cs-11. 条件分岐，リストと繰り返し，ステップ実行

（コンピューターサイエンス）

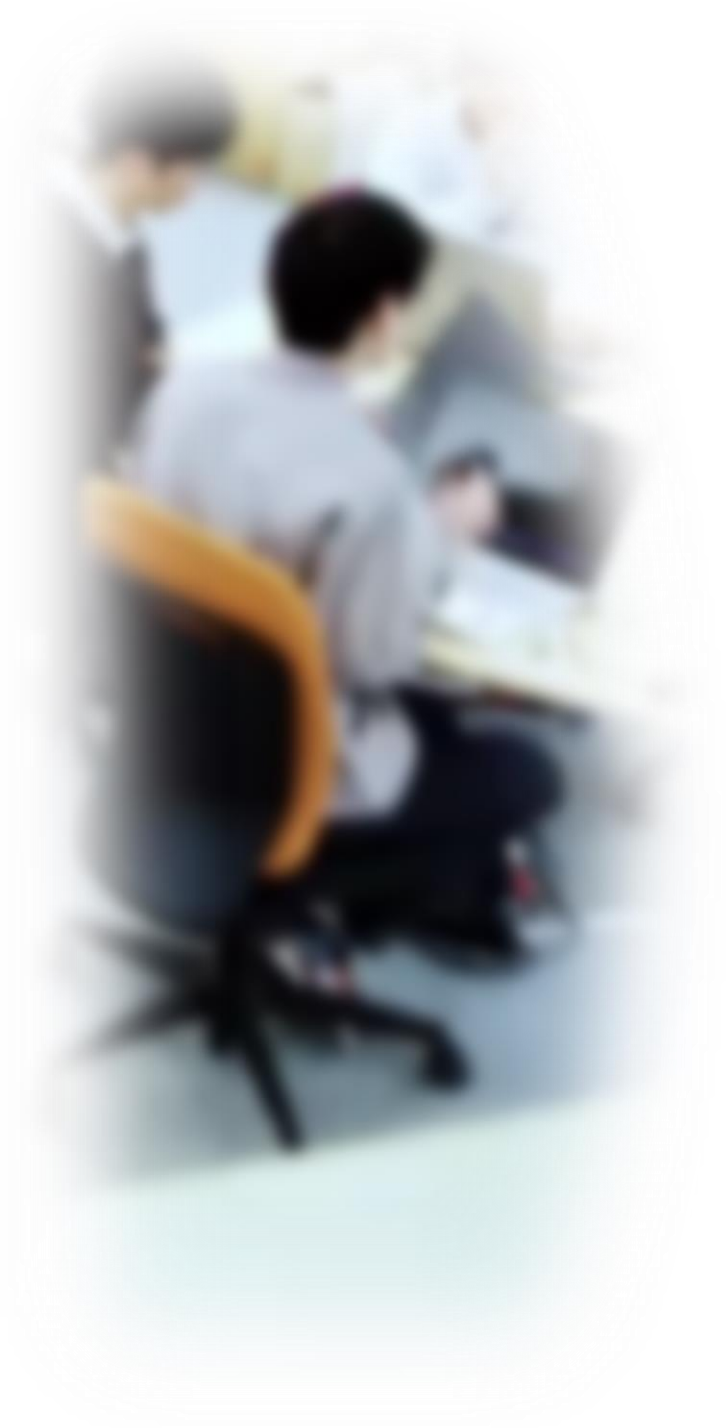
URL: https://www.kkaneko.jp/cc/cs/index.html

金子邦彦

①コンピュータでのプログラム実行は、通常実行が基本

②プログラムの流れの制御

③複数のデータをまとめて扱うリスト



アウトライン

1. 条件分岐
2. 条件分岐のステップ実行
3. 演習問題
4. リストと繰り返し

11-1. 条件分岐

条件分岐

条件分岐では，**変数や式の値**によって**結果が変わる**などの判断を行う

age の値が **11**以下　→ **500**

12以上　→ **1800**

条件式は「**age <= 11**」のようになる

条件分岐の Python プログラム

age の値が **11**以下　→ **500**

12以上　→ **1800**

```
age = 18
if age <= 11:
    print(500)
else:
    print(1800)
```

条件式は「**age <= 11**」のようになる

trinket

- **Trinket** は**オンライン**の **Python**、**HTML** 等の**学習サイト**
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）

- **Python の標準機能**を登載、その他、次のモジュールやパッケージがインストール済み

math, matplotlib.pyplot, numpy, operator, processing, pygal, random, re, string, time, turtle, urllib.request

trinket でのプログラム実行

- **trinket は Python, HTML などのプログラムを書き実行できる**サイト
- https://trinket.io/python/0fd59392c8

