aa-9. 遷移関数、知的なゲーム世界の実現

（人工知能）

URL: https://www.kkaneko.jp/ai/mi/index.html

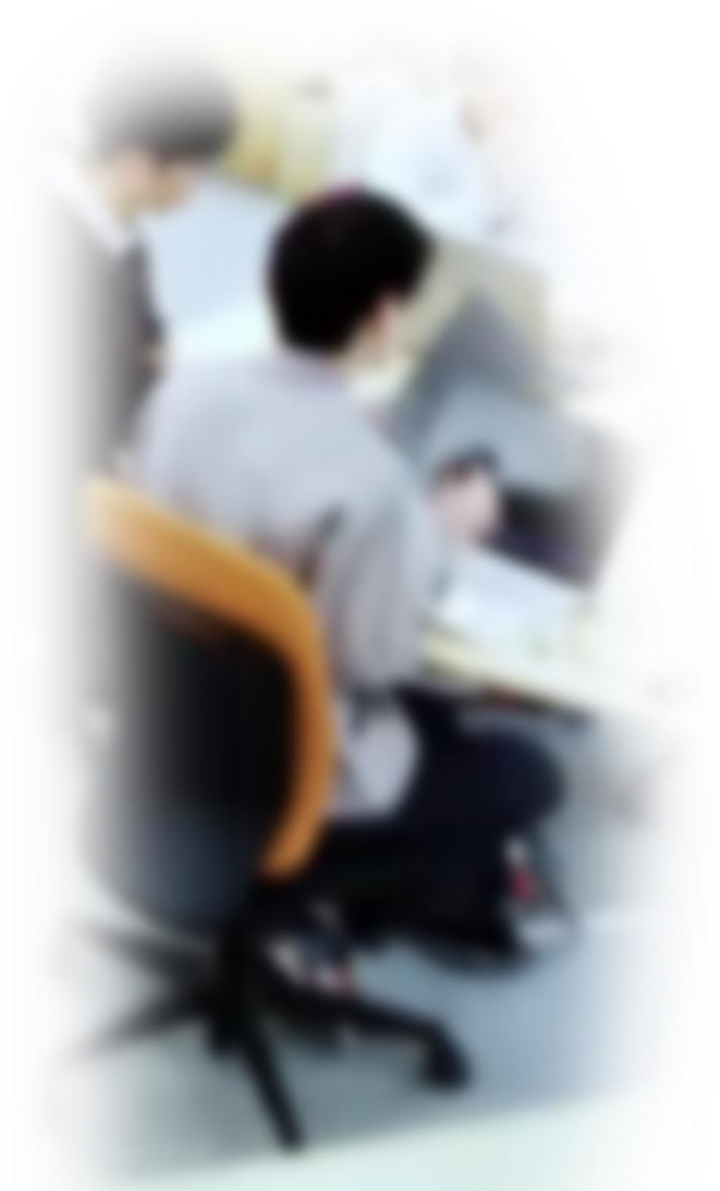
金子邦彦

① 遷移関数の概念説明

② 21ゲームという具体例を用いた遷移関数の理解の深化

③ Pythonプログラミングによる21ゲームの実装という実践に触れる

④ 遷移関数が，AIによる問題解決に役立つ場合があることを知る



アウトライン

1. 人工知能の種類
2. 遷移関数
3. コンピュータとプログラム
4. 状態、行動、遷移をコンピュータで実現
5. コンピュータプレイヤーがゲームに参加

9-1 人工知能の種類

人工知能の種類

人工知能

機械学習

データを用いて自ら学習できる人工知能

データを与えて学習させることで、
より正確に予測や分析や合成ができるように

知的な IT システム

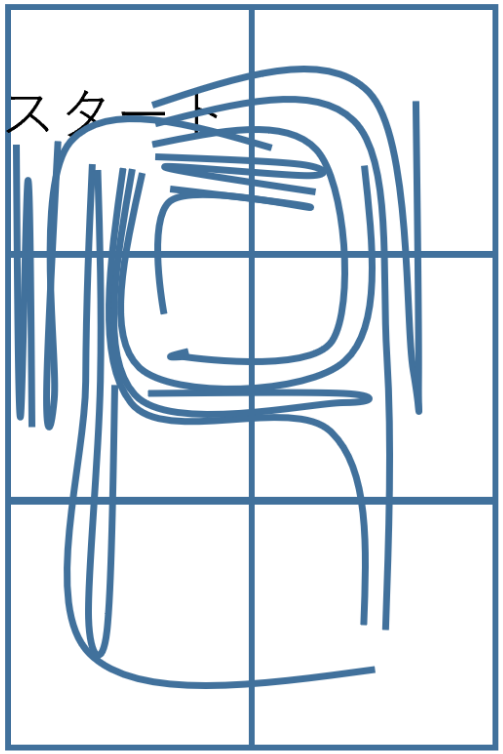
人間が書いた行動や知識を用いて、意思決定や問題解決を行う人工知能

人間が直接プログラムするため、判別性は高いが、柔軟性や汎用性は低いと考える人も

知的な IT システムの例

コンピュータはプログラムで動く

6 の小部屋．３回動くとどこにたどり着くか



解くためのプログラム

```
import itertools
import sys

def move(x, y, r):
    success = False
    if (r == 1) and (x < 1):
        x = x + 1
        success = True
    if (r == 2) and (x > 0):
        x = x - 1
        success = True
    if (r == 3) and (y < 2):
        y = y + 1
        success = True
    if (r == 4) and (y > 0):
        y = y - 1
        success = True

    return(x, y, success)

nsteps = 3
seq = [1, 2, 3, 4]
success = False
for j in list(itertools.product(seq, repeat=nsteps)):
    x, y = 0, 0
    for i in j:
        x, y, success = move(x, y, i)
