プログラミング教室のテクノロ

Pythonの道

トレーニングドリル①

もくじ

・Pythonの基礎を学ぼう
・変数を使ってみよう

文字列を出力してみよう

「print」を用いると文字を出力（表示）することができる。
printの後ろの()の中に書いた文字が画面に出力される。

【コード】

```
print(“ Hello Python ” )
```

【出力結果】

```
Hello Python
```

文字列とは？

「Hello Python」という文字は、プログラミングの世界では「文字列」と呼ばれる。
文字列はシングルクォーテーション「'」またはダブルクォーテーション「"」で囲む必要がある。どちらで囲んでも出力結果は同じとなる。どちらかで囲んでいない場合、コードは動かなくなる。

【コード】

```
print(“ Hello Python ” )
print(‘ Hello Python ‘ )
```

【出力結果】

```
Hello Python
Hello Python
```

コメント

コード内にはコメントを書くことができる。行頭に「#」を書くことで、行末までコメントとみなされる。
コメントはコードが実行されるときにすべて無視されるので、コードに関するメモなどを残しておくことができる。

【コード】

```
# この行はコメントです
```
※コメントは実行されない

```
print(“ Hello Python ” )
# print(“ こんにちは、Python” )
```
※コメントは実行されない

【出力結果】

```
Hello Python
```

練習

問題を読んでコードを書いてみよう。

【問題】

「Hello World」と出力してください

【答え】

```
print(‘ Hello World’ )
```

数値とは？

プログラミングでは、「数値」も扱うことができる。数値は文字列と違って、クォーテーションで囲む必要がない。
数値は数学と同じ記号「+」「-」を用いて、足し算と引き算が可能。数値や記号はすべて半角で記述する。
また記号の前後の半角スペースはなくても構わないが、入れた方がコードが見やすくなるよ。

【コード】

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

【出力結果】

```
3
10
4
```

文字列と数値の違い

「3+5」と入力すると計算結果である「8」を出力する。
一方、「”３＋５”」のようにクォーテーションで囲むと、文字列として解釈されて、そのまま「3+5」が出力される。
このようにプログラミングの世界では、文字列と数値は全く異なるものとして扱われる。

【コード】

```
print(3+5)
print(“3+5”)
```

【出力結果】

8 ※数字
3+5 ※文字列

練習

問題を読んでコードを書いてみよう。

【問題】

数値の7を出力してください
9に3足した値を出力してください
「9 + 3」を文字列として出力してください

【答え】

```
# 数値の7を出力してください
print(7)
# 9に3足した値を出力してください
print(9 + 3)
# 「9 + 3」を文字列として出力してください
print(‘ 9 + 3’ )
```

その他の計算

プログラミングでは、足し算・引き算以外の計算をすることもできる。
掛け算は「*」、割り算は「/」で表す。また、「%」で割り算の余りを計算することができる。この3つの記号は数学で用いる記号と少し違うので、しっかり覚えよう。

【コード】

```
print(3*7)
print(3/2)
print(7%3)
```

【出力結果】

21
1.5 ※小数で出力される
1 ※7割る3は、2余り1

練習

問題を読んでコードを書いてみよう。

【問題】

9を2で割った値を出力してください
7に5を掛けた値を出力してください
5を2で割った時の余りを出力してください

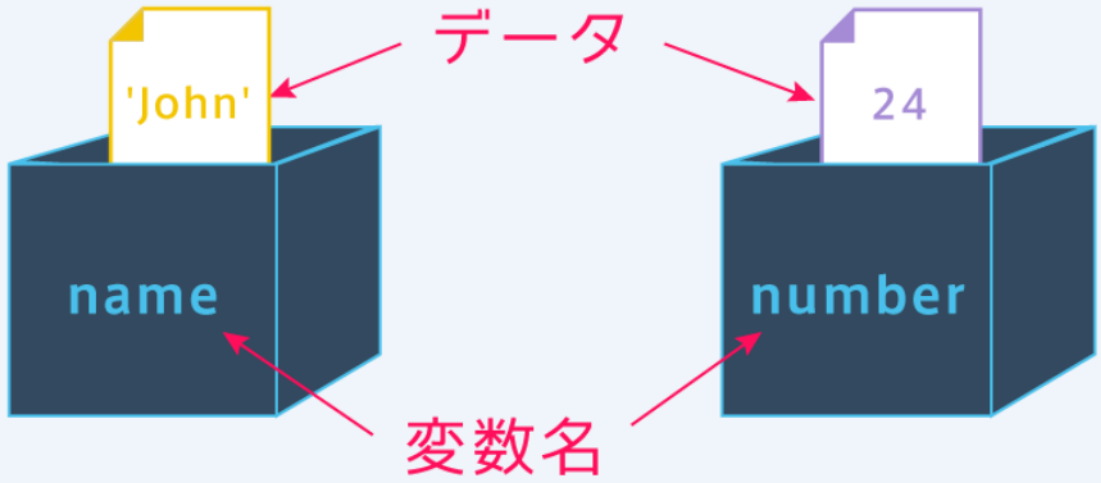
【答え】

```
# 9を2で割った値を出力してください
print(9 / 2)
# 7に5を掛けた値を出力してください
print(7 * 5)
# 5を2で割った時の余りを出力してください
print(5 % 2)
```

