python™ プログラミング勉強会

公式サイトから資料を
DLできます。
パスワード: pk_py1116
puzzleknot.wp.xdomain.jp

puzzleknot
Tohoku University Programming Club

今日の流れ

- 第2回
- コンピュータにできること
- プログラム設計の考え方
- Python文法
  - データ構造
  - for文
  - if文
  - 関数

2

コンピュータにできること

3

できること

- コンピュータ = 計算機
- どのようなプログラムも突き詰めれば
  高速で計算を繰り返しているだけ

- 一般的なPC
  計算速度は 10^8 回/s くらい！

4

できること

- プログラム内では…
  - データを送受信する
  - データを変形する(計算はここに含まれる)

2つの処理が行われているだけ

5

できること

- 例：キーボードで打った文字を表示
  - 押されたキーの番号を**受け取る**
  - 入力モード(英数字or日本語 など)に応じて適切な文字コードに**変換**
    (復習：コンピュータは0と1の羅列で
    データを処理 => 文字コード)
  - ディスプレイに**出力**

6

プログラムの設計

7

プログラムはどう動く？

- 大原則
  - 上から下へ動く
  - 書かれたことしかやらない
  - 指示を待たない
    （ある行に書かれた処理が終わると
    即座に次の行の処理を行う）

8

プログラムはどう動く？

例えば、こんなプログラム

```
a = int(input())
b = int(input())
print(a + b)
```

9

プログラムはどう動く？

例えば、こんなプログラム

```
a = int(input())
b = int(input())
print(a + b)
```

変数（データの入れ物）a, bに
キーボードから受け取った入力を整数で
受け取る

↓

a + b という計算結果のデータを
ディスプレイに出力

10

プログラムはどう動く？

例えば、こんなプログラム

```
a = int(input())
b = int(input())
print(a + b)
```

変数（データの入れ物）a, bに
キーボードから受け取った入力を整数で
受け取る

↓

a + b という計算結果のデータを
ディスプレイに出力

11

プログラムはどう動く？

例えば、こんなプログラム

```
a = int(input())
b = int(input())
print(a + b)
```

変数（データの入れ物）a, bに
キーボードから受け取った入力を整数で
受け取る

※input() は入力を文字列として受け取る

↓

a + b という計算結果のデータを
ディスプレイに出力

12

プログラムはどう動く？

例えば、こんなプログラム

```
a = int(input())
b = int(input())
print(a + b)
```

変数（データの入れ物）a, bに
キーボードから受け取った入力を整数で
受け取る

※input() は入力を文字列として受け取る

↓

a + b という計算結果のデータを
ディスプレイに出力

13

プログラムはどう動く？

例えば、こんなプログラム

```
a = int(input())
b = int(input())
print(a + b)
```

変数（データの入れ物）a, bに
キーボードから受け取った入力を整数で
受け取る

↓

a + b という計算結果のデータを
ディスプレイに出力

14

プログラムはどう動く？

例えば、こんなプログラム

```
a = int(input())
b = int(input())
print(a + b)
```

変数（データの入れ物）a, bに
キーボードから受け取った入力を整数で
受け取る

↓

a + b という計算結果のデータを
ディスプレイに出力

15

プログラムはどう動く？

- コードと処理が**1対1対応**している
- 書いてあることを忠実にこなします

=> 逆に言えば、
書いてあること**しか**やりません！

- 仕様を正しく理解していないと
思わぬところでハマります
(例えばさっきの input() など)

16

プログラムはどう動く？

```
a = int(input())
b = int(input())
print(a + b)
```

a に 2, b に 3 を与えると
結果はそれぞれどうなるでしょう?

```
a = input()
b = input()
print(a + b)
```

17

プログラムはどう動く？

```
a = int(input())
b = int(input())
print(a + b)
```

output : 5　　…数字として足される

```
a = input()
b = input()
print(a + b)
```

output : 23　　…文字列として足される

18

プログラムはどう動く？

- 仕様を正しく理解していないと
  思わぬところでハマります
  (例えばさっきの input() など)
- 変にハマったら自分で調べる癖を
  付けましょう (エラーをコピペして
  検索 など…)

