



# プログラミング技法II

担当： 新田 直子

大学院工学研究科 電気電子情報工学専攻

naoko@comm.eng.osaka-u.ac.jp

<http://www2c.comm.eng.osaka-u.ac.jp/~prog2/>

# 使用言語＋開発環境

- Python(Python3)

- インタプリタ言語

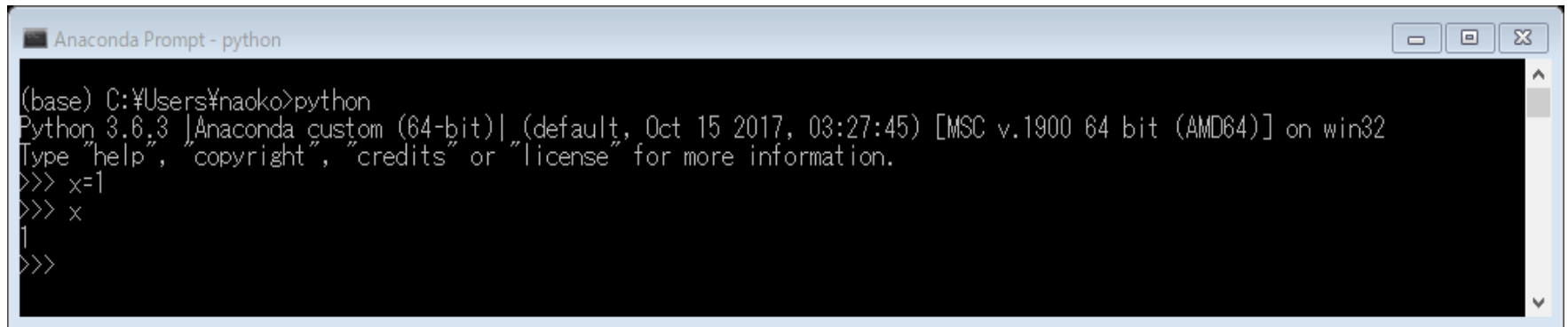
- (C、C++:コンパイラ言語)

- Anacondaディストリビューション

- (<https://www.anaconda.com/>)

# Anaconda Prompt

## ■ インタラクティブシェル

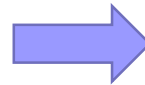


```
Anaconda Prompt - python
(base) C:\Users\Ynaoko>python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> x=1
>>> x
1
>>>
```

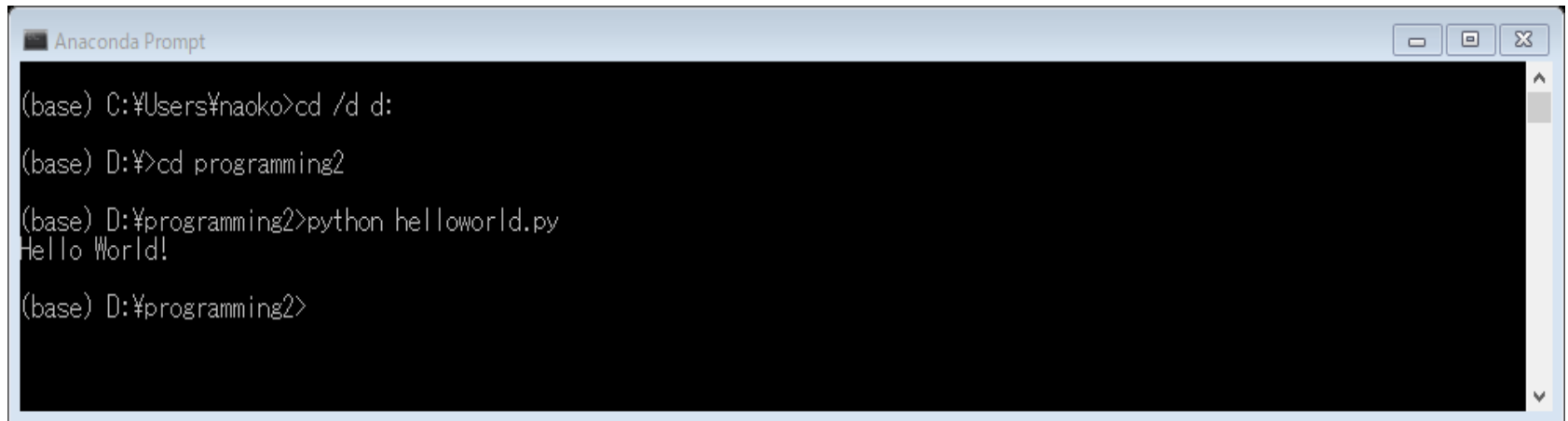
# Anaconda Prompt

## ■ ソースコードを実行

```
print("Hello World!")
```



helloworld.pyとして保存



```
Anaconda Prompt
(base) C:\Users\ynaoko>cd /d d:
(base) D:\>cd programming2
(base) D:\programming2>python helloworld.py
Hello World!
(base) D:\programming2>
```