のように、違うプログラムには違う URL が割り当てられる

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習

資料：**10 ～ 11**

【トピックス】

- **条件分岐**
- **if**
- **else**

① trinket の次のページを開く

https://trinket.io/python/0fd59392c8

② 実行する。1800 が表示されることを確認

③ 「age = **18**」を「age = **10**」に書き替える

main.py

```
age = 10
if age <= 11:
    print(500)
else:
    print(1800)
```

Powered by
500

④ 実行する。**500 が表示される**ことを確認

Result

Powered by trinket
500

⑤ age の値が 8, 9, 10, 11 のときは 500 になり、12, 13, 14, 15 のときは 1800 になることを確認

条件分岐　まとめ

- **条件分岐**は、特定の条件に基づいて、異なる結果を得ることを可能にする

- Python の**条件分岐**では、if, else などのキーワードを使用

ある映画館で、11歳以下のチケットと，12歳以上のチケットで値段の違いがあるとき、条件分岐を使用して、チケット料金を算出できる

```
age = 18
if age <= 11:
    print(500)
else:
    print(1800)
```

age <= 11 のときは、print(500) が実行される

そうでないときは、print(1800) が実行される

11-2. 条件分岐のステップ実行

ステップ実行

- **ステップ実行**では、**１行ずつの実行**が行われ、そのときの変数の値の変化などを**確認**できる

- **ステップ実行**により、**プログラムの動作を細かく追跡**でき、不具合が発生している箇所の特定、プログラムの学習に役立つ

- **通常実行**は、**プログラムを最初から最後まで一度に実行**するもの（プログラム実行中の変数の値の変化を確認するなどは困難）。**ステップ実行**は、**プログラムを１行ずつ実行**し、**実行後にプログラムを一時停止**するもの。

条件分岐の Python プログラム

age の値が **11**以下　→ **500**

12以上　→ **1800**

```
age = 18
if age <= 11:
    print(500)
else:
    print(1800)
```