19

条件分岐とループ

20

プログラムはどう動く？

- プログラムは一直線に動くだけではありません
- 条件に応じた**判断**をさせることができます

- 例えば人間なら…
  - 赤信号 : 止まれ
  - 青信号 : 進め

21

プログラムはどう動く？

- プログラム風に書くと…

もし 赤信号なら:
　　止まれ
そうでなければ:
　　進め

22

プログラムはどう動く？

- Python風（Pythonic）に書くと…
  - 変数 traffic_light に信号の色のデータを入れておくことにしましょう

```
if(traffic_light == "赤"):
    stop()
else:
    go()
```

23

プログラムはどう動く？

- Python風（Pythonic）に書くと…
  - 変数 traffic_light に信号の色のデータを入れておくことにしましょう

```
if(traffic_light == "赤"):
    stop()
else:
    go()
```

if 文といいます!

24

if 文

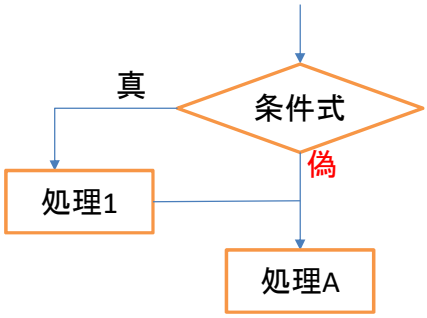
```
if(条件式):
    処理1
処理A
```

- 条件式が満たされると処理1を行います
- 処理Aはif文の結果に**関係なく**
  実行されます
  (復習：Pythonは意味のまとまりを
  インデントで示す)

25

if 文

```
if(条件式):
    処理1
処理A
```



26

if 文

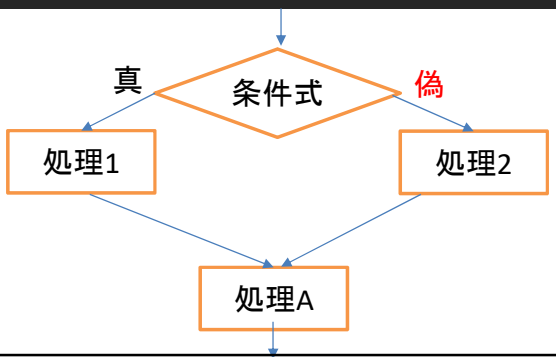
```
if(条件式):
    処理1
else:
    処理2
処理A
```

- 条件式が満たされると処理1を、
  満たされなければ処理2を行います
- 処理Aはif文の結果に**関係なく**
  実行されます(if文のまとまりの外)

27

if 文

```
if(条件式):
    処理1
else:
    処理2
処理A
```



28

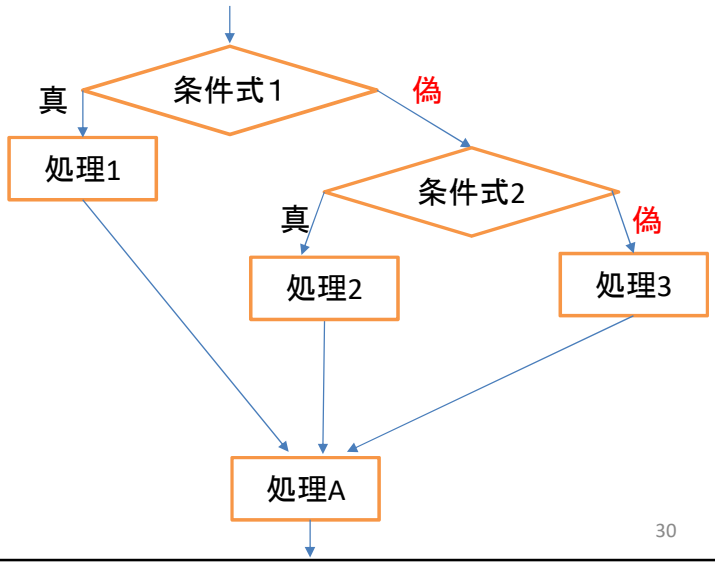
if 文

```
if(条件式1):
    処理1
elif(条件式2):
    処理2
else:
    処理3
処理A
```