# Spyder

(Promptから起動)

新しいコンソールを開ける

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Jan 22 12:47:30 2019
4
5 @author: naoko
6
7
8 print("Hello World!")
```

Annotations and their corresponding elements:

- A red arrow points to the "コンソール(O)" menu item in the top toolbar.
- A red box highlights the "変数エクスペローラー" (Variable Explorer) window, which displays a table with the following data:

名前	型	サイズ	値
x	int	1	1

**変数エクスペローラー**

- A red arrow points to the "変数エクスペローラー" tab in the bottom-right pane.
- A red box highlights the "IPythonコンソール" (Interactive Shell) window, which displays the following text:

```
Python 3.6.3 [Anaconda custom (64-bit)] (default, Oct 15 2017, 03:27:45) [MSC
v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]: x=1
In [2]:
```

**インタラクティブシェル  
(コンソール)**

**変数エクスペローラータブ**

# Spyder

実行

現在のディレクトリ  
(ソースコードを実行すると自動で移動)

The screenshot shows the Spyder Python IDE interface. The main editor window on the left contains a Python script named `helloworld.py` with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Jan 22 12:47:30 2019
4
5 @author: naoko
6 """
7
8 print("Hello World!")
```

The top toolbar contains various icons for file operations, editing, and execution. A red arrow points to the 'Run' icon (a green play button), with the label '実行' (Execute) next to it.

The top right of the interface shows the current directory path `D:\programming2` in a dropdown menu, which is highlighted by a red box. A red arrow points to this box with the label '現在のディレクトリ (ソースコードを実行すると自動で移動)' (Current directory (Automatically moves when source code is executed)).

Below the editor, the 'Variable Explorer' pane shows a single variable `x` of type `int` with a value of `1`. A red arrow points to this pane with the label '実行結果' (Execution result).

The bottom pane is the 'IPython Console', which displays the output of the executed code:

```
Python 3.6.3 [Anaconda custom (64-bit)] (default, Oct 15 2017, 03:27:45) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]: x=1

In [2]: runfile('D:/programming2/helloworld.py', wdir='D:/programming2')
Hello World!

In [3]:
```

A red arrow points to the 'Hello World!' output line in the IPython Console.

The bottom status bar shows the file encoding as UTF-8 and the current line and column numbers as 5 and 15, respectively.

**エディタ (ソースコードを書く)**

# Spyder

セルの実行

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script with the following code:

```
1 #-*- coding: utf-8 -*-
2 """
3 Created on Tue Jan 22 12:47:30 2019
4
5 @author: naoko
6 """
7
8 x = 1
9 print(x)
10
11 y=2
12 print(y)
```

A red arrow points to the line number 11, with the text "部分的にセルを選択して" (Partially select the cell).

On the right side, the "変数エクスプローラー" (Variable Explorer) window shows the current state of variables:

名前	型	サイズ	値
y	int	1	2

A red arrow points to the variable 'y', with the text "変数の状態" (State of the variable).

At the bottom, the "IPythonコンソール" (IPython Console) window shows the execution results:

```
In [1]: y=2
...: print(y)
2
In [2]:
```

A red arrow points to the output '2', with the text "セルの実行結果" (Execution result of the cell).

The bottom status bar shows: 権限: RW | 改行: CRLF | エンコード: UTF-8 | 行: 11 | 列: 1 | メモリ: 49 %

# Pythonの基礎

## ■ 出力

```
print("Hello World!")
```

## ■ 式

$i+j$ ,  $i-j$ ,  $i*j$ ,  $i/j$ ,  $i//j$ ,  $i\%j$ ,  $i**j$  (代数演算子)

$i==j$ ,  $i!=j$ ,  $i<j$ ,  $i>j$ ,  $i<=j$ ,  $i>=j$  (比較演算子)

$a$  and  $b$ ,  $a$  or  $b$ , not  $a$  (ブール演算子)



# Pythonの基礎

## ■ 代入 (変数の宣言は不要)

pi=3, pi=3.14159,

message= "test" ('test'でもよい)

flag= True

x, y = 2, 3 (複数の代入)

x += y, x -= y, x \*= y, x /= y, etc.