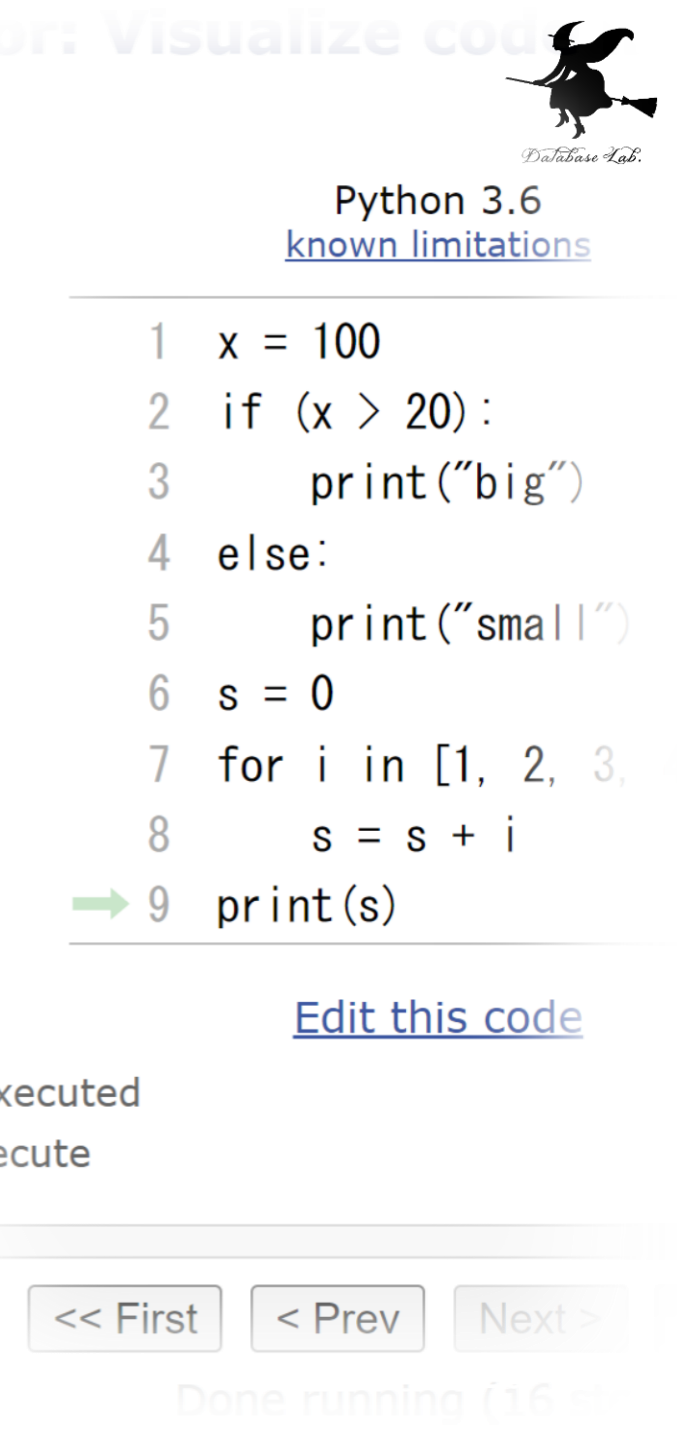
条件式は「**age <= 11**」のようになる

Python Tutor

- **Python Tutor** という**ウェブサイト**を利用しよう
  http://www.pythontutor.com/

- **Web ブラウザ**を使ってアクセスできる

- **PythonTutor** では，**Pythonだけでなく**，Java，C,，C++，JavaScript，Ruby など，多くのプログラミング言語を学ぶことができる.



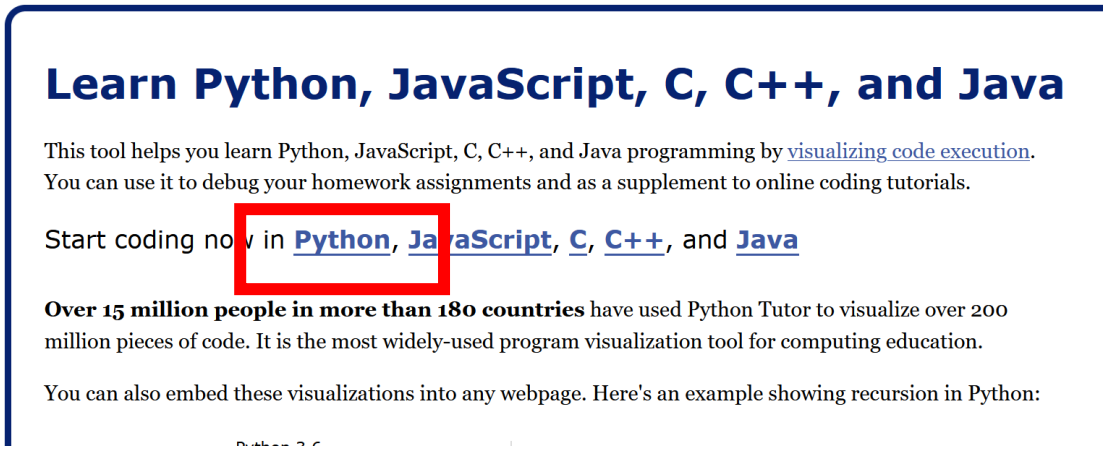
Python Tutor の使用方法

① まず，**ウェブブラウザ**を開く

② **Python Tutor** を利用するために，以下の URL にアクセス

http://www.pythontutor.com/

③ 「**Python**」をクリック　⇒ **編集画面**が開く



Python Tutor の編集画面

Python debugger - pdb interface to Python Tutor - Learn Python by visualizing code (also debug JavaScript, Java, C, and C++ code)

Write code in Python 3.6

「Python 3.6」になっている

1

エディタ
（プログラムを書き換えることができる）

Visualize Execution

実行のためのボタン

hide exited frames [default]
inline primitives, don't nest objects [default]
draw pointers as arrows [default]

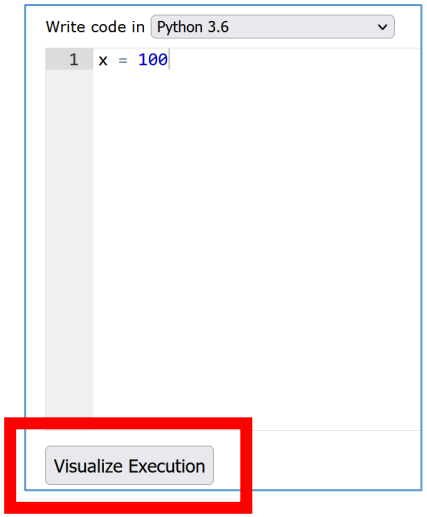
Show code examples

Generate permanent link

Python Tutor でのプログラム実行

- **Python Tutor** は **Python などのプログラムを書き実行できる**サイト．**ステップ実行**、**変数の値表示**などの機能がある。
- **Python Tutor**のウェブサイトにアクセス．「**Python**」を選択
  https://www.pythontutor.com/

