

Python プログラミング基礎 教材 2

九州大学 数理・データサイエンス教育研究センター

全体の構成

● 教材 1

- インTRODクシヨン～プログラムってなんだ？
- プログラミング言語 Python ～無料で遊べるのに凄い
- 【準備】Gogole Colaboratoryへ行こう！
- 文字を表示してみよう ～ 記念すべき最初のプログラム

● 教材 2

- 計算させてみよう ～ たす, ひく, かける, わる
- 変数 ～ 電卓を越える！
- If文 ～ ますますプログラムっぽく
- 便利なコメントアウト ～ ちょっと休憩
- 数当てゲームを作ってみよう ～ ゲームの基本中の基本

● 教材 3

- For文 ～ 人間には面倒な繰り返しを簡単に
- 【寄り道】コンピュータの計算精度 ～ 意外と正確じゃない
- 再びゲームを作ってみよう ～ 工夫次第でどんどん楽しく

● 教材 4

- List ～複数の変数を固めたもの
- 関数とライブラリ ～ ひみつ道具で遊ぼう

● 付録

- if文のちょっと進んだ使い方
- AIを用いたプログラミング
- Google Driveとの連携

計算させてみよう

たす, ひく, かける, わる

電卓のでもできる四則演算は プログラムでも当然できます！

- 足し算： $+$
 - $1+2$
- 引き算： $-$
 - $5-2$
- 掛け算： $*$ (アスタリスク)
 - $5*6$
- 割り算： $/$ (スラッシュ)
 - $5/2$
- 括弧： $()$ 内を優先的に計算
 - $(1-4)*5$

練習：計算させてみよう

▶ $(1+2)*3/4$

↔ 2.25

▶ $(1.01-3.14)/5.963$

↔ -0.3572027502934764

▶ $(1/3+2/3)*(1/2+1/2)$

↔ 1.0

▶ $10000000000000*1000000000000000$

↔ 100000000000000000000000000000000

小数もOK

分数もOK

大きな数もOK

練習：もう少し凝った計算もやってみよう

- 小学生の問題
 - 八百屋で100円のリンゴ5個と200円のナシ1個を買いました
 - 合計にいくらになるかプログラムで計算させてみよう！

- BMI をプログラムで計算してみよう
 - ヒント：BMI は $\text{体重} \div (\text{身長} \times \text{身長})$
 - 体重は kg, 身長は m

変数

電卓を超える！

電卓のこのボタン，知っていますか？



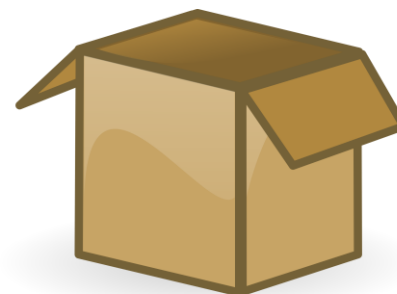
「M+」と書かれたボタン

計算結果を一時記憶できる
「メモリー」ボタン！

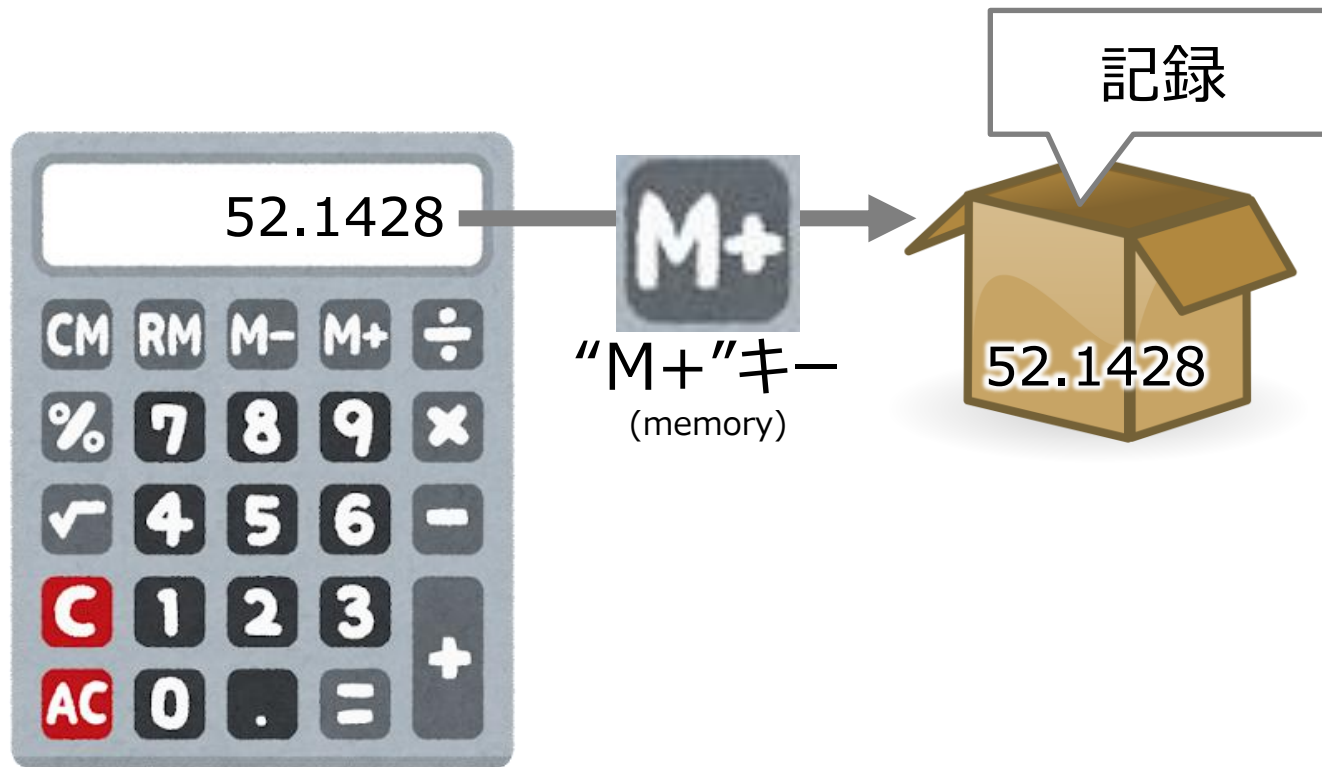
例えば，"365/7="の後にM+を押すと，
52.1428...が記憶
(あとでRMボタンで呼び出せる)