        if(not(success)):
            break

    if(success):
        print("%s %d %d" % (str(j), x, y))
```

上下左右に
動くことができる
という行動を
プログラム化

得られた結果（抜粋）

```
(1, 2, 1) 1 0
(1, 2, 3) 0 1
(1, 3, 2) 0 1
(1, 3, 3) 1 2
(1, 3, 4) 1 0
(3, 1, 2) 0 1
```

9-2 遷移関数

遷移関数

- **遷移関数**は、**特定の行動**を取ったときに**現在の状態**が**どのように変化するか**を定める**規則**。

- AIが「**ある行動をとると結果としてどのような状態になるか**」を理解するために使用

遷移関数を学ぶ意義

- **AIでの応用**

遷移関数を用いて、現在の状態から**可能な次の状態**をすべて導き、その結果を評価して**最適な行動を選択**

- **問題解決スキルが向上する**

遷移関数は、AIだけでなく、論理的な問題解決全般に役立つ。
「**現在の状態**が**どのように変化するか**」を理解することで、解決策の立案、最善の結果の達成につながる。

遷移関数を学ぶことは、AIの理解を深めるだけでなく、問題解決スキルの向上にも有用

遷移関数を理解するために

① **遷移関数**

遷移関数は、**特定の行動**を取ったときに**現在の状態**が**どのように変化するか**を定める**規則である。**

② **遷移関数を学ぶ意義**

遷移関数の理解はAIの働きや、問題解決スキルの向上に**役立つ。**

③ 今から、遷移関数を具体的に理解していく

今から、「**エレベーターの遷移関数**」や「**21ゲームの遷移関数**」を学び、それらがどのように動作するかを理解することで、**遷移関数**の働きを**具体的に理解**する。

エレベーターの遷移関数

- **状態**

  エレベーターの現在の階数

- **行動**

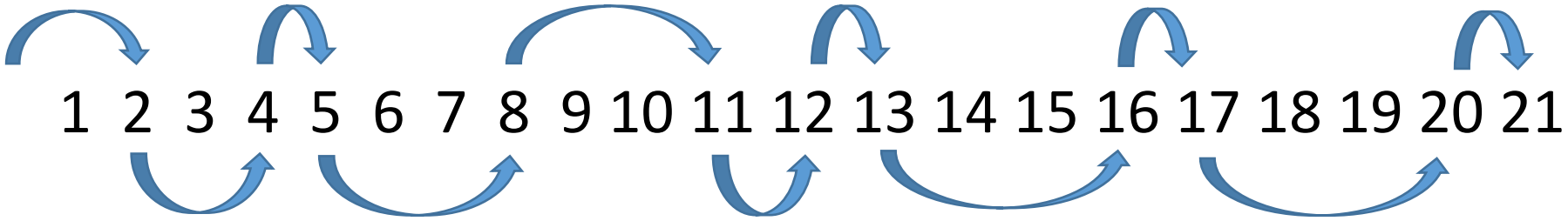
  エレベーターを**上に移動させる（行動１）**か、
  下に移動させる（行動２）か、
  停止する（行動３）

- **遷移関数**

  行動１：　もし、現在が最上階でなければ
  階数 = 階数 + 1　・・・階数が１増える
  行動２：　もし、現在が最下階でなければ
  階数 = 階数 – 1　・・・階数が１減る
  行動３：　階数 = 階数　・・・階数は変わらない