## ■ 型の確認

type(2) ⇒ int

type(2.0) ⇒ float

type("test") ⇒ str

# Pythonの基礎

## ■ 入力

```
x = input()
```

```
x = input("Please input a message: ")
```

## ■ 型の指定

```
int("1") : "1" (str) を 1 (int) へ変更
```

```
float("1") : "1" (str) を 1.0 (float) へ変更
```

```
int("test") ⇒ invalid literal for int() with base 10: 'test'
```

# Pythonの基礎

## ■ 構文の基本ルール(例: 条件分岐)

**# の後はコメント**

```
# 入力された整数値 (str) をintに変更してxに代入  
x = int(input("Please input an integer: "))
```

**文の最後にセミicolon(;) 不要**

```
if x%2==0:  
    print(x, " is an even number.")  
else:  
    print(x, " is an odd number.")
```

**コロンを入れる**

**必ずインデント**  
**Pythonは括弧{ }を使わない!**  
**(インデントでブロックを表す)**  
**if ... elif .... else ...**

※Pythonにswitchはない

# Pythonの基礎

## ■ 繰り返し

```
for i in range(10):  
    print(i)
```

```
for i in range(2, 10):  
    print(i)
```

```
for i in range(2, 10, 2):  
    print(i)
```

```
for i in range(10, 0, -2):  
    print(i)
```

range関数:

range([最初の数値,] 最後の数値[, 増加する量])

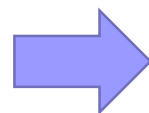
省略可  
デフォルト値: 0

省略可  
デフォルト値: 1

for (int i=start, i<end; i=i+step)



for n in range(start, end, step)



試してみよう！

# Pythonの基礎

## ■ 繰り返し

# 11と12で割り切れる最小の整数値を求める

x=1

while True:

if x%11==0 and x%12==0:

break

x += 1

print(x, " is divisible by 11 and 12.")

# Pythonの基礎

## ■ 関数

# xとyで大きい方を返す

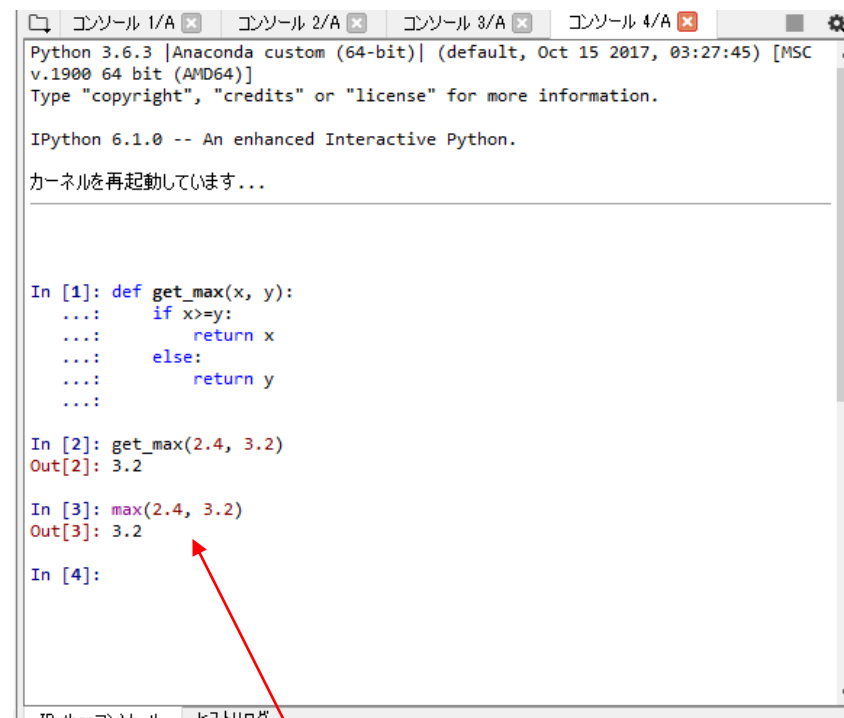
**def** get\_max(x, y):

if x>=y:

return x

else:

return y



```
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC
v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.
カーネルを再起動しています...

In [1]: def get_max(x, y):
...:     if x>=y:
...:         return x
...:     else:
...:         return y
...:

In [2]: get_max(2.4, 3.2)
Out[2]: 3.2

In [3]: max(2.4, 3.2)
Out[3]: 3.2

In [4]:
```

