum = {:d}".format(sum))

print("calc finished.")

1. 文字列を表すのにC言語やBasicでは " で囲みます。FortranやBashでは ' で囲みます。pythonやJavaScriptは両方使えます。perlは両方使えますが、意味が異なります。' で囲んだ文字列はそのまま扱われるのに対して、" で囲んだ文字列は、その中に変数が含まれる場合（変数は $a のように名前に $ がつきます）、変数の内容で置換されます。 [↑](#footnote-ref-1)
2. a = int("12.3") のように括弧の中が実数を表す文字列のときエラーが発生しますが、a = int(12.3) のように括弧の中に数値を入れると、エラーは発生せず整数部分のみを取り出します。aは12になります。 [↑](#footnote-ref-2)
3. python 2.xではtabと半角空白の混在が許されます。tabは半角空白8文字として扱われます。エディタの表示オプションでtab幅を8字に設定している場合は、tabと半角空白を混在させても混乱は生じませんが、tab幅を4字に設定している場合に、tabと半角空白を混在させると、見かけ上はインデントが揃っているように見えても、「tabは半角空白8文字」なのでエラーが発生します。 [↑](#footnote-ref-3)
4. 関数の ( ) の中を引数と呼びます。 [↑](#footnote-ref-4)
5. intの引数に負の数を与える場合、プログラミング言語によって挙動が異なります。pythonやC言語ではint(-1.5) は小数点の部分を切り落として-1となります。それに対してBasic (ex. Excel VBA) では「その値を超えない最大の整数（数直線上で左側（その値を含む）にある一番近い整数）を返すので -2 になります。 [↑](#footnote-ref-5)