21 ゲームのルール

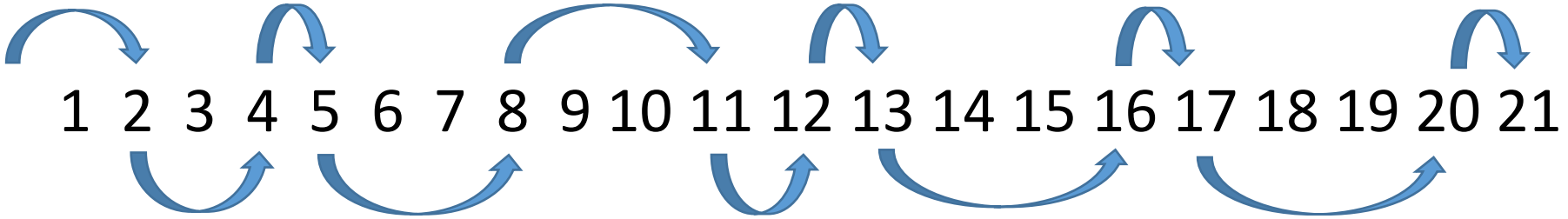
- ２人のゲーム
- ゲームは**「０」から開始**
- ２人で交互に、現在の数に、**１または２または３を足す**ことができる。これにより、数が増える。
- **２１を言った人が負け**とする



21 ゲームの対戦イメージ

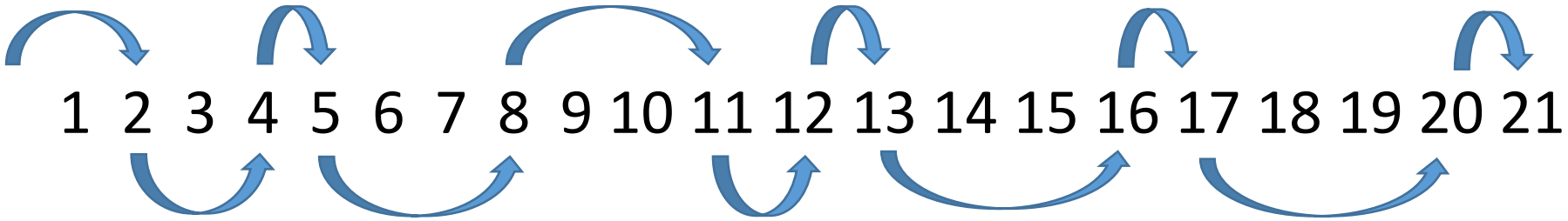
最初は０

先攻	後攻
１，２	３，４
５	６，７，８
９，１０，１１	１２
１３	１４，１５，１６
１７	１８，１９，２０
２１（負け）	



21 ゲームの魅力

- 21ゲームの面白さは、**戦略的な思考**が必要となるところにある。
- プレイヤーの目的は、**自分の番で21を作り出さない**ようにしつつ、**相手を21を作り出させる**ことである。
- そのため、プレーヤーは戦略的に数字１，２，３を選ばなければならない。



21 ゲームの遷移関数

- **状態**

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21　　※ 最初は 0 である

- **行動**

  １を足す（行動１）、２を足す（行動２）、３を足す（行動３）

- **遷移関数**

  行動１：　（数＜２１　のときのみ可能）

  数 = 数 + １　・・・数が１増える

  行動２：　（数＜２１　のときのみ可能）

  数 = 数 + ２　・・・数が２増える

  行動３：　（数＜２１　のときのみ可能）

  数 = 数 + ３　・・・数が３増える

遷移関数まとめ

- **遷移関数**は、**特定の行動**を取ったときに**現在の状態**が**どのように変化するか**を定める**規則である。**
- **状態**は、**具体的な問題や状況**を表す。

例：エレベーターでは、**現在の階数**が**状態**で、**エレベーターの上昇、下降、停止**が**行動**となる。

例：21ゲームでは、**現在の数**が**状態**で、**１、２、３の追加**が**行動**となる。

- **遷移関数**を用いて，現在の状態から**可能な次の状態**をすべて導きだすことができる。AI は，その結果を評価して**最適な行動を選択する**

9-3 コンピュータとプログラム

コンピュータとプログラム

- **コンピュータ**は，**プログラム**に従って動作
- **プログラム**は，**コンピュータ**に指示を出し，所定の作業を遂行させる

プログラミング

- **プログラム**を設計し作成するプロセス（プログラミング）は，創造的な活動
- **アイデアを形にできる**ことが，**プログラミング**の魅力

ソースコード

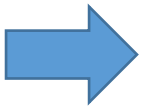
- **ソースコード**は，**プログラミング言語**で書かれた**プログラム**のもの
- 人間も**読み書き**，**編集**できる
- **ソースコード**により，**プログラムの動作を理解**し，必要に応じて**改変**できる