※ Pythonにはすでにmax()という関数がある

# 例：平方根を求める

```
a = int(input("Please input a positive integer: "))

x = 0.0
epsilon = 0.01
step = epsilon**2
numGuesses = 1

# x=0.0からx**2とaの誤差がepsilon未満となるxを
# step=0.0001ごとに総当たりで調べる
while abs(x**2-a)>=epsilon and x<a:
    x += step
    numGuesses += 1

print("numGuesses = ", numGuesses) # 調べた回数
if abs(x**2 - a) >= epsilon:
    print("Failed on square root of ", a)
else:
    print(x, " is close to square root of ", a)
```

# 例：平方根を求める

## ■ $a = 25$ の場合

numGuesses = 49990

4.9990000000001688 is close to square root of 25

## ■ 課題1-1: $a = 0.25$ の場合も答えが求められるように変更せよ

出力例:

numGuesses = 4899

0.489899999999996237 is close to square root of 0.25



# 例：平方根を求める

## ■ $a = 123456$ の場合

numGuesses = 3513631  
Failed on square root of 123456



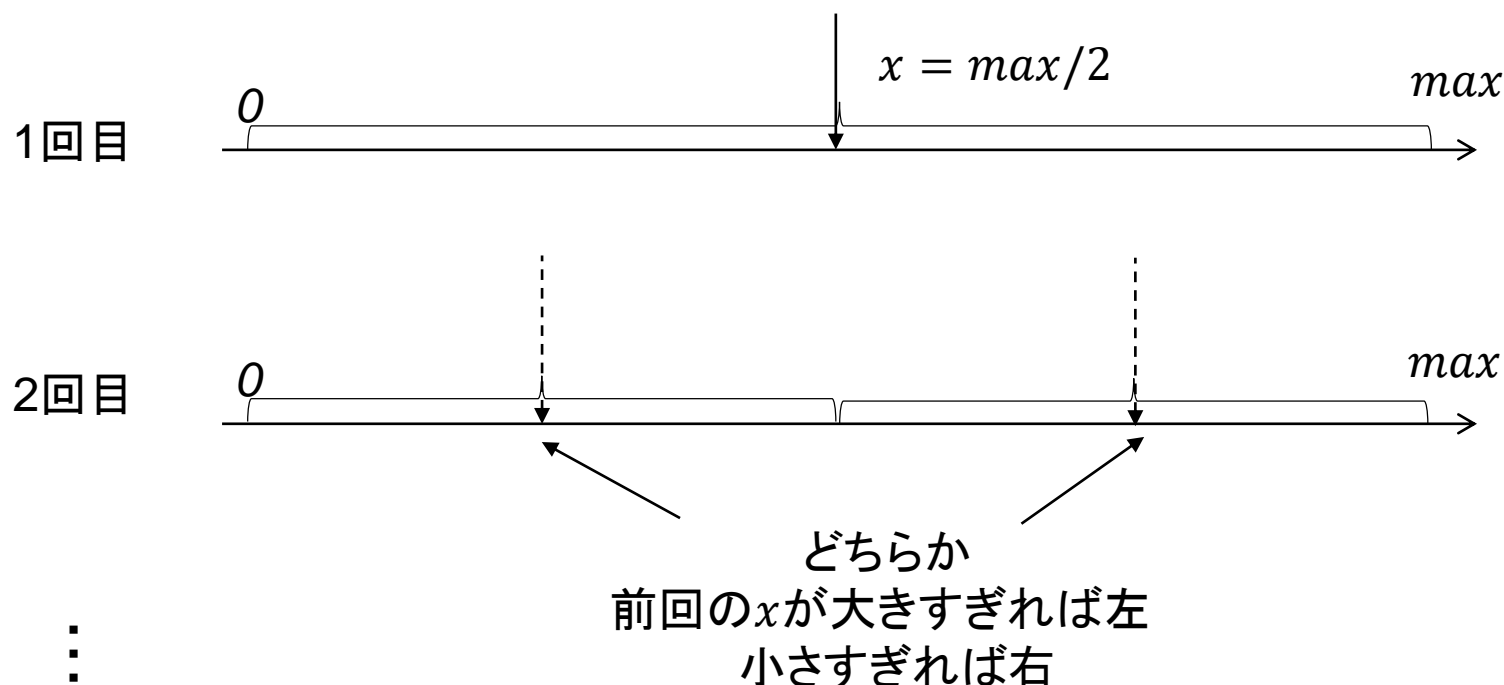
stepが大きすぎる！  
step = epsilon\*\*3にすると...