- 3つ以上の分岐も可能です
- 条件式1が
  - 真 => 処理1
  - 偽 => 条件式2が
    - 真 => 処理2
    - 偽 => 処理3

29

if 文



30

条件式

- 条件式の書き方
- 数学の命題みたいなもんです
  - 満たされれば真(True)
  - そうでなければ偽(False)

31

条件式

- A == B … AとBの**値**が等しい
- A >= B … A ≧ B
- A <= B … A ≦ B
- A < B
- A > B

  } 同じ意味

- A != B … AとBの値が等しくない
  (A ≠ B)

- (A is B … AとBが同じ**オブジェクト**)
  - 例えば type(“hello”) is string

32

問題です

```
a = 10
if(a == 10):
    print("1")
elif(a >= 10):
    print("2")
else:
    print("3")
print("4")
```

- なんと出力されるでしょう？

33

問題です

```
a = 10
if(a == 10):          ← 真
    print("1")
elif(a >= 10):
    print("2")         ← あとのif文のまとまりは
else:                     無視される
    print("3")
print("4")            ← if文に関係なく実行
```

output : 1
4

34

問題です

```
a = 10
if(a == 10):
    print("1")
elif(a < 20):
    print("2")
if(a < 20):
    print("3")
    print("4")
print("5")
```

- なんと出力されるでしょう？

35

問題です

ブロック1 >

ブロック2 >

```
a = 10
if(a == 10):  ← 真
    print("1")
elif(a < 20):  ← あとのif文のまとまりは
    print("2")    無視される
if(a < 20):   ← 真
    print("3")
    print("4")
print("5")  ← if文に関係なく実行
```

output: 1
3
4
5

36

やってみよう

- じゃんけんゲームを作ってみましょう

- 入力を２つ受け取る
- G（グー）, C（チョキ), P（パー）で場合分け
- 各場合でさらに場合分け
  - if文の中にif文を書くとできます

37

やってみよう

- じゃんけんゲームを作ってみましょう

- ```
  if( a == “G”):
      if( b == “G”):
          print(“あいこ!”)
      elif( b == “C”):
          …
  ```

38
  ```

データ構造

- 復習: 変数はデータを格納するもの
- Pythonには他にも効率よくデータを扱う便利なデータ構造があります！

39

データ構造

- list
- データの集まりを数列のように扱う（C言語でいう配列）

- 厳密には、「**順序づけられた要素の集合**」
- C言語の100倍扱いやすいです

40

データ構造

- list
- 宣言
  a = [1, 2, 3] みたいな感じ

- 呼び出し
  print(a[0])　… 1 が出力
  a[0] = 1　　… 1 を代入
- index(添え字) は0から始まります

$$a_0 = 1$$

41

データ構造

- listは可変長です(大きさを変えられる)

- aというlistの末尾に値を追加する…
  - a.append(追加したい値)

- listのある位置に値を挿入する…
  - a.insert(挿入したい位置の直後のindex, 追加したい値)

42

データ構造

- listは可変長です(大きさを変えられる)
- listの特定の要素を削除する…
  - del a[削除したい位置のindex]
    または a.remove(削除したい値)
  - list内に同じ値が入っていると
    先頭のもののみ削除される

43

データ構造

- listをソートする
- a.sort()

```
a = [4, 1, 3, 2]
print(a)
a.sort()
print(a)
```

output : [4, 1, 3, 2]
[1, 2, 3, 4]

44

データ構造

- listの一部を取り出す（スライス）
- a[先頭のindex : 末尾のindex +1]