プログラミングの目的

- **プログラム**は，**コンピュータ**に指示を出し，所定の作業を遂行させる
- 複雑な作業も**自動化**し，効率化することが可能

```
a = [200, 400, 300]
for i in a:
    print (i * 1.08)
```

Python プログラムの
ソースコード



```
216.0
432.0
324.0
```

プログラムの
実行結果

9-4 状態、行動、遷移関数をコンピュータで実現

いまから行うこと

- **21 ゲーム**の**状態、行動、遷移関数**を，
  コンピュータの**プログラム**として書き，
  コンピュータのゲームとして動くようにする

- プログラミング言語として Python を使用する

21 ゲームの遷移関数

- **状態**

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21　　※ 最初は 0 である

- **行動**

  １を足す（行動１）、２を足す（行動２）、３を足す（行動３）

- **遷移関数**

  行動１：　（数＜２１　のときのみ可能）

  数 = 数 + １　・・・数が１増える

  行動２：　（数＜２１　のときのみ可能）

  数 = 数 + ２　・・・数が２増える

  行動３：　（数＜２１　のときのみ可能）

  数 = 数 + ３　・・・数が３増える

a は数
r は行動番号

（数＜２１　のときのみ可能）
数 = 数 + 1
・・・数が１増える

行動１

```
if (r == 1) and (a < 21):
    a = a + 1
```

選んだ行動が **1** のときは，
a < 21 を調べ，
a を **a + 1** に変化

プログラムの
ソースコード

演習

21 ゲームの**状態、行動、遷移関数**を実現する Python 言語プログラムを動作させ、理解を深める

【トピックス】

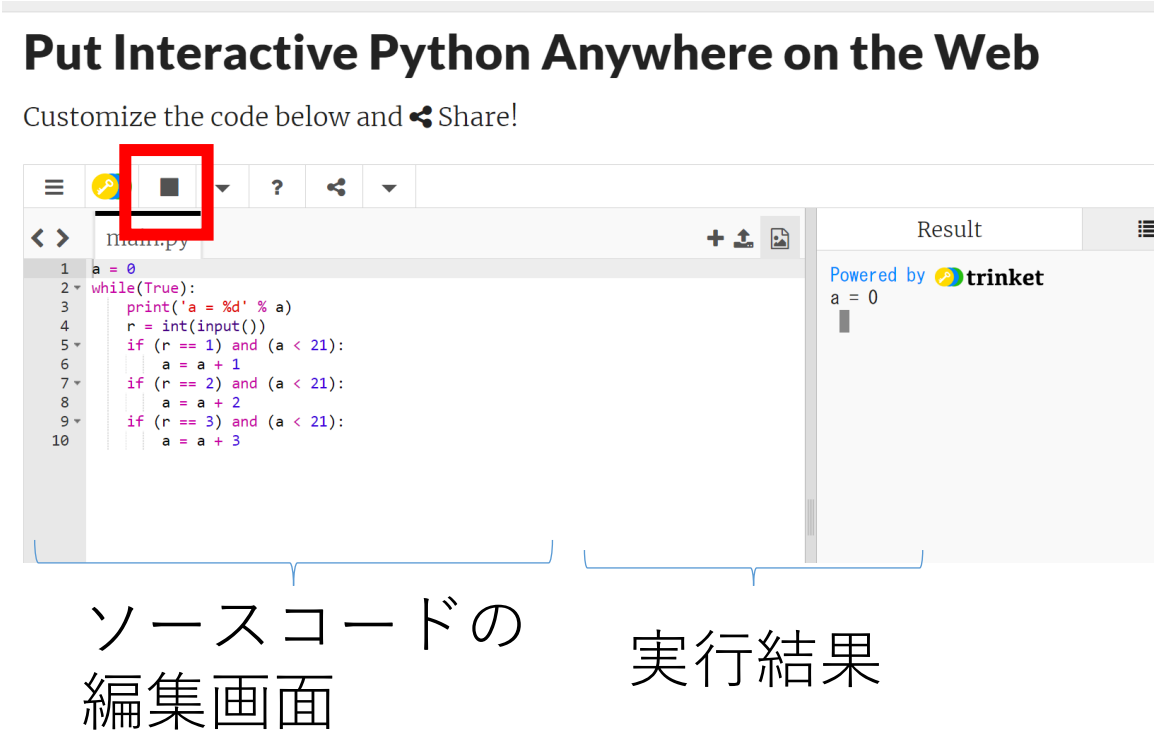
- **trinketでのプログラム実行**
- **状態、行動、遷移関数**

trinket でのプログラム実行

- URL: https://trinket.io/python/58bb317816

違うプログラムには違う URL が割り当てられる

実行、STOP ボタン



- 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

ソースコード

```
a = 0
while(True):
    print('a = %d' % a)
    r = int(input())
    if (r == 1) and (a < 21):
        a = a + 1
    if (r == 2) and (a < 21):
        a = a + 2
    if (r == 3) and (a < 21):
        a = a + 3
```