numGuesses = 351363047  
351.36304620491023 is close to square root of 123456

時間がかかる！

# 例：平方根を求める

- **課題1-2:** 下記のように2分法で探索するように変更し、調べた回数を変更前と比較せよ



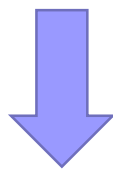
※ $a = 0.25$ のときも答えが求められることを確認せよ！

# 例：平方根を求める

- **課題1-3:** 下記のようにニュートン法(※)で探索する  
ように変更し、調べた回数を変更前と比較せよ。

$x^2 = a$ となる $x$ を求める

$\Rightarrow f(x) = x^2 - a = 0$  となる  $x$ を求める



$$x^{new} = x^{old} - \frac{f(x^{old})}{f'(x^{old})} \quad \text{で更新して求める}$$

# 例：平方根を求める

## モジュールを用いる例

```
import math    # モジュールを用いる

a = int(input("Please input an integer: "))
print(math.sqrt(a), " is the square root of ", a)
```

```
import numpy as np    # モジュールを用いる

a = int(input("Please input an integer: "))
print(np.sqrt(a), " is the square root of ", a)
```

# 自作のプログラムをモジュールに

squareroot.pyとして保存

```
def get_squareroot(a):
```

```
    x = 0.0
```

```
    epsilon = 0.01
```

```
    step = epsilon**2
```

関数にする

```
    while abs(x**2-a)>=epsilon and x<a:
```

```
        x += step
```

```
    if abs(x**2 - a) >= epsilon:
```

```
        print("Failed on square root of ", a)
```

```
    else:
```

```
        return x
```

import squareroot

で他のプログラムで利用できる  
(squareroot.get\_squareroot(x))

このプログラム  
をモジュールとして  
用いたときは無視される

```
if __name__ == "__main__":
```

```
    a = float(input("Please input a positive integer: "))
```

```
    print("The square root of", a, "is", get_squareroot(a))
```

※ モジュールは一度importしたらメモリ上に保持される。importした後のsquareroot.pyの変更を反映する場合はリロードが必要

```
from importlib import reload  
reload(squareroot)
```

# 例：平方根を求める

```
def get_squareroot(a):
```

```
    x = 0.0
```

```
    epsilon = 0.01
```

```
    step = epsilon**2
```

```
    while abs(x**2-a)>=epsilon and x<a:
```

```
        x += step
```

```
    if abs(x**2 - a) >= epsilon:
```

```
        print("Failed on square root of ", a)
```

```
    else:
```

```
        return x
```

```
def main():
```

```
    a = float(input("Please input a positive integer: "))
```

```
    print("The square root of", a, "is", get_squareroot(a))
```

```
if __name__ == "__main__":
```

```
    main()
```

こう書いてもいい



# 例：平方根を求める

- **課題1-4:** 課題1-2と1-3の考え方で平方根を求める関数をそれぞれ作成し、スライド22枚目のようにmain関数を用いて、動作を確認するプログラムを作成せよ。  
また、他のプログラムからこのプログラムをモジュールとして用いて、各関数の動作を確認せよ。

# レポートの提出

- 課題1-1～1-4に取り組む。
- 各課題に対し、プログラム作成時の考え方、ソースプログラム(コメント入り)、実行結果(適切なテストケースに対する動作確認)をレポートに記載する。
- レポートとソースコードを入れたフォルダを圧縮し、下記アドレスに提出する。

prog2@nanase.comm.eng.osaka-u.ac.jp

- 読みやすいレポートとするよう心がけること。
- メールのSubjectは「【Report1】学籍番号 氏名」、フォルダ名は「氏名」とする。
- 提出期限：4月25日(木)