プログラミング演習 I

授業開始までしばらくお待ちください。

授業前にこれを見ている人へ。

- 内容を先に読んで予習しておくことは大いに結構なことだと思います。ぜひ予習してください。
- 内容は随時更新しています。授業前に再度最新版を確認してください。
- 課題の提出 (Google Forms によるもの) は **授業当日の指示があるまで行わないように** してください。

プログラミング演習 I
Exercise on Programming I

本スライドは Teams で配布しています。

小林裕之・瀬尾昌孝
prog1@oitech.ddns.net

大阪工業大学 RD 学部システムデザイン工学科
OSAKA INSTITUTE OF TECHNOLOGY

2 of 14

a LaTeX + Beamer slideshow

第 2 回以降の課題の提出について

Google に工大組織アカウントでログイン

ポイント

- 課題提出は Google に**工大の組織アカウント**でログインした状態でないと行えない。
- 自分のログイン状態は **Google のページに行けばわかる**。
- プライベートなアカウントでログイン中だったら**一度ログアウト**して、あらためて組織アカウントでログイン。

次ページで説明する手順で確認しよう。

Google のログイン状態の確認

1. www.google.comを開く。

Google のログイン状態の確認

1. www.google.comを開く。

2. ここが……
   - 【ログイン】だったらクリックして**工大組織アカウントでログイン。**
   - そうでなければクリックし、表示が工大組織アカウントでなければ【ログアウト】して、**工大組織アカウントでログイン。**
   - そうでもなければ (工大組織アカウントなので) ok。

Google のログイン状態の確認

1. www.google.comを開く。

2. ここが……
   - 【ログイン】だったらクリックして**工大組織アカウントでログイン。**
   - そうでなければクリックし、表示が工大組織アカウントでなければ【ログアウト】して、**工大組織アカウントでログイン。**
   - そうでもなければ (工大組織アカウントなので) ok。

課題の自己チェックと提出: 全体のざっくりイメージ

授業では毎回課題を出します。

点数に納得できるまで 1 と 2 を繰り返し、最後に 3 で課題提出。

自分の PC

自己チェッカー

https://edu2.rd.oit.ac.jp:4000

G Forms

https://forms.gle/Rgo7xAzsRnUcaGqVA

課題の自己チェックと提出: 全体のざっくりイメージ

授業では毎回課題を出します。

点数に納得できるまで 1 と 2 を繰り返し、最後に 3 で課題提出。

1 `.py`を作る

自分の PC

自己チェッカー

https://edu2.rd.oit.ac.jp:4000

G Forms

https://forms.gle/Rgo7xAzsRnUcaGqVA

課題の自己チェックと提出: 全体のざっくりイメージ

授業では毎回課題を出します。

点数に納得できるまで 1 と 2 を繰り返し、最後に 3 で課題提出。

1 .pyを作る

自分の PC

2 .pyをチェック

自己チェッカー

https://edu2.rd.oit.ac.jp:4000

G Forms

https://forms.gle/Rgo7xAzsRnUcaGqVA

課題の自己チェックと提出: 全体のざっくりイメージ

授業では毎回課題を出します。

点数に納得できるまで 1 と 2 を繰り返し、最後に 3 で課題提出。

1 .pyを作る

自分の PC

2 .pyをチェック

自己チェッカー

https://edu2.rd.oit.ac.jp:4000

G Forms

https://forms.gle/Rgo7xAzsRnUcaGqVA

3 .pyを提出

課題の自己チェックと提出: 手順 2. 自己チェック

1. 学外からの場合は**VPN に接続**した状態で、
2. `https://edu2.rd.oit.ac.jp:4000`にアクセスし、
3. 課題番号を選択し、
4. 自分で作った`.py`ファイルをアップロードする。

これで課題の**自己採点**ができる。(満点を目指そう！)

課題の自己チェックと提出: 手順 3. 提出

- 最終提出は**Google Forms**を使う。(OIT 組織アカウントで Google にログインが必要。VPN は不要。)
- **課題提出 Google Forms**
  `https://forms.gle/Rgo7xAzsRnUcaGqVA`
  から**`.py`**ファイルをアップロードする。
- 提出できたら工大のメイルアドレスに**確認 e-mail が届く**ので必ず確認。
- 提出期間は「授業の日から次回の授業の前日」。これ以外には決して提出しないこと。(前回もしくは次回の課題として処理してしまうため。)

やってみよう。

以上、“hello, world” について

1. 作った`hello.py`を
2. 自己チェックして (学外からは**要 VPN**)
3. Forms から提出 (今回の課題の回答になっていないので満点はつきません。必ず授業後に正しい解答を再提出してください。)

してみよう。(今後の授業はこの流れで課題提出。)