箱をイメージするとわかりやすい

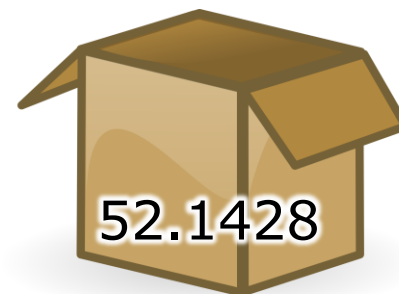
“ $365/7=$ ”の結果



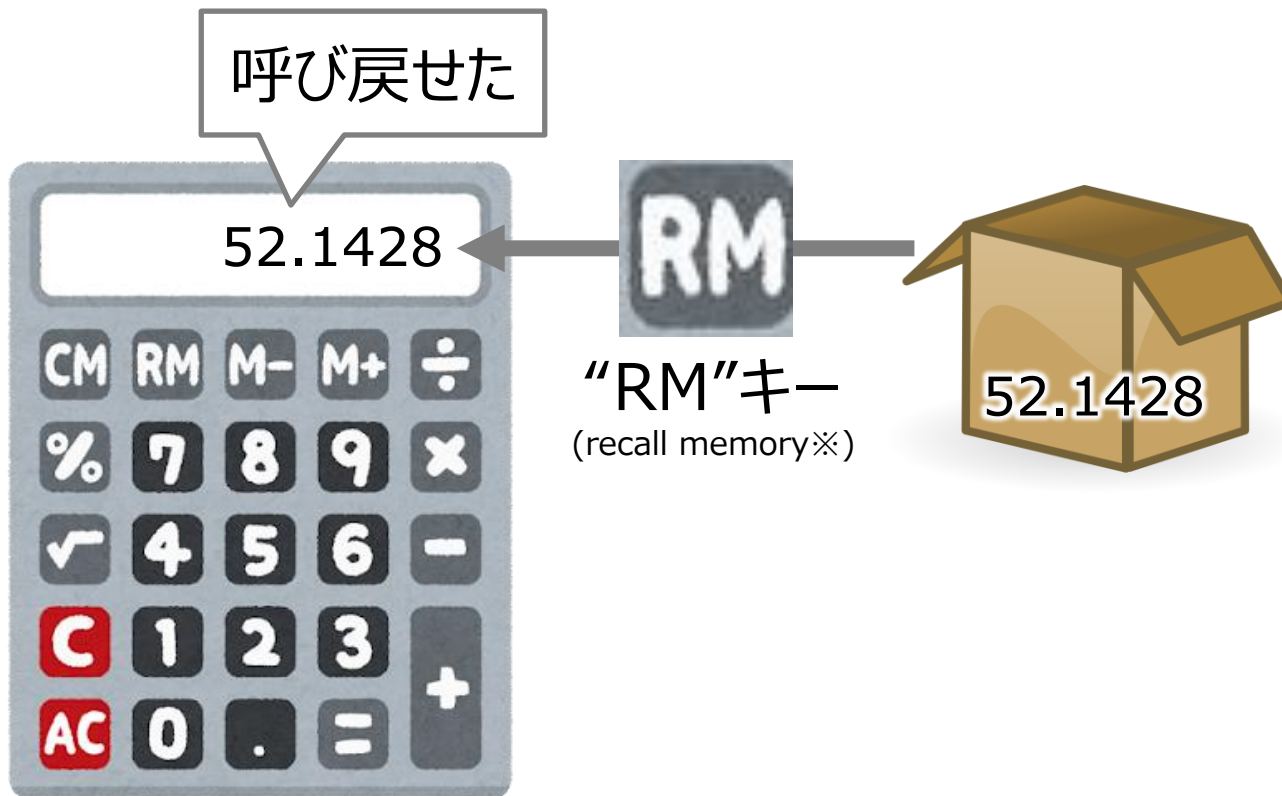
箱をイメージするとわかりやすい



箱をイメージするとわかりやすい



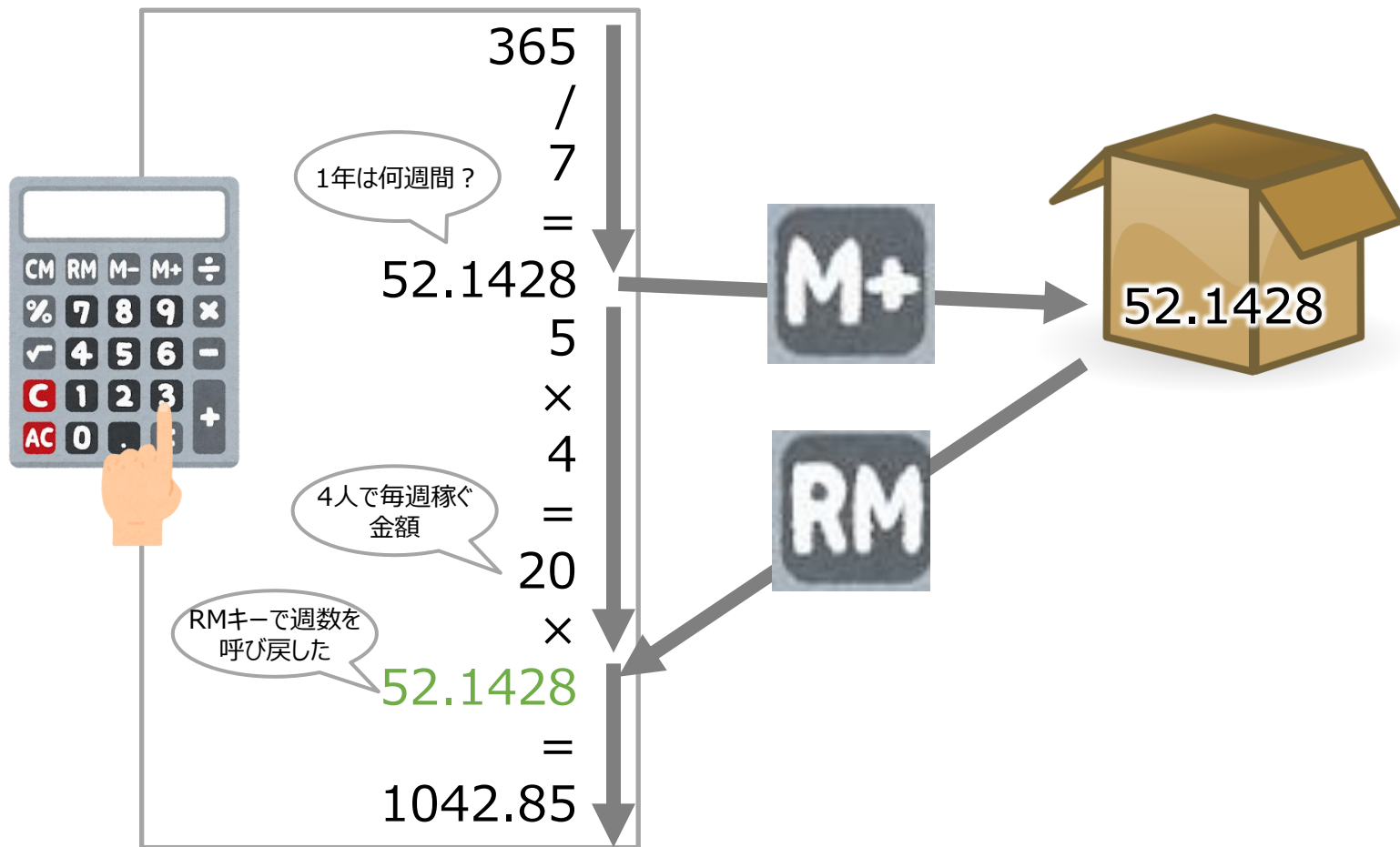
箱をイメージするとわかりやすい



※電卓のメーカーによってはMRキーと呼ばれます

電卓操作の「手順」と箱

「4人がそれぞれ毎週5万円稼いだら
1年でおおよそ幾ら？」の計算**手順**



実は先ほどのゲームの例でも箱が使われている！

1. 森をふらふら歩く

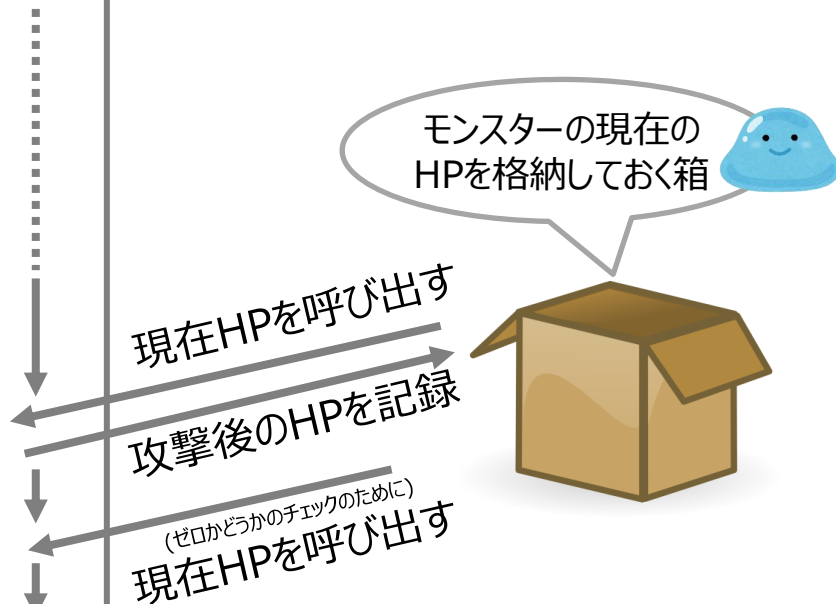
2. モンスターに出会う

●逃げる？

- Yesなら「1. 森をふらふら歩く」に戻る
- Noなら次の「3. 戦う」に進む

3. 戦う

- モンスターを攻撃する
- モンスターのHP(ヒットポイント)が減る
- モンスターのHPがゼロになった？
 - Yesならモンスターを倒したので、「1. 森をふらふら歩く」に戻る
 - Noなら、「3. 戦う」に戻って攻撃を繰り返す