最初 a の値は 0

行動 1
行動 2
行動 3

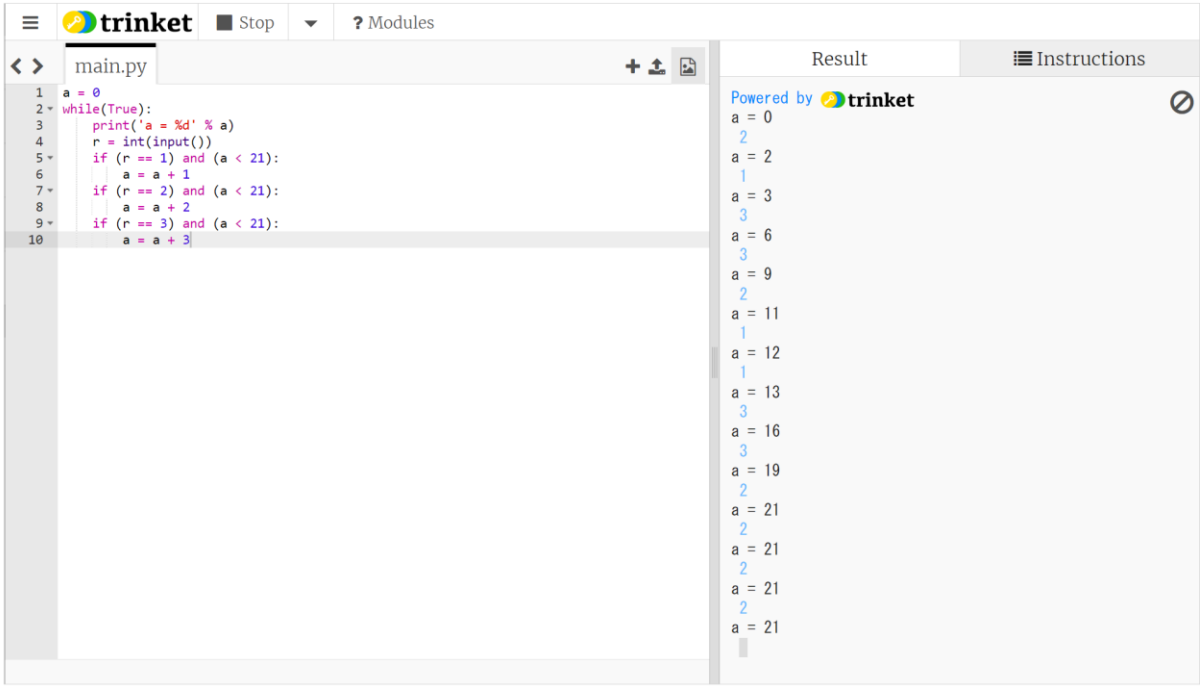
ユーザーから入力を受け取り（1、2、または3を足す）、それが21以下であればその数を足し、新しい値を出力する。

実行開始後の手順

① キーボードで、１＋Enter、２＋Enter、３＋Enter の操作を続ける

１：行動１　２：行動２　３：行動３

② 画面を確認する。２１になったらゲームは終わり。「**Stop**」をクリックして終了。



まとめ

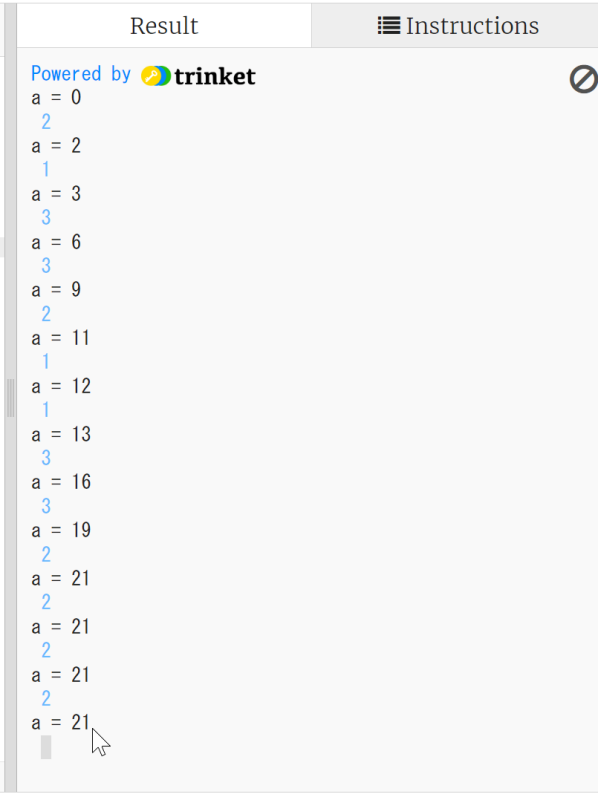
21 ゲームの**状態、行動、遷移関数**を，
コンピュータの**プログラム**として書き，
コンピュータのゲームとして動くようにする