話を、戻す。

練習

問

Ready と表示するプログラムを新規に作成し、実行せよ。

「VSCode のエクスプローラー」から【新しいファイル】で始める。

補足: 引用符について

- Python 言語では文字列を作る引用符にはクォーテーションマーク (") とアポストロフィ (') があるが、意味は **どちらも同じ**(どちらを使っても良い)。
- ただし「開き」と「閉じ」は整合している必要がある。

  ○ `print("hello")`

  × `print('world')`
- 応用編として、こんな使い方もあり。

  `print('"')`

一行から複数行へ

たった 1 行ではつまらないプログラムしか書けないか？

```
print('hello, world')    # つまらない
```

一行から複数行へ

たった 1 行ではつまらないプログラムしか書けないか？

```
print('hello, world')    # つまらない
```

いや、たった 1 行でも頑張ればいろいろできる。

```
print([x for x in range(2, 100) if all([x % i for i in range(2, x)])])
```

一行から複数行へ

たった 1 行ではつまらないプログラムしか書けないか？

```
print('hello, world')    # つまらない
```

いや、たった 1 行でも頑張ればいろいろできる。

```
print([x for x in range(2, 100) if all([x % i for i in range(2, x)])])
```

とは言え無理に頑張る必要もないので素直に**複数行**にしよう。

複数行のプログラム

当たり前だけれどとても重要なこと

次のプログラムはどのような結果になるか想像してみよう。

```
print('Ready')
print('Set')
print('Go')
```

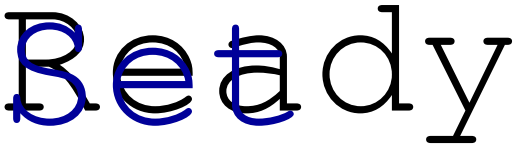
想像し終えたら追加の 2 行を入力してください。

予想その 1 (注: わかりやすくするために色をつけています。)

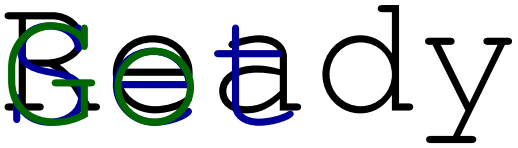
予想その 1 (注: わかりやすくするために色をつけています。)

Ready

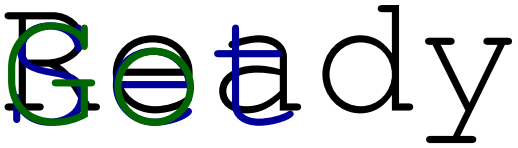
予想その 1 (注: わかりやすくするために色をつけています。)



予想その 1 (注: わかりやすくするために色をつけています。)



予想その 1 (注: わかりやすくするために色をつけています。)



……とはなりません。(まぁそりゃそうね)

予想その 2

予想その 2

Go
Ready
Set

予想その 2

Go
Ready
Set

……ともなりません。(ですよね、さすがに。)

予想その 3

予想その 3

ReadySetGo

予想その 3

ReadySetGo

……も違います。(へぇそうなんだ)

予想その 4

予想その 4

Ready
Set
Go

予想その 4

Ready
Set
Go

これが正解!

大変重要なこと

- プログラムは（基本的には）**上から、順番に、**実行される。

  この流れを変える**制御構造**[1]は少し先の授業でやる。

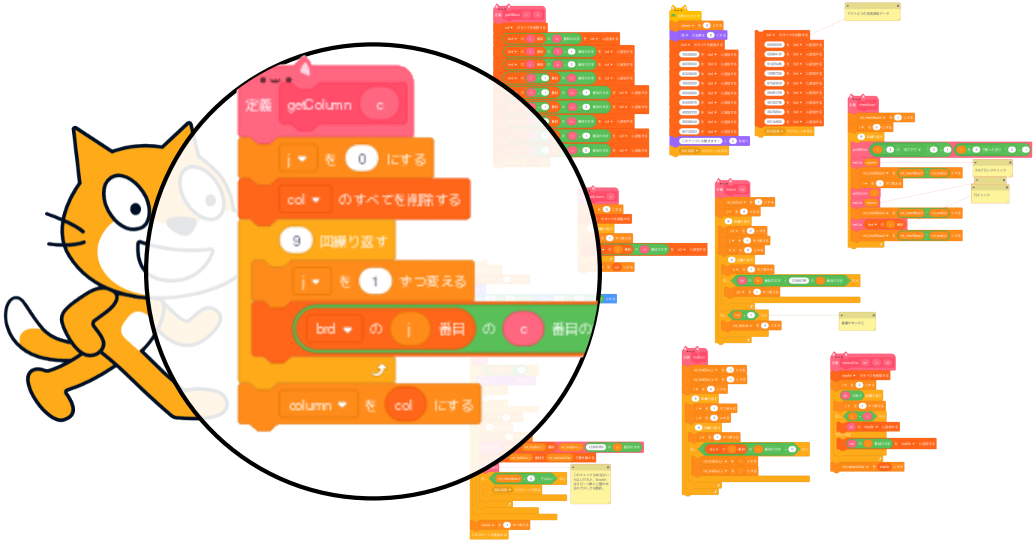
- **`print()`** 関数は表示したら改行する。

  これは Python 言語の `print` 関数の仕様なので前項に比べたら重要度はうんと低い。

[1] プログラミング経験がある人はご存知と思いますが、`if` や `while` などのこと。

余談: 『上から、順番に』？

余談: 『上から、順番に』？


定義 getColumn c
j を 0 にする
col のすべてを削除する
9 回繰り返す
j を 1 ずつ変える
brd の j 番目の c 番目の
column を col にする

