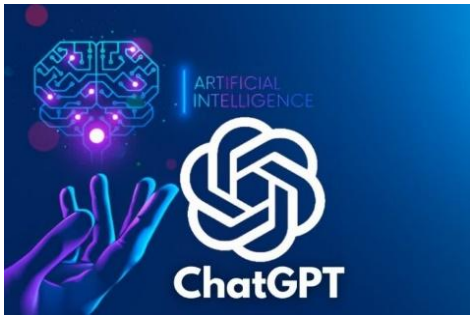
AIプログラミング

PYTHONによるEXCEL・WORD連携、AIプログラミング

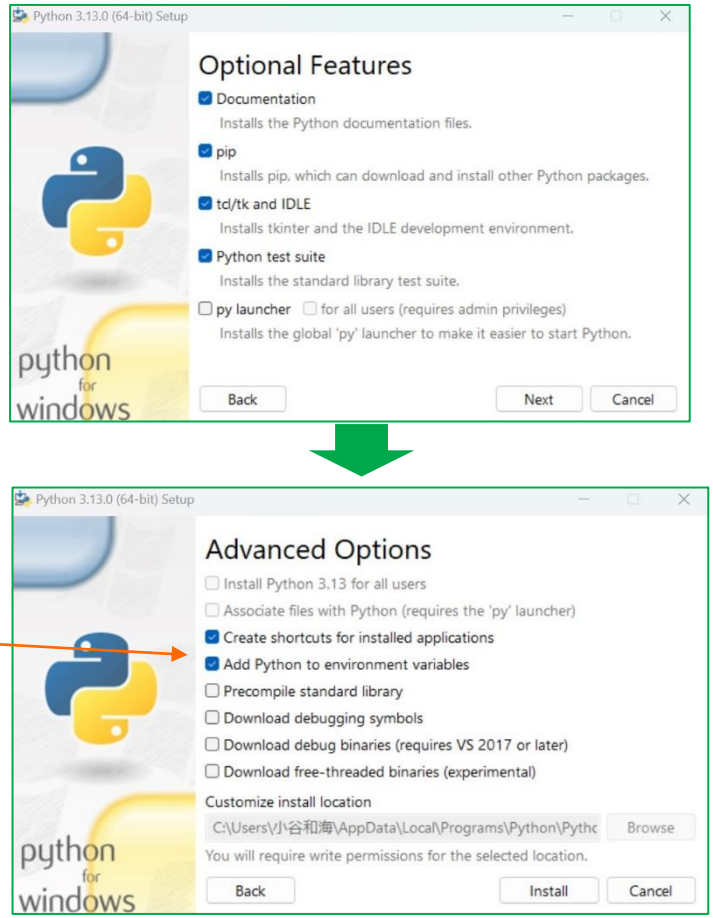
第二回



情報システムアナリスト
小谷　和海
ksys@kkotani.net

Windows
Pythonのインストール

- Pythonインストール
  - https://www.python.org/downloads/
  - Pythonの3系インストール
  - install時に Customize installation
    → Optional Features
    → Advanced Optionsの
    Add Python to environment variables
    をクリック



- openpyxlのインストール
  dos画面から以下を入力
  py -m pip install openpyxl
  py -m pip install python-docx
  py -m pip install openai