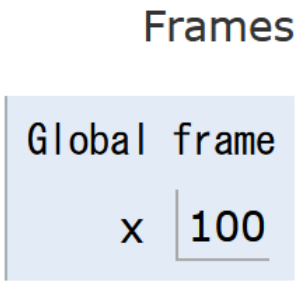
メイン画面で、プログラムを書く



Visualize Execution ボタン

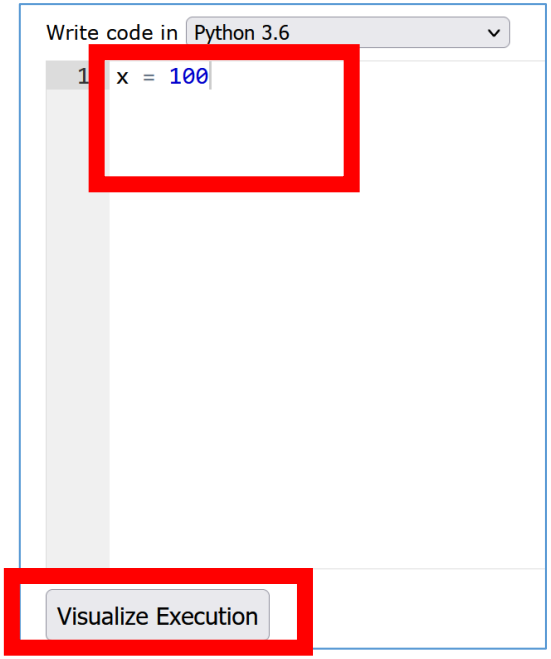
メイン画画に
戻るには
Edit this code

変数の値を
視覚的に
確認できる



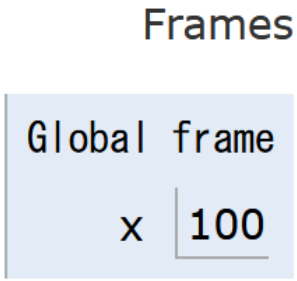
通常実行: Last
ステップ実行: 他のボタン

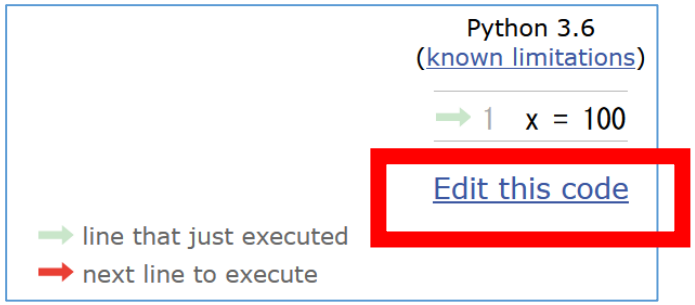
Python Tutor でのプログラム実行手順



(1)「**Visualize Execution**」をクリックして**実行画面**に切り替える

(2)「**Last**」をクリック.





(3) **<u>実行結果を確認</u>**する.

(4)「**Edit this code**」をクリックして**編集画面**に戻る

Python Tutor 使用上の注意点①

実行画面で，赤いエラーメッセージが出ることがある

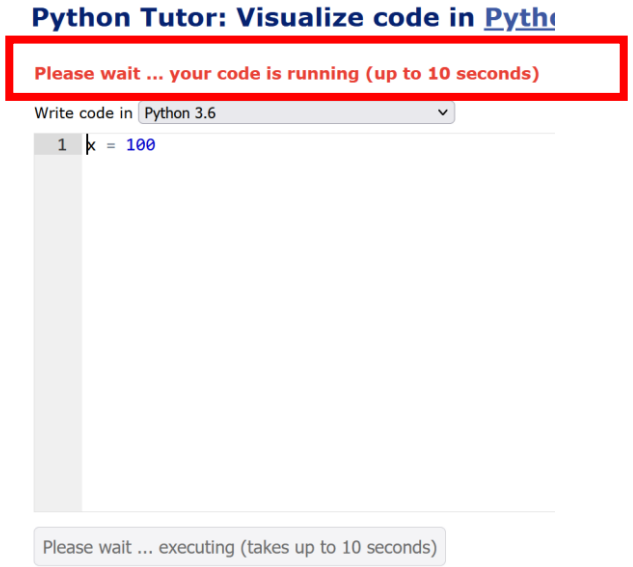
過去の文法ミスに関する確認表示.

基本的には，**無視**して問題ない

邪魔なときは「**Close**」

Python Tutor 使用上の注意点②

「**please wait ... executing**」のとき，**１０秒ほど待つ**.