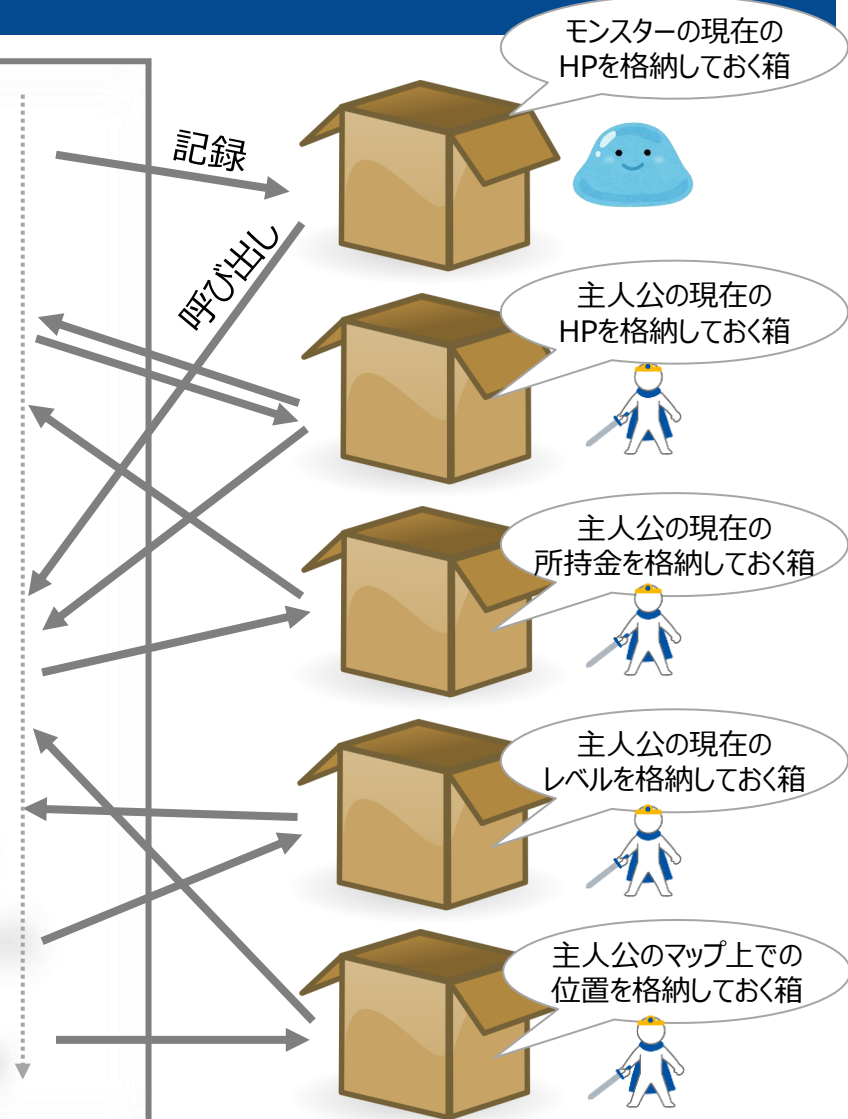


注1：モンスターに出会った瞬間に、そのモンスターの最初のHPが箱に入ります
 注2：箱の中身は攻撃を繰り返す度に100→60→25→5→0と減っていきます



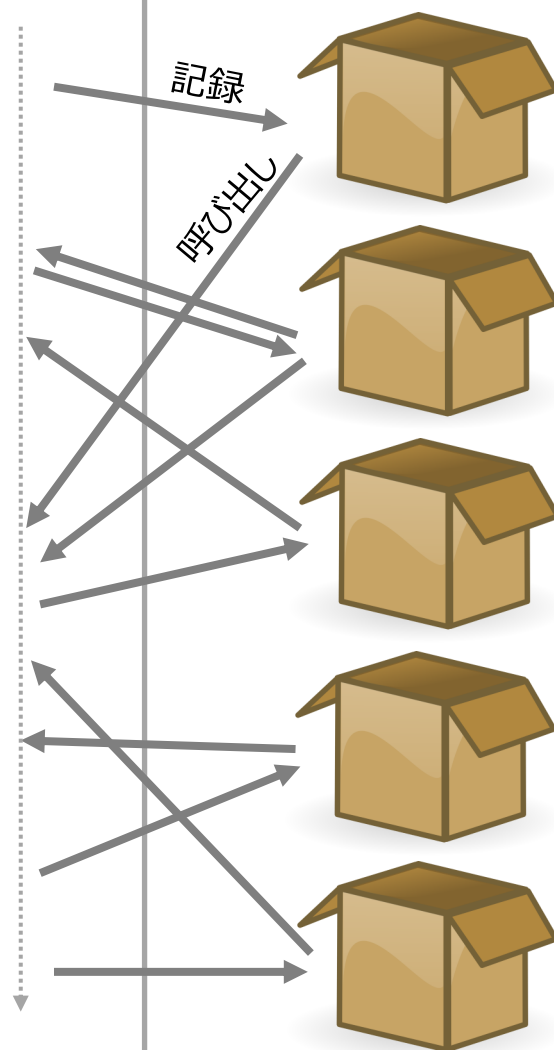
あれ？ 本当にRPGを作ろうと思ったら、
箱って1つじゃ足りないんじゃない？

実際の RPGの手順



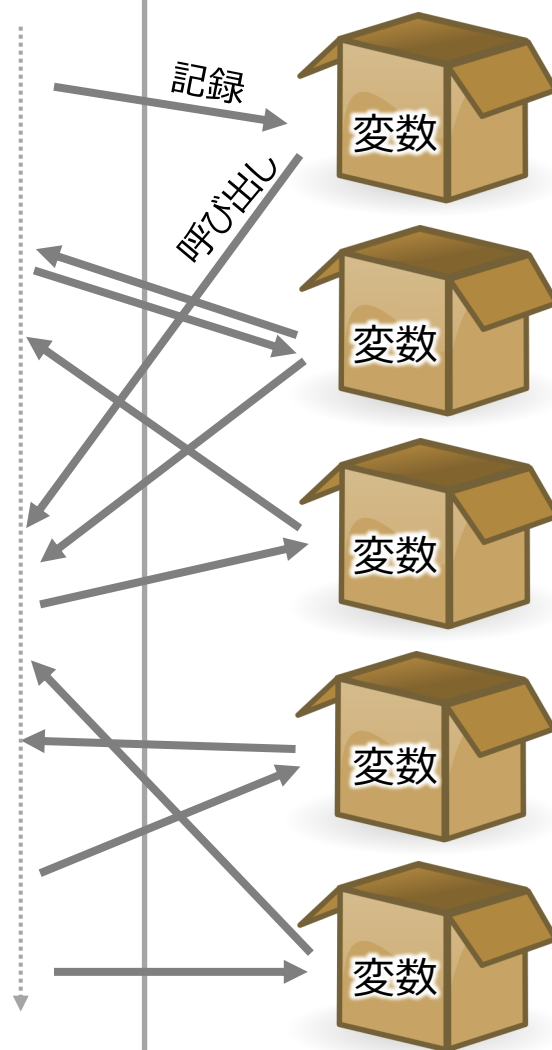
電卓は「1つの箱だけ」。
一方、プログラムは「たくさんの箱」を使える！

プログラムの
手順



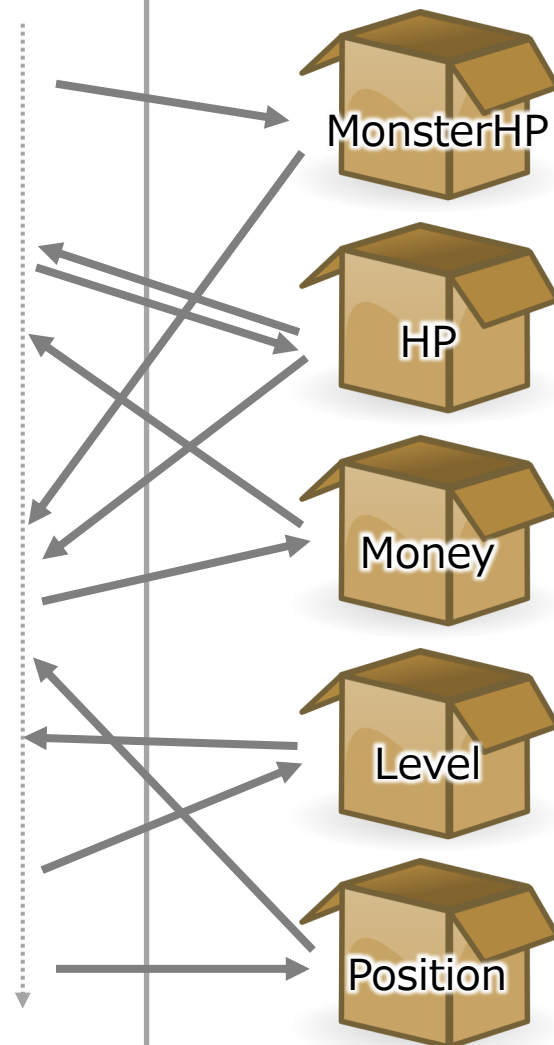
この「箱」を「変数」と呼ぶ

プログラムの
手順



何を入れる箱か区別できるように
変数には**名前を付けて**区別する

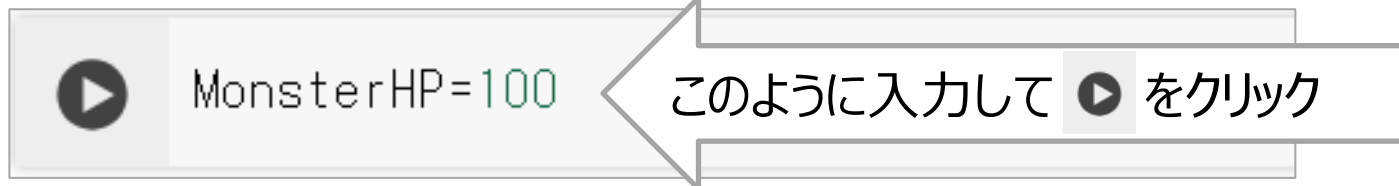
プログラムの
手順





変数に値を記録(代入)する

- 例：MonsterHP=100
 - これでMonsterHPという変数に100が記録される
 - 「記録する」ことを「**代入する**」とも言います
- 実際にプログラムで代入みる



あれ？クリックしても何も出ない…本当に代入できたのかな？





変数の中身を**確認**する

- `print(変数名)` で確認できます



```
[1] MonsterHP=100
```

```
▶ print(MonsterHP)
```

```
↔ 100
```

このように入力して

100と表示された

ちゃんと入ってた!



- **ここでは引用符("...")は不要!**
 - `print("MonsterHP")`と書くと、中身ではなく、MonsterHPという文字列が表示される

変数の中身を**操作**する

- MonsterHPの中身を40減らしてみよう

```
[1] MonsterHP=100
```

```
[2] print(MonsterHP)
```

```
⇒ 100
```

```
[3] MonsterHP = MonsterHP - 40
```

```
▶ print(MonsterHP)
```

```
⇒ 60
```

先ほどまでの結果
(100が入っている)

このように入力して ▶

このように入力して ▶

60と表示された

確かに60には
なったけど、何か
ヘンだな...