遷移関数

行動１：　（数＜２１　のときのみ可能）
数 = 数 + １　・・・数が１増える

行動２：　（数＜２１　のときのみ可能）
数 = 数 + ２　・・・数が２増える

行動３：　（数＜２１　のときのみ可能）
数 = 数 + ３　・・・数が３増える



9-5 コンピュータ・プレイヤーがゲームに参加

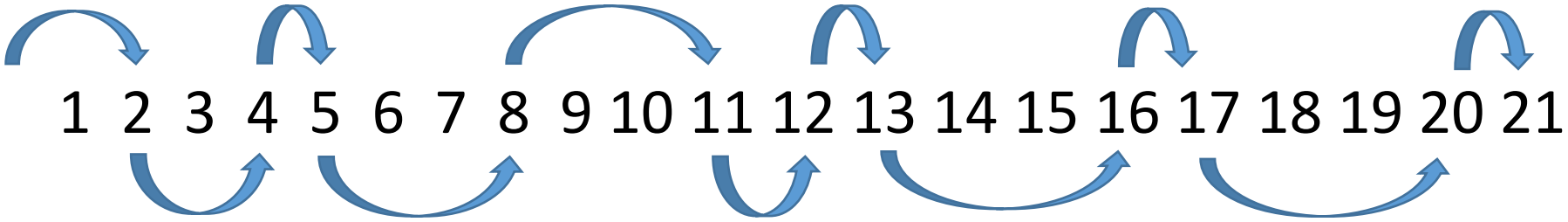
はじめに

- **コンピュータを，ゲームのプレイヤーとして振舞わせてみる**
- **コンピュータは知的に振舞う**

後攻は必勝

- 後攻での必勝のやり方

後攻は４，８，１２，１６，２０になるように数を増やす



後攻での必勝のやり方

今の数を **a** とする

- **a = 1** のとき，行動 **3** （a を 3 増やす）
- **a = 2** のとき，行動 **2** （a を 2 増やす）
- **a = 3** のとき，行動 **1** （a を 1 増やす）
- **a = 5** のとき，行動 **3** （a を 3 増やす）
-
-
-

（続く）

後攻での必勝のやり方

今の数を **a** とする

- **a = 1** のとき，行動 **3**
- **a = 2** のとき，行動 **2**
- **a = 3** のとき，行動 **1**
- **a = 5** のとき，行動 **3**
- **a = 6** のとき，行動 **2**
- **a = 7** のとき，行動 **1**
- **a = 9** のとき，行動 **3**
- **a = 10** のとき，行動 **2**
- **a = 11** のとき，行動 **1**
- **a = 13** のとき，行動 **3**
- **a = 14** のとき，行動 **2**
- **a = 15** のとき，行動 **1**
- **a = 17** のとき，行動 **3**
- **a = 18** のとき，行動 **2**
- **a = 19** のとき，行動 **1**

※ このやり方のとき，

後攻は，a = 4, 8, 12, 16, 20 で自分の手番がくることはない

今の数を **a** とする

a = 1 のとき，行動 **3**

行動番号を r とする

```
if (a == 1):
    r = 3
```

プログラムの
ソースコード

演習

21 ゲームでコンピュータープレイヤーを実現する Python 言語プログラムを動作させ、理解を深める

【トピックス】

- **trinketでのプログラム実行**
- **状態、行動、遷移関数**

ソースコード

```
def computer(a):
    if( a == 1 ):
        return 3
    elif( a == 2 ):
        return 2
    elif( a == 3 ):
        return 1
    elif( a == 5 ):
        return 3
    elif( a == 6 ):
        return 2
    elif( a == 7 ):
        return 1
    elif( a == 9 ):
        return 3
    elif( a == 10 ):
        return 2
    elif( a == 11 ):
        return 1
    elif( a == 13 ):
        return 3
    elif( a == 14 ):
        return 2
    elif( a == 15 ):
        return 1
    elif( a == 17 ):
        return 3
    elif( a == 18 ):
        return 2
    elif( a == 19 ):
        return 1
    else:
        return 0
```