- **Python Tutor が混雑しているとき，「Server Busy・・・」と表示される場合**がある.
- このメッセージは，サーバが混雑していることを示す.
- **数秒から数十秒待つ**と**自動で処理が始まる**はずです（しかし，表示が変わらないときは，操作をもう一度試してください）

演習

資料：**24 ～ 33**

【トピックス】

- **Python Tutor**
- **字下げ**
- **:**
- **条件分岐**
- **if**
- **else**
- **ステップ実行**

ステップ実行により確認できること

ステップ実行により，ジャンプの様子を観察

ジャンプの様子を示す矢印

Python 3.6

```
1  age = 30
2  if age <= 12:
3      print(500)
4  else:
5      print(1200)
```

Edit this code

Print output (drag lower right

Frames

Global frame

age 30

変数の値の確認もできる

<< First | < Prev | Next > | Last >>

Step 3 of 3

ステップ実行は、ボタンやスライダーでコントロール

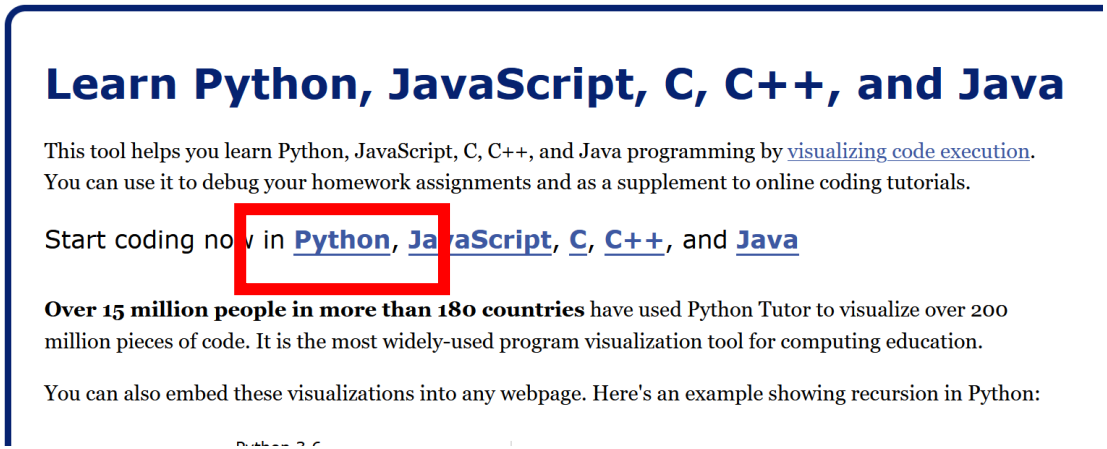
Python Tutor の起動

① **ウェブブラウザ**を起動する

② **Python Tutor** を使いたいので，次の URL を開く

https://www.pythontutor.com/

③ 「**Python**」をクリック　⇒ **メイン画面**が開く



Learn Python, JavaScript, C, C++, and Java

This tool helps you learn Python, JavaScript, C, C++, and Java programming by visualizing code execution. You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start coding now in Python, JavaScript, C, C++, and Java

Over 15 million people in more than 180 countries have used Python Tutor to visualize over 200 million pieces of code. It is the most widely-used program visualization tool for computing education.

You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:

④ Python Tutor のエディタで次のプログラムを入れる

```
1  age = 18
2  if age <= 11:
3      print(500)
4  else:
5      print(1800)
```

if (age <= 11)の直後に「:」
else の直後に「:」
（どちらも，コロン）

字下げも正確に！
print の前に，「タブ (Tab)」を 1つだけ

```
1  age = 18
2  if age <= 11:
3      print(500)
4      else:
5          print(1800)
```

正しくない字下げ

```
1  age = 18
2  if age <= 11:
3      print(500)
4  else:
5      print(1800)
```

正しい字下げ

「delキー」などを使いながら編集

⑤ **通常実行**するために，「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果 1800 を確認

Python 3.6
known limitations

```
1  age = 18
2  if age <= 11:
3      print(500)
4  else:
5      print(1800)
```

Edit this code

ecuted
cute

<< First | < Prev | Next > | Last >>

Done running (3 steps)

Print output (drag lower right corner to

1800

Frames | Objects

Global frame

age 18

結果の
「**1800**」を確認

⑥プログラム実行を最初の行に戻す操作

「**First**」をクリックして，**<u>最初の行に戻す</u>**

Python 3.6
known limitations

```
1  age = 18
2  if age <= 11:
3      print(500)
4  else:
5      print(1800)
```

Edit this code

executed
xecute

<< First | < Prev | Next > | Last >>

Done running (3 steps)