補足：「何かヘンだな？」と思った人は鋭い

- 先ほどの式，よく見るとなんか変だ…

```
[3] MonsterHP = MonsterHP - 40
```



- 数学で習ってきた「等式」なら，右辺と左辺は絶対「イコール（＝）」にはならないはず…
 - ある数から40引いたものが，元の数に等しいなんてことはあり得ない

- 実はこのプログラムにおいて，「＝」はイコールではなく，**代入を意味する**

```
[3] MonsterHP = MonsterHP - 40
```



①元のMonsterHPの中身呼び出し

②それから40をマイナスする

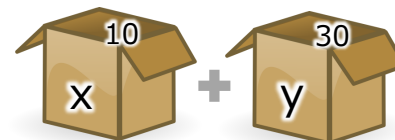
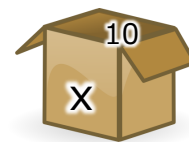
③その結果を左辺(MonsterHP)に代入する

- 右の①②③の手順と解釈するのが正解
 - まず右辺を処理し，その結果を左辺に代入

変数を用いて四則演算

- 例えば、変数を足し算すると、中身が足し算される

[5] x=10	このように入力して
[6] y=30	このように入力して
 x+y	このように入力して
 40	40と表示された！

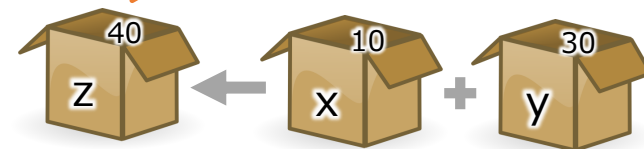


変数を用いて四則演算

- 計算の結果を，別の変数に代入することも可能

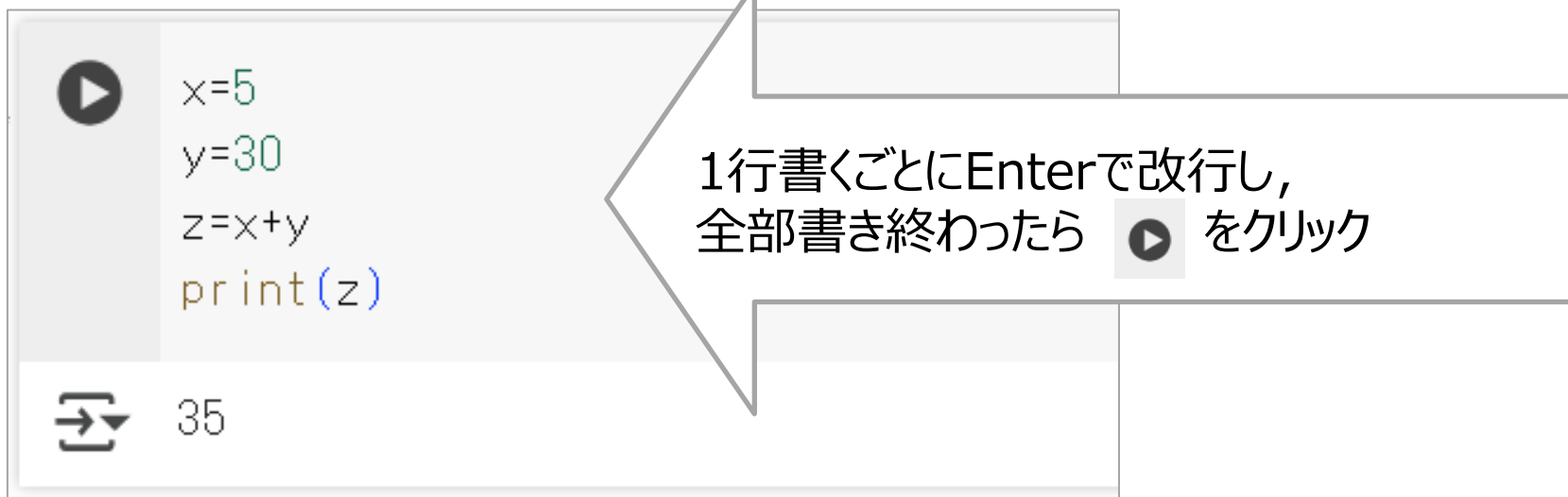
[5] x=10	このように入力して	▶
[6] y=30	このように入力して	▶
[9] z=x+y	このように入力して	▶
▶ print(z)	このように入力して	▶
↔ 40	40と表示された!	

数字は入っていない式だが，ちゃんとxとyの中身が足されてzに代入される



補足：1行書くごとに「▶」をクリックするのは面倒だ

- その通り。多くの場合、複数行を書いてから、▶ をクリック
 - 複数行に書いた手順がまとめて実行されます



```
x=5
y=30
z=x+y
print(z)
```

▶

⇌ 35

1行書くごとにEnterで改行し、
全部書き終わったら ▶ をクリック

- なお、▶ をクリックする代わりに、ShiftキーとEnterキーを同時押しでも実行可能
 - 同時に新しいセルも作ってくれます

練習：BMIを計算してみよう

- 以下を入力して実行してみよう！
 - weight, height, そしてBMIという3つの変数を用いています



```
weight=50  
height=1.6  
BMI=weight / (height*height)  
print(BMI)
```



```
19.531249999999996
```

体重 ÷ (身長 × 身長)

割と重要な

補足：変数に名前を付けるには…

- 英文字 (a-z, A-Z), アンダースコア (_), 数字 (0-9)の組み合わせならOK
 - ただし, 数字で始めることはできない. **日本語もダメ**. アンダースコア以外の記号もダメ
 - OKな名前: x, y, K, weight, room01, person_name, START
 - NGな名前: 001, 1st, 身長, good-bad, key+
- あとはセンスだが, **変数の役割に応じた名前をつけるほうがよい**

```
▶ weight=50
height=1.6
BMI=weight / (height*height)
print(BMI)
```

```
⇒ 19.531249999999996
```

わかりやすい



```
▶ a=50
b=1.6
c=a / (b*b)
print(c)
```

```
⇒ 19.531249999999996
```