変数とは？

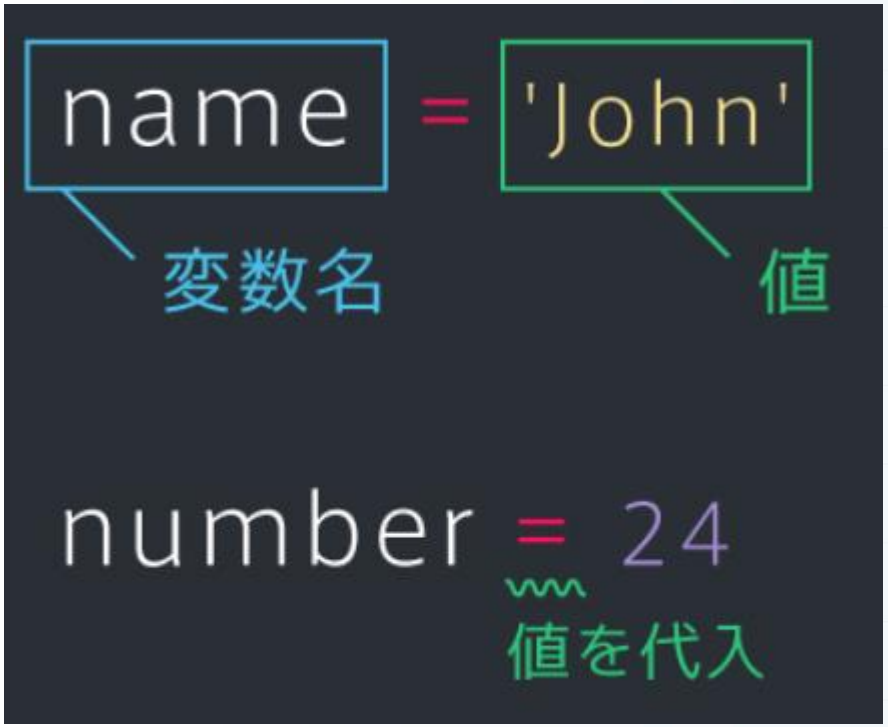
ここからは「変数」について学習する。
変数とは、データ（値）を入れておく箱のようなもの。
この箱（変数）に名前（変数名）をつけることで、その名前を用いて変数に値を入れることや、変数から値を取り出すことができる。