⑦「**Step 1 of 3**」と表示されているので，

全部で，**ステップ数**は **3** あることが分かる

（ステップ数と，プログラムの行数は**違うもの**）

Python 3.6

known limitations

```
1  age = 18
2  if age <= 11:
3      print(500)
4  else:
5      print(1800)
```

Edit this code

ecuted

cute

<< First | < Prev | Next > | Last >>

Step 1 of 3

⑧ **ステップ実行**したいので，「**Next**」をクリックしながら，**矢印の動きを確認.**

※「**Next**」ボタンを何度かクリックし，それ以上進めなくなったら終了

見どころ
2行目から **5行目**へ
ジャンプするところ

Python 3.6
known limitations

```
1  age = 18
2  if age <= 11:
3      print(500)
4  else:
5      print(1800)
```

Edit this code

xecuted
ecute

<< First | < Prev | Next > | Last >>

Step 3 of 3

⑨ First, Prev, Next, Last ボタンとスライダーによりプログラム実行を制御してみる

Python 3.6
known limitations

```
1  age = 18
2  if age <= 11:
3      print(500)
4  else:
5      print(1800)
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 2 of 3

⑩ メイン画面に戻るには、「**Edit this code**」をクリック

Python 3.6
known limitations

```
1  age = 18
2  if age <= 11:
3      print(500)
4  else:
5      print(1800)
```

Edit this code

: executed
execute

<< First | < Prev | Next > | Last >>

Done running (3 steps)

ステップ実行　まとめ

- **通常実行**は、**プログラムを最初から最後まで一度に実行**する

- **ステップ実行**は、**プログラムを１行ずつ実行**し、**実行後にプログラムを一時停止**するもの

- **ステップ実行**により、**プログラムの動作を細かく追跡**でき、不具合が発生している箇所の特定、プログラムの学習に役立つ

11-3. 演習問題

条件分岐

- 次のプログラムを作成

① weight と料金の関係は次の通り

weight の値が **100以下** → **0**

100より大きい → **1000**

② **weight = 80** に設定してテスト実行

正解の例

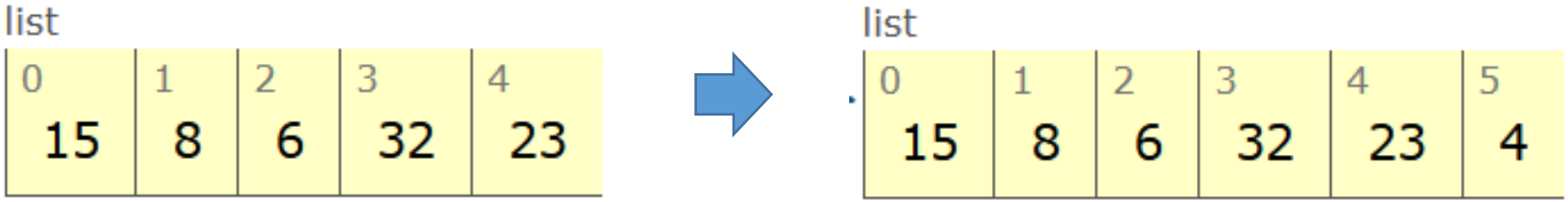
trinket のページ
https://trinket.io/python/62f74d3bfc

11-4. リストと繰り返し

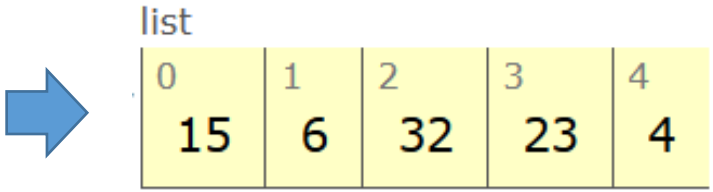
Python のリスト

- **リスト**は，**複数の要素を保持**できる
- リストの要素には順序を持ち、**順序の番号**は　**0 から開始**する
- リストの要素は変更可能（**新しい要素の挿入**，**既存の要素の削除**が可能）

4 を末尾に挿入



8 の削除



演習

資料：41 ～ 46

【トピックス】

- **リスト**
- **繰り返し**
- **for**

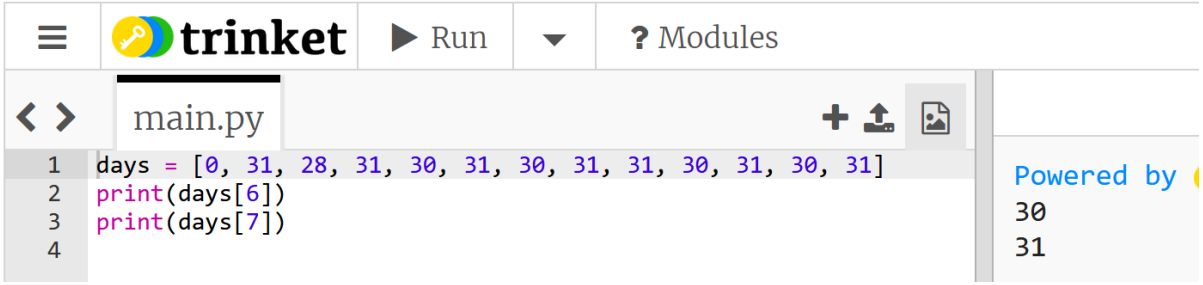
1．月の日数

6 月は **30**日まである．**7** 月は **31**日まである．
※ うるう年のことは考えないことにする

① trinket の次のページを開く

https://trinket.io/python/88a728c3cb

② 実行する。**30, 31 が表示**されることを確認



③ 8月や9月について表示できるように、プログラムを変更し実行してみる

リストの
組み立て

trinket ▶ Run ? Modules

main.py