わかりにくい



変数のメリットは「記憶」だけじゃない！ 「使いまわせる」ことも大事！

電卓じゃできない…
プログラムならではのメリット

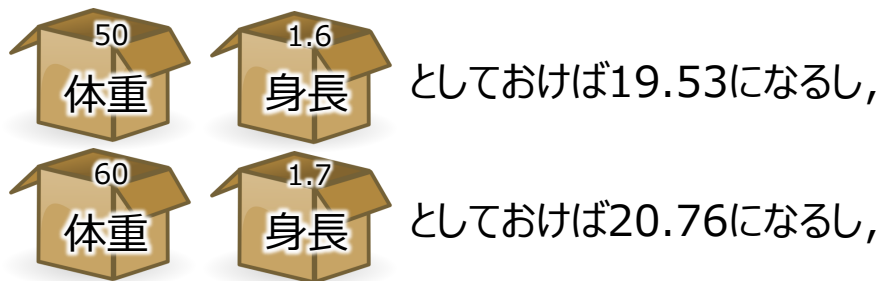


- BMIの式：体重 ÷ (身長×身長)
 - 体重が50kgで身長が1.6mだったらBMIは約19.53
 - 体重が60kgで身長が1.7mだったらBMIは約20.76
 - 1つの式があれば，どんな体重・身長でもBMIを計算できる
 - 体重と身長の値を書き換えればよだけ！

● これと同じ！



という式を準備しておけば，様々な身長・体重に対して使いまわせる！例えば…



次スライドで
実際のプログラムを
見てみよう



練習：実際に使いまわしてみよう

```
▶ weight=50  
height=1.6  
BMI=weight / (height*height)  
print(BMI)
```

⇒ 19.531249999999996

先ほどのプログラム

```
▶ weight=60  
height=1.7  
BMI=weight / (height*height)  
print(BMI)
```

⇒ 20.761245674740486

この数字だけ書き直して再実行

式はそのまま使いまわし

$$\text{体重} \div (\text{身長} \times \text{身長})$$

文字列も「変数」に記録できる！

- プログラムの変数は文字列も記録可能
 - 変「数」とはいえ，数だけじゃないところが自由！



```
▶ MonsterName="スライム"  
print(MonsterName)
```

⇒ スライム

変数 MonsterName に文字列 "スライム" を代入

文字列を足すこともできる

- 2つの文字列変数を “+” で足すと, つなげてくれる

```
▶ MonsterName = "スライム"  
  Action = "倒した"  
  Sentence = MonsterName + Action  
  print(Sentence)
```

```
⇒ スライム倒した
```



参考：文字列を引くこともできる？



- 残念ながらできません！ **エラー**が起きます

マークが赤くなる

```

MonsterName = "スライム"
Action = "倒した"
Sentence = MonsterName - Action
print(Sentence)

```

赤い波線でエラーを起こしている箇所を指示

```

-----
TypeError                                Traceback (most recent call last)
<ipython-input-10-3cdb87b8508b> in <cell line: 0>()
      1 MonsterName = "スライム"
      2 Action = "倒した"
----> 3 Sentence = MonsterName - Action
      4 print(Sentence)

TypeError: unsupported operand type(s) for -: 'str' and 'str'

```

どんなエラーかを説明

次のステップ: [エラーの説明](#)

- どんなエラーがでてでも「爆発」はしません！
 - エラーを怖がらずにいろいろ試してみましょう！それがプログラムの自由さだ！

If文

イフ=もしも

ますますプログラムっぽく



実は先ほどのゲームの例に出てくる 「条件に応じてやることを変える」

1. 森をふらふら歩く

2. モンスターに出会う

●逃げる？

- Yesなら「1. 森をふらふら歩く」に戻る
- Noなら次の「3. 戦う」に進む

3. 戦う

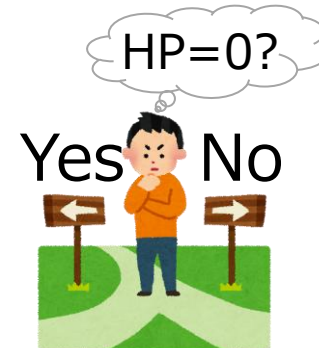
- モンスターを攻撃する
- モンスターのHP(ヒットポイント)が減る
- モンスターのHPがゼロになった？
 - Yesならモンスターを倒したので、「1. 森をふらふら歩く」に戻る
 - Noなら、「3. 戦う」に戻って攻撃を繰り返す



条件

1.に戻る 3.に進む

やること



条件

1.に戻る 3.に進む

やること

皆さんも経験している 「条件に応じてやることを変える」

- 60点以上なら, 合格です!
- 休みの日は, 朝寝する.
- 徒歩で間に合いそうなら徒歩で行く. 間に合いそうにないなら電車で行く.
- A君とB君が出したぐーちょきぱーが同じなら, あいこ.
- 知り合いとすれ違ったら, 挨拶する.
- 所持金が700円以上ならかつ丼, 500~700円なら親子丼, 400~500円なら素うどん, 400円未満ならごはん抜き!



プログラムでは **if** を使う！ まずは例を見てみましょう



```
score = 70
```

scoreという変数に70を入れておく

```
if score >= 60:
```

scoreの中身が60以上かどうか？

```
    print("合格")
```

Yesなら"合格"と表示



合格

70は60以上なので, "合格"と表示された

“if”が出てくるところを詳しく

```
if score >= 60:
```

もし

変数scoreの中身が

以上

条件文の最後には
“:” (コロン)を付ける
(pythonのルール)



条件：scoreの中身が60以上かどうか？

条件を満たさなければどうなるか？



```
score = 50
```

scoreという変数に50を入れておく

```
if score >= 60:
```

scoreの中身が60以上かどうか？

```
    print("合格")
```

Yesなら"合格"と表示

50は60以上ではないので、何も表示されない
(print("合格")はスキップされた)

あれ？ ずれてる？



```
score = 50  
if score >= 60:  
    print("合格")
```



あれ？ printの前に
空白が…

実はこの「ずれ」が超重要！



```
score = 50
if score >= 60:
    print("合格")
```



インデントと呼ばれます
4文字分のスペースが基本

インデントされている部分が、 条件が成り立った時に実行される部分を表す



```
score = 50
if score >= 60:
    print("合格")
```

インデントされているので、
Yesならここが実行される

インデントされている部分が、 条件が成り立った時に実行される部分を表す



```
score = 70
```

← scoreに70を入れておく

```
if score >= 60:
```

```
    print("合格")
```

```
    print("よかったね")
```

どちらもインデントされている
→ Yesならどちらも実行



合格

よかったね

2つのprintがどちらも実行された

うっかりインデントを忘れると…



```
score = 50
```

← 今度は50点 (不合格点)

```
if score >= 60:
```

```
    print("合格")
```

← Yesならこの部分を実行

```
print("よかったね")
```

← インデントされていないので、
条件とは無関係に実行される

インデント忘れ



よかったね

← 条件とは無関係に実行された

練習：次のプログラムを実行してみよう



```
score = 70
if score >= 60:
    print("合格")
    print("よかったね")
    print("おめでとう")
```

自主練習：いろいろ変えて遊んでみよう



```
score = 70
if score >= 60:
    print("合格")
    print("よかったね")
    print("おめでとう")
```