```
a = [0, 1, 2, 3]
print(a[1:3])
```

- output: [1, 2]

45

データ構造

- 他にもいろいろある
- tuple
  - 変更できない(immutableな) listのこと
  - 宣言は a = (1, 2, 3, 4) みたいな感じ
  - print(a[0]) …できる
  - a[0] = 値 …できない
  - a.append(値) …できない

46

データ構造

- 他にもいろいろある
- dict
  - キーと値をセットで保持する
  - indexの代わりにキーを指定する
    （順序付けされていない）
  - a = {“dog” : “イヌ”, (キー) : (値)}
    のように宣言する
  - a[追加したいキー] = 追加したい値
    で追加できる

47

データ構造

- 他にもいろいろある
- dict

```
a = {"dog": "イヌ", "cat": "ネコ"}
print(a["dog"])
a["dog"] = "犬"
print(a["dog"])
```

output: イヌ
犬

48

データ構造

- 他にもいろいろある
- set
  - 集合
  - 同じ値は**保持しない**
  - 順序付けされていない
  - 宣言: a = {3, 1, 4, 1, 5, 9}
    これは {1, 3, 4, 5, 9} という集合になる
  - 追加: a.add(追加したい値)
  - 削除: a.remove(削除したい値)

49

データ構造

- 他にもいろいろある
- いまは全部覚える必要ありません
  （使いたいときに調べよう）

- 参考: Dive into Python3

http://diveintopython3-ja.rdy.jp/native-datatypes.html

50

for 文

- 同じ処理を何度も繰り返したいとき…
  - 同じ文を何度も書くのはバカバカしい
- 処理をループさせると
  スマートにかける

51

for 文

- データ構造はこういうときに
  役立ちます
- a = [1, 2, 3, 4, 5, 6, 7]
- 各項を3で割ったあまりを計算したい

52

for 文

- ループを使わないと…

```
a = [1, 2, 3, 4, 5, 6, 7]
print(a[0] % 3)
print(a[1] % 3)
print(a[2] % 3)
print(a[3] % 3)
…
```

- やってられませんね

53

for 文

- ループを使うと…

```
a = [1, 2, 3, 4, 5, 6, 7]
for i in a:
    print(i%3)
```

- スマート！！

54

for 文

- ループを書く文はfor文です

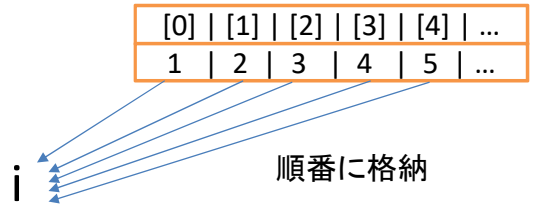
```
for i in リスト(タプル):
    処理
```

- i の中にリスト(タプル)の中身が順番に入れられていきます
- 1周目では i = a[0],
  2周目では i = a[1], となります

55

for 文

- i の中にリスト(タプル)の中身が順番に入れられていきます
- 1周目では i = a[0],
  2周目では i = a[1], となります



56

for 文

- i に渡されるのはあくまで値です

```
a = [1, 2, 3, 4, 5, 6, 7]
print(a)
for i in a:
    i += 1
print(a)
```

- これでaは書き換わりません
  output: [1,2,3,4,5,6,7]
  [1,2,3,4,5,6,7]

57

for 文

Q. リストとか使わないけど、 n回ループさせたいときはどうしたらいいの？

A. range(n) を代わりに書きます

```
for i in range(3):
    print(i)
```

output: 0
1
2

58

やってみよう

- Fizzbuzz
- 1から100までの値について…
  - 3で割り切れれば “Fizz”
  - 5で割り切れれば “buzz”
  - 両方で割り切れれば “Fizzbuzz”
  - どちらでも割り切れなければ値をそのまま
- forループ + if 文を組み合わせて
  書いてみましょう

59