```
1 days = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
2 print(days[6])
3 print(days[7])
4
```

Powered by

```
30
31
```

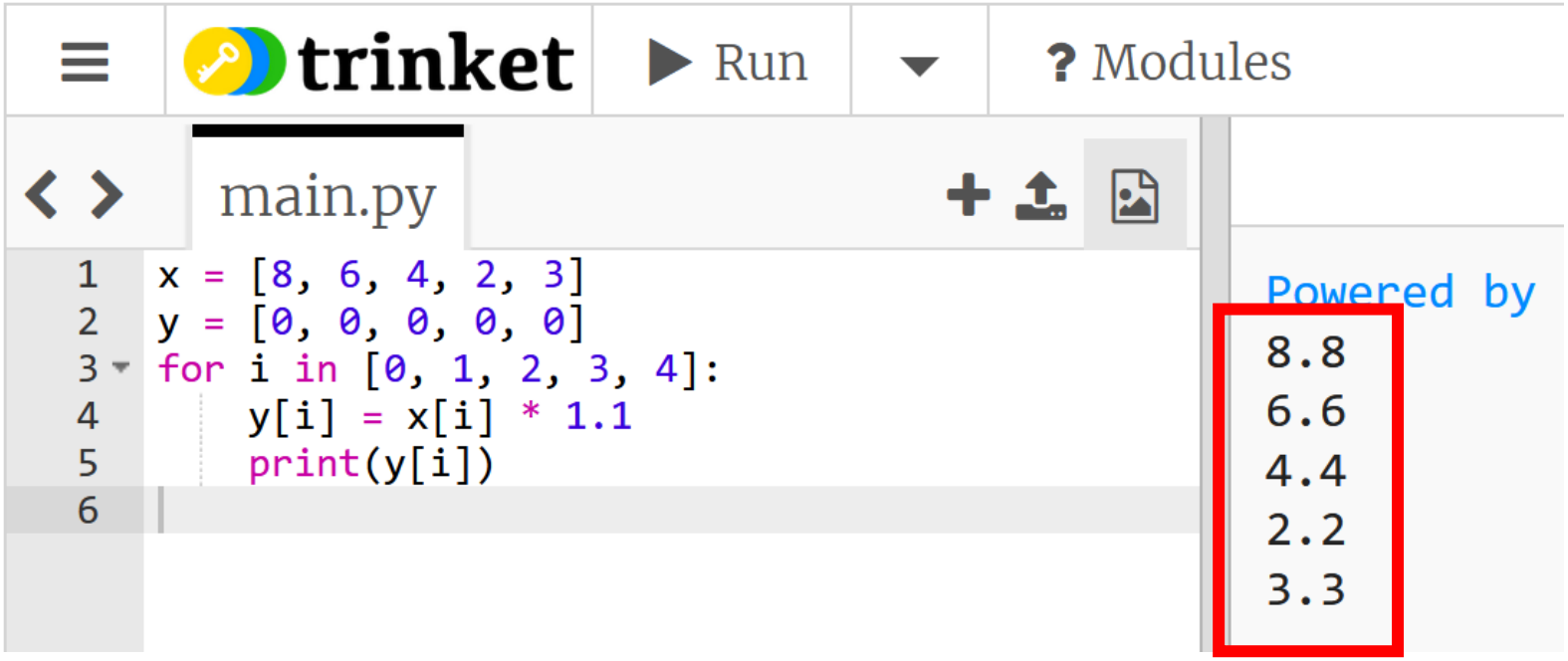
６番の要素、７番の要素の表示

2．計算の繰り返し

① trinket の次のページを開く

https://trinket.io/python/cc2c13d793

② 実行する。**結果が５つ表示**されることを確認



リストの
組み立て

main.py