遊び方①
scoreに代入する値を
色々変えてみよう

遊び方②
合格基準点を
変えてみよう

遊び方③
表示するメッセージや
その数を色々変えてみよう

遊び方④
インデントをいろいろ変えてみよう
エラーが出るかもしれないが、それも面白いところ

条件が満たされないときにも何かしたい

74

条件を満たさなければどうなるか？



```
score = 50
```

scoreという変数に50を

```
if score >= 60:
```

scoreの中身が60以上

```
    print("合格")
```

Yesなら"合格"と表示

50は60以上ではないので、何も表示されない
(print("合格")はスキップされた)

60点以下なら
"不合格"と
表示したい



elseを使えばできる！



```
score = 50
```

scoreという変数に50を入れておく

```
if score >= 60:
```

scoreの中身が60以上かどうか？

```
    print("合格")
```

Yesなら"合格"と表示

```
else:
```

Noなら, else以下を実行

```
    print("不合格")
```

"不合格"と表示



不合格

50は60以上じゃないので, "不合格"と表示

ところで、「以上」以外にもいろいろあります

`if score >= 60:`

<code>a == b</code>	a と b が等しい?
<code>a != b</code>	a と b が異なる?
<code>a > b</code>	a が b より大きい?
<code>a >= b</code>	a が b 以上?
<code>a < b</code>	a が b より小さい?
<code>a <= b</code>	a が b 以下?

注意

「`a=b`」は「`a`を`b`に代入」
「`a==b`」は「`a`と`b`は等しい?」

- 【付録】の「if文のちょっと進んだ使い方」も適宜参照してください
- 実はもっといろいろなこともできるんです！

練習：ifを使ってプログラムを書こう

- 身長と体重を与えて，BMIが18.5未満なら「やせ型」と表示

- ヒント：BMIの計算は，以前やりましたね

```
weight=50  
height=1.6  
BMI=weight / (height*height)
```

- 身長と体重を与えて，BMIが18.5未満なら「やせ型」と表示，そうでないならば「やせてない」と表示

- 所持金とかつ丼代を与えて，所持金がかつ丼代以上なら「食べられる！」，以下なら「がまん！」と表示

- ヒント：変数どうしの比較になる

```
money = 500  
price = 700  
if money >= price:
```

便利なコメントアウト

ちょっと休憩

コメントアウト： 一部を実行しないようにする

行頭に#を付ける

```
▶ score = 70
  if score >= 60:
    print("おめでとう")
    print("合格です")
```

⇒ おめでとう
合格です

```
▶ score = 70
  if score >= 60:
    # print("おめでとう")
    print("合格です")
```

⇒ 合格です

print("おめでとう")が
無視された！

実行時には無視されるので メモにも使える！



```
# 点数を入力  
score = 70
```

#がついているので単なるメモ

```
# 合格点以上かを判定
```

#がついているので単なるメモ

```
if score >= 60:  
    print("おめでとう")  
    print("合格です")
```



```
おめでとう  
合格です
```

こういうメモのことを
コメントと呼びます。
丁寧につけておくと、
ミスを少なくできます

#以降がコメントになる



```
score = 70 # 点数を入力
```

#以降がコメント

```
if score >= 60: # 合格点以上かを判定
```

#以降がコメント

```
    print("おめでとう")
```

```
    print("合格です")
```



```
おめでとう  
合格です
```

もう一度注意：printの中とコメント以外では、 日本語入力を使わないようにしましょう

- 変数名に日本語は使えない
- ファイル名にも日本語を使わないほうが無難
 - OK：shincho_taiju.xlsx, height_weight.csv
 - NG：身長体重データ.csv
- 半角カタカナ（アイウ）も利用しない
- 特に全角の空白（スペース）の混入に注意！
 - エラーになる
 - （見えないので）エラーの原因が非常に見つけづらい

a + b
全角スペースが入っている…

数当てゲームを作ってみよう！



様々なゲームの基本中の基本！
(進んだ練習として「ちょっとしたロールプレイングゲーム」も)



数当てゲームを作る材料

- 変数
- print
- if
- else

これまでに学んだツール

+

- random
- input

数当てゲームのために新たに使うツール



まずは **random** : コンピュータにサイコロを振らせる！



- 何のことかわからないとは思いますが，以下のようなプログラムを作ると…



```
import random
```

謎の「おまじない」

```
r = random.randint(1, 6)  
print(r)
```

何かを変数rに代入

変数rの中身を表示



```
3
```

変数rの中身は3…なんで??

練習：解説は後ですとして、
とにかく同じプログラムを作ってみて実行してみよう



```
import random
```

```
r = random.randint(1, 6)  
print(r)
```

同じプログラムなのに  を押す度に結果が変わる



```
import random
```

```
r = random.randint(1, 6)  
print(r)
```

 4



```
import random
```

```
r = random.randint(1, 6)  
print(r)
```

 2



```
import random
```

```
r = random.randint(1, 6)  
print(r)
```

 1

1から6までがランダムに出てくる！（サイコロ完成）



解説：まずはこの部分

```
▶ import random
```

```
r = random.randint(1, 6)  
print(r)
```

`random.randint(1, 6)`

簡単に言えば

1から6までの間の整数(1,2,...,6)を
ランダムに1つ選んで返してくれる！

解説：残りの「おまじない」部分

```
▶ import random  
r = random.randint(1, 6)  
print(r)
```

import random

簡単に言えば
さきほどのrandom.randintを使えるように
するためのおまじない

もっとちゃんと言うと…



Randomという名前の

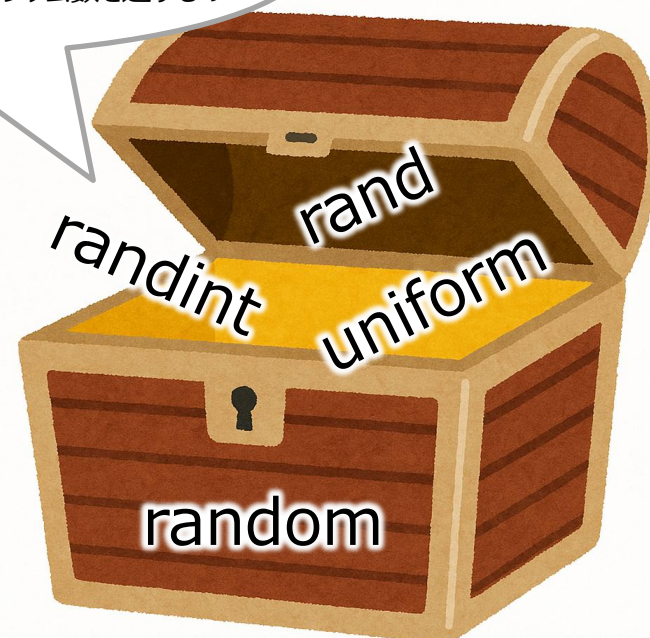
ライブラリ※

(ライブラリ= 道具箱みたいなもの)