コンピュータが自分の使う行動を決める

```
def move(a, r):
    if (r == 1) and (a < 21):
        a = a + 1
    if (r == 2) and (a < 21):
        a = a + 2
    if (r == 3) and (a < 21):
        a = a + 3

    return a

a = 0
while(True):
    print('a = %d' % a)
    r = int(input())
    a = move(a, r)
    print('a = %d' % a)
    r = computer(a)
    print('computer: %d' % r)
    a = move(a, r)
```

行動

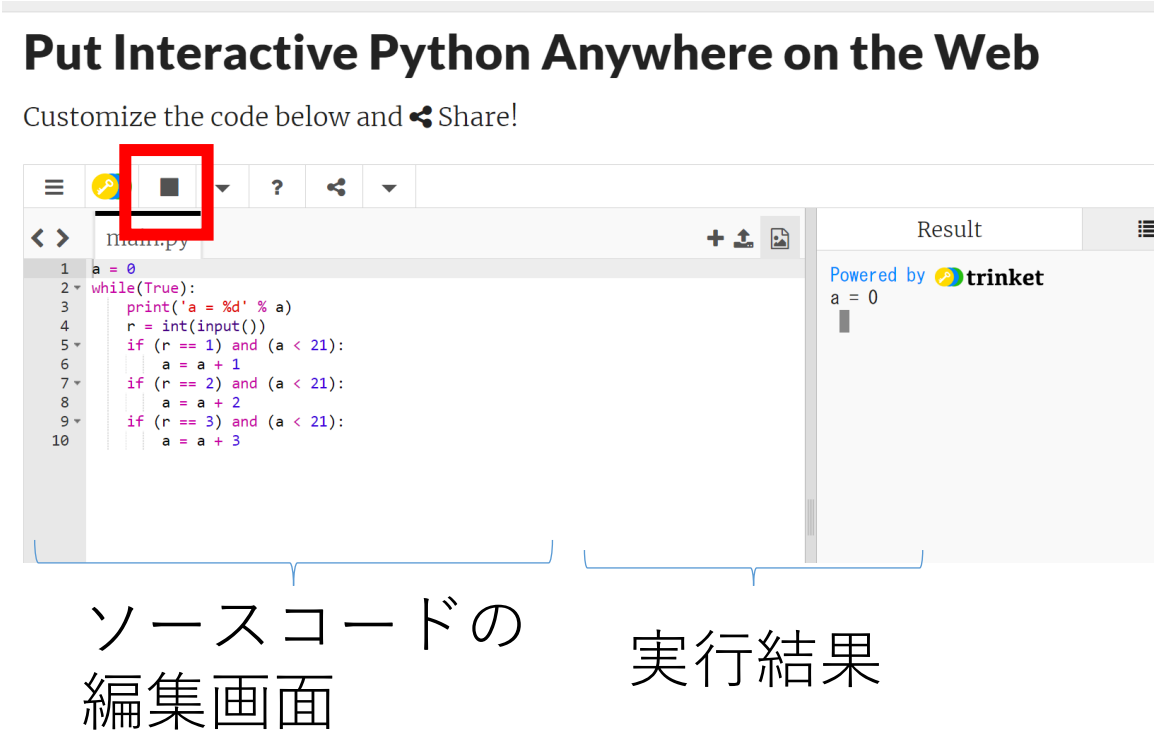
最初 a の値は 0

表示など

trinket でのプログラム実行

- URL: https://trinket.io/python/5468e2724a
- 違うプログラムには違う URL が割り当てられる

実行、STOP ボタン



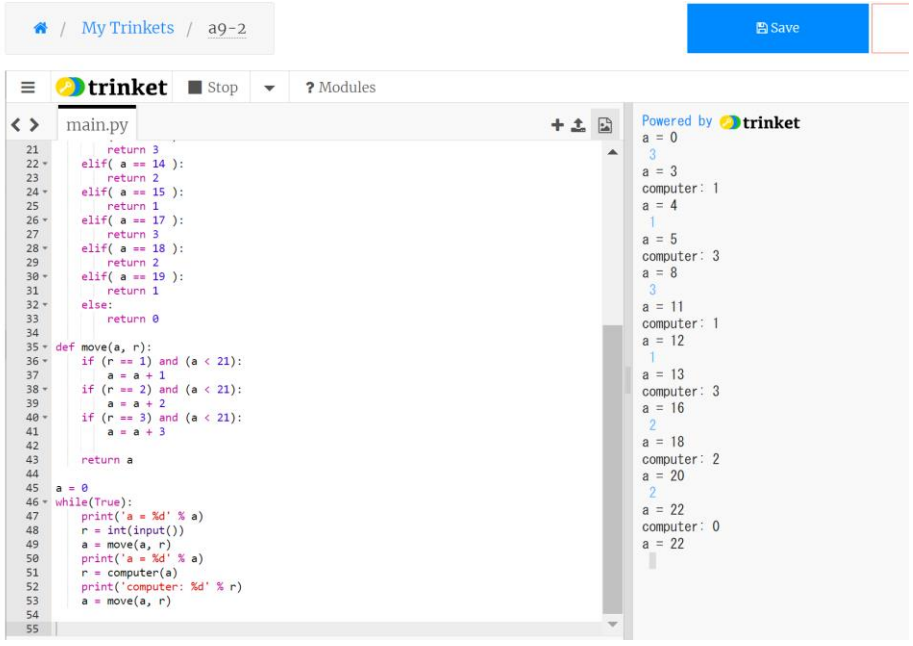
- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

実行開始後の手順

① キーボードで、１＋Enter、２＋Enter、３＋Enter の操作を続ける

１：行動１　２：行動２　３：行動３

② 今度はコンピュータが対戦相手である。画面を確認する。２１になったらゲームは終わり。「**Stop**」をクリックして**終了**。



ここまでのまとめ

コンピュータを，ゲームのプレイヤーとして振舞わせてみた

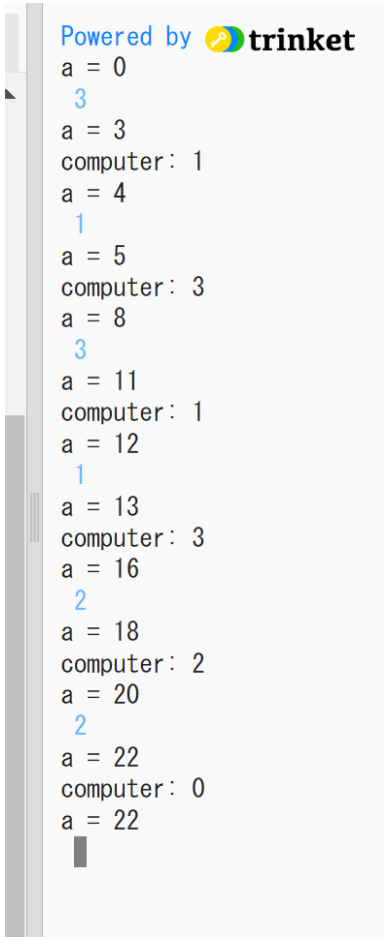
今の数を **a** とする

- **a = 1** のとき，ルール **3**
- **a = 2** のとき，ルール **2**