```
x = [8, 6, 4, 2, 3]
y = [0, 0, 0, 0, 0]
for i in [0, 1, 2, 3, 4]:
    y[i] = x[i] * 1.1
    print(y[i])
```

Powered

```
8.8
6.6
4.4
2.2
3.3
```

「y[i] = x[i] * 1.1」を
i の値を変えながら
5回繰り返す

3．重力と落下距離

物体を落とすと **9.8 × (時間)^2 ÷ 2** の分，落ちていく．（空気抵抗は無視する）

① trinket の次のページを開く

https://trinket.io/python/e27702ef75

② 実行する。**結果が表示**されることを確認

main.py

```
x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for t in x:
    print((9.8 / 2) * t * t)
```

Resu

Powered by trinket

```
0.0
4.9
19.6
44.1
78.4
122.5
176.4
240.1
313.6
396.9
490.0
```

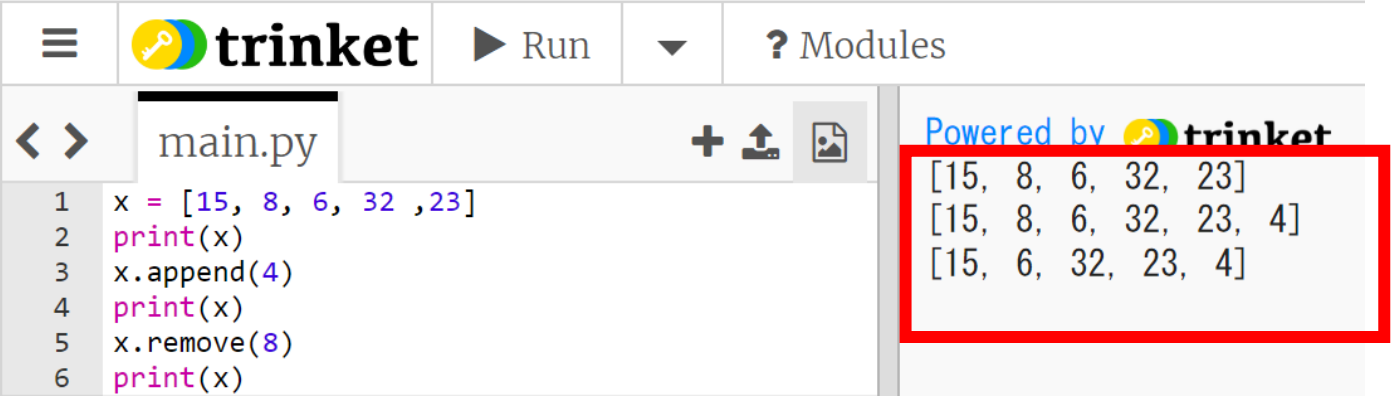
4．リストの組み立て，要素の挿入と削除

```
x = [15, 8, 6, 32 ,23]
print(x)
x.append(4)
print(x)
x.remove(8)
print(x)
```

① trinket の次のページを開く

https://trinket.io/python/f33df42c8d

② 実行する。**結果が表示**されることを確認



リストのプログラム例

trinket ▶ Run ? Modules

main.py

```
# リストの作成
months = ["", "January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"]
print(months[6])
```

Powered by

```
June
```

①コンピュータでのプログラム実行は、通常実行が基本

通常実行では、**プログラムは、最初から最後まで一度に実行**される。途中の変数の値を観察するには print やステップ実行を活用しよう。

②プログラムの流れの制御（条件分岐、繰り返し）

条件分岐（if など）や繰り返し（for など）は、プログラムの流れが制御される。**条件分岐**では、**特定の部分のみ実行**される。**繰り返し**では、プログラム実行が**繰り返される。**

③複数のデータをまとめて扱うリスト

リストの中の要素は順序を持ち、**特定の位置の要素にアクセス可能**

全体まとめ

- **通常実行**は、**プログラムを最初から最後まで一度に実行**する
- **ステップ実行**は、**プログラムを１行ずつ実行**し、**実行後にプログラムを一時停止**する。**プログラムの動作を細かく追跡**できる。
- **条件分岐**では、変数や式の値によって結果が変わるなどの**判断**を行う。**年齢（age）が11以下**であれば500を、それ以上であれば1800を出力するといった場合、**条件式**は「**age <= 11**」となる。
- Pythonの**リスト**は、**複数の要素を一度に保持**できる。リストの要素は順序を持ち、番号は0から始まる。

```
months = ["", "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]

print(months[6])
```

- **繰り返し**: 繰り返し計算では、特定の計算を**何度も行う**。