コラム: 『上から、順番に』？

制御構造 (p. 18参照) は別として、基本的に**『上から、順番に』**というのは Python をはじめ、多くのプログラミング言語に共通した特徴ですが、例外もあります。特に手続き的な処理ではなく、定義については上から順番に書かなくても良いケースがある言語も多い気がします。以下の半径 2 の円の面積を求める Haskell と Rust のプログラムはどちらもエラーなく期待通りに動きます。ですが、右の Python はエラーになります。Python 初学者のみなさんはとりあえず今の段階では**『上から、順番に』**でいいです。

```haskell
-- Haskell
main = print s
s = p * r * r
r = 2
p = 3.14
```

```rust
// Rust
fn main() {
    println!("{S}");
}
const S: f32 = P * R * R;
const R: f32 = 2.0;
const P: f32 = 3.14;
```

```python
# Python(エラー)
print(s)
s = p * r * r
r = 2
p = 3.14
```

プログラムの中に人間向けのメモを書きたい！

こんな「メモ」を書けたら（人間にとって）プログラムが理解しやすくなるはず。

スタートを仕切るプログラム

```
print('Ready')  これは［位置について］の意
print('Set')    ［用意］
print('Go')     ［ドン］
```

…なのだが、これをそのままやったら**エラー**になってしまう。

コメント: プログラム中に書く（人間向けの）メモ

「#」から行末まではコメント。コンピュータは無視する。

```
# スタートを仕切るプログラム
print('Ready')  # これは [位置について] の意
print('Set')    #[用意]
print('Go')     #[ドン]
```

人間が理解しやすいように、コメントを積極的に入れよう！

文字列以外の `print()`

文字列と計算式と print()

- `print(文字列)`
- `print(計算式)`

文字列

引用符記号で囲んだ文字の並び
例: `"hello, world"`

計算式 (厳密な用語ではありません)

数字や括弧や演算子記号などの並びによる計算式
例: `(2 + 4) * 7`

例題: 半径 2 の円の面積を求めるプログラム

つまんねー、というご意見はごもっともですがお付き合いください。

```
print('半径 2 の円の面積は…')
print(3.14 * 2 * 2)
print(" です。")
```

問. $0.1 + 0.2$ を計算して出力するプログラムを作成せよ。

まとめ

- はじめての Python**関数**: `print()`
- プログラムは基本的に**上から、順番に**実行
- 引用符記号で囲んで作る**文字列**
- ふつうに計算できる**計算式**[2]
- 人間向けのメモが書ける**コメント**

[2] くどいようですが計算式というのは正確な用語ではありません。数値計算を行う**式 (expression)**を平たい言葉で表したもの。

paiza 自習

いろいろ終わってヒマな人はこちらをどうぞ。(「問題集」は音は出ないので授業中でも閲覧 ok。)

今回の内容に近い 5 問

1. 問題集 > チャレンジする言語 >Python3
2. 最初の一歩 > 自己紹介問題セット
3. D ランクレベルアップメニュー >2 つの数値を出力 (STEP1〜4)

課題: 以下の仕様のプログラムを作成せよ。

- 実行すると`123`と出力する。
- 余計なものがつかないこと。(例: `123.0`とかはダメ)
- コード中に「1」「2」「3」という文字を使わないこと。
- 何か (`#`で) コメントをつけること。

実行例

```
hkoba@mypc:~/prog1$ python3 num123.py
123
hkoba@mypc:~/prog1$
```

ヒント: `print(1 + 2)`というプログラムには「3」という文字は使われていないけれど実行すると「3」と出力される。

第 2 回 (4/16) 課題 (提出期間: 4/16〜4/22)

- 提出前に自己チェック。(学外からは**要 VPN**)
  `https://edu2.rd.oit.ac.jp:4000`
- 課題提出 Google Forms から提出。(VPN 不要)
  最後の 送信 クリックを忘れないこと！
- 提出できたら**確認 e-mail が届く**ので必ず確認。
- 期間外の提出は**自動的に** 別の課題を上書きするような処理をされてしまうので期間外の提出は厳禁。(早く提出するのも、遅れて提出するのもどっちもダメ。)

`https://forms.gle/Rgo7xAzsRnUcaGqVA`