- **a = 3** のとき，ルール **1**
- **a = 5** のとき，ルール **3**
- **a = 6** のとき，ルール **2**
- **a = 7** のとき，ルール **1**
- **a = 9** のとき，ルール **3**
- **a = 10** のとき，ルール **2**
- **a = 11** のとき，ルール **1**
- **a = 13** のとき，ルール **3**
- **a = 14** のとき，ルール **2**
- **a = 15** のとき，ルール **1**
- **a = 17** のとき，ルール **3**
- **a = 18** のとき，ルール **2**
- **a = 19** のとき，ルール **1**

※ このやり方のとき，

後攻は，a = 4, 8, 12, 16, 20 で自分の手番がくることはない

後攻での必勝法



プログラム化し実行

全体まとめ

- **遷移関数**は、**特定の行動**を取ったときに**現在の状態**が**どのように変化するか**を定める**規則である。**
- 現在の状態から**可能な次の状態**をすべて導き、その結果を評価して**最適な行動を選択できるようになる**
- 「21ゲーム」では、**現在の数**が**状態**、１の追加と２の追加と３の追加が**行動**となる。**遷移関数**は**行動**により**状態がどう変化するか**を定める。
- **遷移関数**は、21ゲームに限らず、様々な問題解決に応用できる。

例：迷路の最短経路問題では、各マスを状態、移動方向を行動とみなす。遷移関数を用いて最適な経路を見つけることができる。

今回の授業で学ぶ意義と満足感

① 遷移関数の理解と実践

21ゲームを通した遷移関数の理解． Pythonプログラミングによる実行

② AIの動作原理の理解

問題解決への応用。遷移関数を AI の意思決定プロセスの基礎として利用可能

③ AIプログラミングスキル

21ゲームを題材とした状態、行動、遷移関数の理解

④ 実践的応用

AIシステム設計，AI応用での遷移関数の有用性