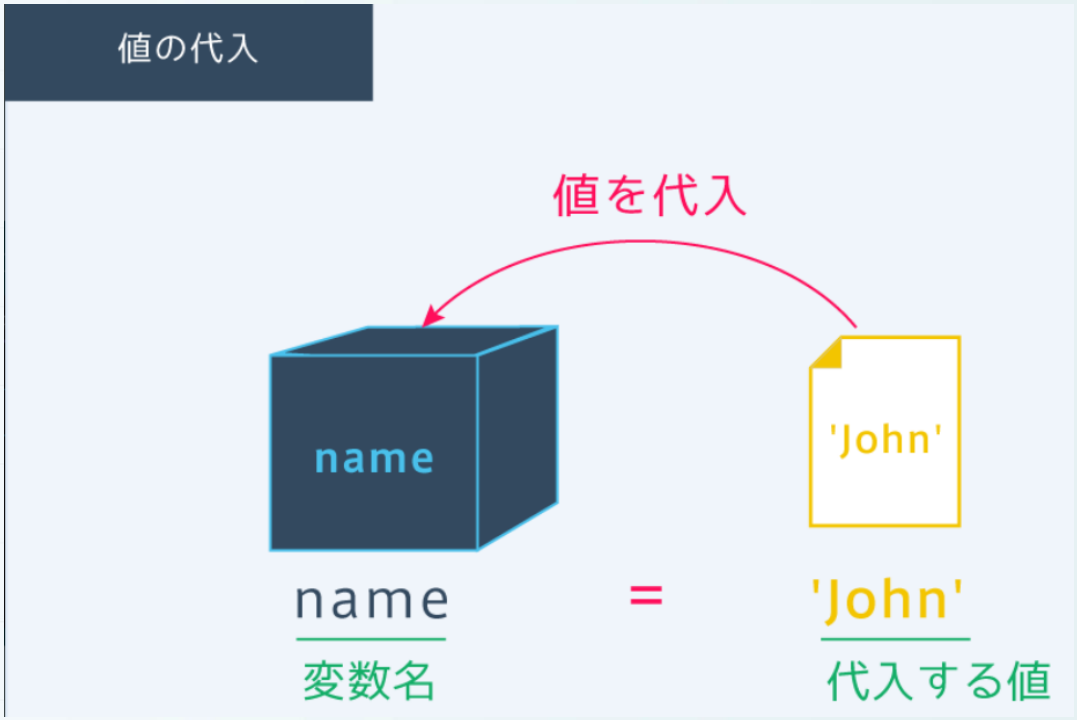
変数のイメージ



変数の定義

変数を活用するために、まず変数を用意（定義）するためのルールを見ていこう。
変数は下図のように「変数名 ＝ 値」で定義する。変数名はクォーテーションで囲む必要はない。また、プログラミングの「=」は「等しい」という意味ではなく、「右辺を左辺に代入する」という意味になるので覚えておこう。





変数の値を取り出してみよう

次は変数の値の取り出し方について学習しよう。
図のようにprint(name)とすると、変数nameの値を出力することができる。しかし、print(‘ name’)のようにクォーテーションで囲ってしまうと、nameが変数ではなく文字列として認識され、「name」とそのまま出力されてしまうので注意しよう。

【コード】

```
name = ‘ John’
print(name) ※変数nameの値
print(‘ name’ ) ※「name」という文字列
```

【出力結果】

```
John
name
```

練習

問題を読んでコードを書いてみよう。

【問題】

変数nameに文字列「にんじゃわんこ」を代入してください
変数nameの値を出力してください
変数numberに数値の7を代入してください
変数numberの値を出力してください

【答え】

```
# 変数nameに文字列「にんじゃわんこ」を代入してください
name = ‘ にんじゃわんこ’
# 変数nameの値を出力してください
print(name)
# 変数numberに数値の7を代入してください
number = 7
# 変数numberの値を出力してください
print(number)
```

変数名の付け方

変数名は自由につけることができるが、守らなければいけないルールがいくつかある。例えば変数名の頭文字を数字にすることはできない。また、「user_name」のように2語以上の変数名を使うときは、単語と単語の間を _（アンダーバー）で区切ろう。

◎良い例

date … ◎ 英単語を用いる

user_name … ◎ 2語以上の場合はアンダーバーで区切る

× 悪い例

1name … × 数字開始

namae … △ ローマ字

名前 … △ 日本語

※変数に慣れるまでは変数名に日本語を使用しても問題ない。
ソースコードをすべてアルファベットで記載すると変数の動きが見えづらくなる。

変数を使う意義（1）

変数の使い方、変数名の付け方を説明したが、なぜプログラミングでは変数が使われるのだろうか？
変数を使うメリットの1つは、データに名前をつけることで、扱っているデータの中身が何を表しているのかが明確になることだ。その結果、コードがより読みやすくなる。

■変数を使用した場合

```
apple_count = 3
変数名が何を表しているか分かりやすい

apple_price = 100

total_price = apple_count * apple_price

print(total_price)  # 結果：300
```

■変数を使用しない場合

```
total_price = 3 * 100
              △ 値が何を表しているか分かりにくい

print(total_price)  # 結果：300
```

変数を使う意義（2）

変数を使うことには、他にも以下のようなメリットがある。
・同じデータを繰り返し利用することができる
・変数の値に変更が必要になった場合、変更する箇所が1箇所で済む

■変数を使用した場合

```
# 正方形の面積を計算
length = 5
◎同じデータを繰り返し使える！
area = length * length

print(area) # 結果：25
```

■変数を使用しない場合

```
# 正方形の面積を計算
area = 5 * 5
△変更が複数箇所必要になる

print(area) # 結果：25
```

練習

問題を読んでコードを書いてみよう。

【問題】

```
apple_price = 200
apple_count = 8
# apple_priceとapple_countを掛けた結果を、変数total_priceに代入してください

# total_priceの値を出力してください
```

練習

【答え】

```
apple_price = 200
apple_count = 8

# apple_priceとapple_countを掛けた結果を、変数total_priceに代入してください
total_price = apple_price * apple_count

# total_priceの値を出力してください
print(total_price)
```

変数の値を更新する(2)

すでに定義された変数に数値を足す場合は、下図のように書く。変数自身に数値を足したものを再び同じ変数に代入することで、値を上書きすることができる。この書き方には違和感を感じるかもしれないが、「=」は「代入」の意味であり、「等しい」の意味ではない。引き算などのその他の計算でも同様。