色々なツールが入っている！

Ex. 先ほどのrandintは整数のランダム数を返すもの

この中に…



なのでrandomライブラリをimportすると
乱数関係のいろいろなツールが使えるようになる！

ほかにも様々なライブラリがあるので、どんどんimportするとどんどんパワーアップ

※ライブラリについては、また後ほど説明します。

※初心者向けの本資料では、ライブラリ・パッケージ・モジュールの明確な区別をしていません

残る **input**: 実行中にユーザ（人間）が値を入力




```
height = 1.6
```

```
weight = int(input("体重は？"))
```

```
BMI = weight / (height * height)
```

```
print(BMI)
```



- このプログラムを実行  すると…

残る **input**: 実行中にユーザ（人間）が値を入力

①こんな青枠が出てくるので…

```
height = 1.6
weight = int(input("体重は？"))
BMI = weight / (height * height)
print(BMI)
```

… 体重は？

② 例えば50と入れてリターンキーを押すと

```
height = 1.6
weight = int(input("体重は？"))
BMI = weight / (height * height)
print(BMI)
```

… 体重は？

```
height = 1.6
weight = int(input("体重は？"))
BMI = weight / (height * height)
print(BMI)
```

⇒ 体重は？ 50
19.531249999999996

③ (身長1.6m) 体重50kgのBMIが計算されて表示された！

残る **input**: 実行中にユーザ（人間）が値を入力

- というわけで…

```
▶ height = 1.6  
weight = int(input("体重は？"))  
BMI = weight / (height * height)  
print(BMI)
```

`int(input("体重は？"))`

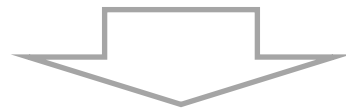
“体重は？”と表示し、それに応じたユーザの入力を
数値として返してくれる！

上の例では、その数値を変数weightに代入

補足：もっと詳しくいうと…
(よくわからなければ飛ばしてOK！)

```
input("体重は？")
```

この段階で出てくる"50"は単に2つの数字"5"と"0"が並んだもの。
数値の50としての意味はない



```
int(input("体重は？"))
```

なので、この部分は `int("50")` となっている。
そして、実は `int` が文字列"50"を数値50に変換するツール！
ゆえに、結果的に数値の50が返される



数当てゲームを作る材料

- 変数
- print
- if
- else

- random
- input

皆さんがすでに学んだツール
(すなわち準備完了！)





数当てゲームの全体



```
import random
```

```
answer = random.randint(1, 6)
```

数字をランダムに選んでanswerに代入

```
guess = int(input("1~6の数字を当ててみて! : "))
```

数字を入力させてguessに代入

```
if guess == answer:
```

当たり判定 (同じかどうか)

```
    print("正解!")
```

```
else:
```

```
    print("はずれ!")
```



練習：実際に数当てゲームを作ってみよう！



```
import random
```

```
answer = random.randint(1, 6)
```

```
guess = int(input("1~6の数字を当ててみて！："))
```

```
if guess == answer:
```

```
    print("正解！")
```

```
else:
```

```
    print("はずれ！")
```

うーん、正解は
何だったんだろう...



```
1~6の数字を当ててみて！： 3  
はずれ！
```



練習：実際に数当てゲームを作ってみよう！ (本当の答えを教えてくれるバージョン)



```
import random
```

```
answer = random.randint(1, 6)
```

```
guess = int(input("1~6の数字を当ててみて! : "))
```

```
if guess == answer:
```

```
    print("正解!")
```

```
else: ①ここにfを付ける
```

```
    print(f"はずれ! 本当は{answer}")
```

②{ }の中に表示したい
変数名

printの進んだ
使い方で!

fを付けるのが
ポイント!



1~6の数字を当ててみて! : 3
はずれ! 本当は6



本当は6だったのか



プログラムをタイプするのが面倒な方へ：
前ページの「練習」コピペ用ページ

```
import random
```

```
answer = random.randint(1, 6)
```

```
guess = int(input("1~6の数字を当ててみて! : "))
```

```
if guess == answer:
```

```
    print("正解!")
```

```
else:
```

```
    print(f"はずれ! 本当は {answer}")
```

Google Colabにペーストできない場合、
windowsなら一旦メモ帳にペーストして、
そこから再度コピペするとうまくいかも

もしかしたら、インデント部分に
こんな謎の黄色い□が出てくるかも。
それは消そう。



```
answer = random.randint(1,
guess = int(input("1~60
```

いままで学習した内容だけでできる「進んだ練習」： ちょっとしたロールプレイングゲーム！



```
import random

print("あなたは勇者です。")
print("モンスターがあらわれた！")

print("0: たたかう")
print("1: にげる")
action = int(input("どうする？ 0か1を入力してください："))

if action == 0:
    player = random.randint(1, 6)
    monster = random.randint(1, 6)

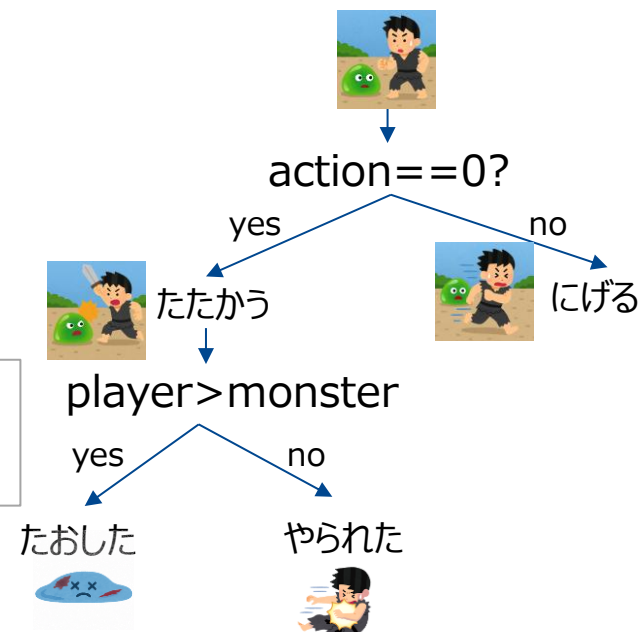
    print(f"あなたの攻撃：{player}")
    print(f"モンスターの攻撃：{monster}")

    if player > monster:
        print("モンスターをたおした！")
    else:
        print("やられてしまった…")
else:
    print("あなたはにげだした！")
```

ifの中に、
またifが！

実行例

あなたは勇者です。
モンスターがあらわれた！
0: たたかう
1: にげる
どうする？ 0か1を入力してください：0
あなたの攻撃：6
モンスターの攻撃：2
モンスターをたおした！



前ページの「進んだ練習」 コピペ用ページ



```
import random

print("あなたは勇者です。")
print("モンスターがあらわれた！")

print("0: たたかう")
print("1: にげる")
action = int(input("どうする？ 0か1を入力してください："))

if action == 0:
    player = random.randint(1, 6)
    monster = random.randint(1, 6)

    print(f"あなたの攻撃：{player}")
    print(f"モンスターの攻撃：{monster}")

    if player > monster:
        print("モンスターをたおした！")
    else:
        print("やられてしまった…")
else:
    print("あなたはにげだした！")
```