【コード】

```
#  変数xを定義
x = 5
print(x)
x = x +3  ※「5+3」を変数xに代入し直す
print(x)
```

【出力結果】

```
5
8  ※xの値が8に上書きされている
```

変数の値を更新する(3)

数値の入った変数の値を更新する場合は、図のように省略して書くことができる。これは引き算などその他の計算でも同様。この書き方はよく使うのでぜひ覚えておこう。

基本形		省略形
x = x + 10	⇒	x+= 10
x = x - 10	⇒	x-= 10
x = x * 10	⇒	x*= 10
x = x / 10	⇒	x/= 10
x = x % 10	⇒	x%= 10

練習

問題を読んでコードを書いてみよう。

【問題】

```
money = 2000
print(money)

# 変数moneyに5000を足して、変数moneyを上書きしてください

# 変数moneyの値を出力してください
```

練習

【答え】

```
money = 2000
print(money)

# 変数moneyに5000を足して、変数moneyを上書きしてください
money += 5000　もしくは　money = money + 5000

# 変数moneyの値を出力してください
print(money)
```

文字列の連結

数値の計算で用いた「+」記号は、計算だけでなく文字列の連結も行うことができる。
文字列の連結は以下のようにする。
他にも、変数と文字列の連結、変数同士の連結をすることができる。

【コード】

```
print(‘ hello’ + ‘ Python’ ) ※文字列同士を連結
name = ‘ John’
print(‘ My name is’ + name)  ※文字列同士を連結
```

【出力結果】

```
Hello Python
My name is John
```

練習

問題を読んでコードを書いてみよう。

【問題】

my_nameという変数に「にんじゃわんこ」という文字列を代入してください

my_nameを用いて、「私はにんじゃわんこです」となるように変数と文字列を連結して出力してください

練習

【答え】
my_nameという変数に「にんじゃわんこ」という文字列を代入してください
```
my_name = 'にんじゃわんこ'
```

my_nameを用いて、「私はにんじゃわんこです」となるように変数と文字列を連結して出力してください
```
print('私は' + my_name + 'です')
```

データ型

ここではデータ型というものについて学習していこう。
これまで「文字列」や「数値」という言葉で値の種類を説明してきたが、これらは「データ型」と呼ばれ、さまざまなデータの種類が存在する。
まずは「文字列型」と、数値のデータ型である「数値型」を覚えよう。

【データ型】
‘ Hello Python’　・・・文字列型
3　　・・・数値型

データ型の違い

データ型が異なるとコードは異なる動作をする。
例えば、図のprint(5 + 7)は数値の計算、
print(‘ 5’ + ‘ 7’)は文字列の連結となる。

【データ型】

```
print(5 + 7)  ※数値の計算
 #結果：12

print(‘ 5’ + ‘ 7’ )  ※文字列の連結
 #結果：57
```

型変換 str

データ型の異なる文字列型と数値型を連結すると、以下のようにエラーが起きてしまう。しかし、数値型を文字列型に変換すると、文字列同士の連結として扱われるようになるため、連結ができるようになる。このようにデータ型を変えることを「型変換」という。数値型を文字列型に変換するには「str」を用いる。

script.py

```
price = 100
print('りんごの価格は' + price + '円です')
```

文字列型　数値型　文字列型

コンソール

```
TypeError: Can't convert 'int' object to str implicitly
```

✕ エラー！！：異なる型同士の連結はできない

script.py

```
price = 100
print('りんごの価格は' + str(price) + '円です')
```

文字列型に変換

コンソール

```
りんごの価格は100円です
```

◎ 文字列型同士なので連結できる

型変換 int

先ほどの例とは反対に、文字列型を数値型に変換したい場合には「int」を用いる。intは下図のように使う。

```
count =  '3'
price = 100
total_price = price * int(count)
print(total_price)
```

数値型に変換

【出力結果】

３００

練習

問題を読んでコードを書いてみよう。

【問題】

```
age = 24
# ageを用いて「私は24歳です」と出力してください

count = '5'
# countに1を足した値を出力してください
```

練習

【答え】

```
age = 24
# ageを用いて「私は24歳です」と出力してください
print('私は' + str(age) + '歳です')

count = '5'
# countに1を足した値を出力してください
print(int(count) + 1)
```

復習＆チャレンジ

ここまで習ったことをScratchでもできるかチャレンジしてみよう。
その過程でScratchでできること、Pythonでないとできないことを整理してみよう。
例えば、データの型という概念はscratchにはあっただろうか？

メモ

プログラミング教室のテクノロ

なまえ：