（Pythonの各VL毎にインストールが必要）

python™

WINDOWS

教材のダウンロード、実行

- 教材のダウンロード

  http://49.212.137.6/

  よりプログラムファイルをダウンロード

  zip形式なので解凍する。

  7zipがお薦め。

  https://sevenzip.osdn.jp/

- IDLEを起動してPython実行

  File→Openでプログラム指定

  プログラムを修正

  Run→Run module

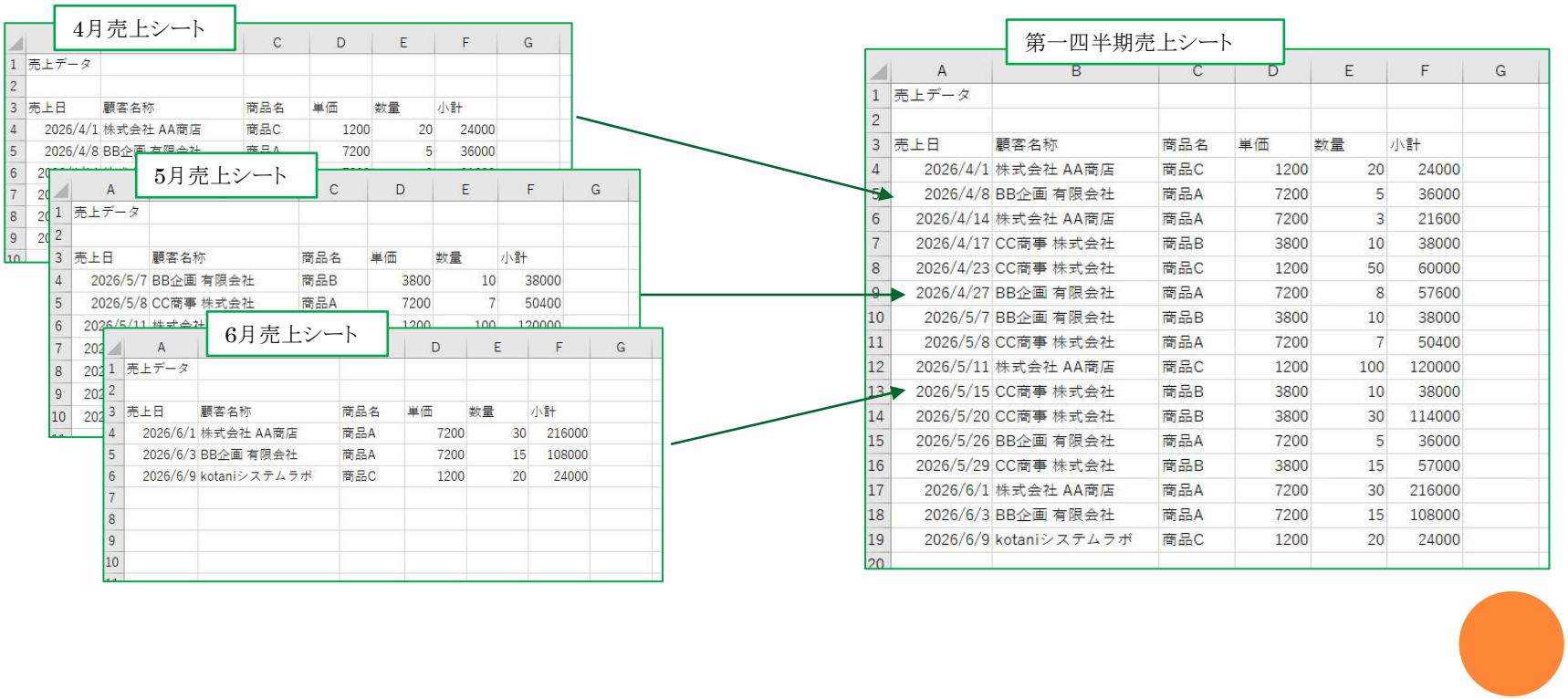
本日の内容

- Excel連携
  - ✓ 複数のシートをまとめる
  - ✓ 顧客マスタからデータを転記する
- word連携
  - ✓ 文書内のテキストを取り出す
  - ✓ wordテンプレートを活用する
  - ✓ EXCELとwordの連携
- AIプログラミングの実習

複数のシートをまとめる

XN10　複数のシートを一つにする

- dataフォルダーの売上データ_6月 入力ブックの中の4月 売上シート、5月 売上シート、6月 売上シートをまとめて第一四半期売上シートを新規に作成する。
- 第一四半期売上シートは先頭のシート位置に作成する。



4月 売上シート

	A	B	C	D	E	F	G
1	売上データ						
2							
3	売上日	顧客名称	商品名	単価	数量	小計	
4	2026/4/1	株式会社 AA商店	商品C	1200	20	24000	
5	2026/4/8	BB企画 有限会社	商品A	7200	5	36000	
6	2026/4/14	株式会社 AA商店	商品A	7200	3	21600	
7	2026/4/17	CC商事 株式会社	商品B	3800	10	38000	
8	2026/4/23	CC商事 株式会社	商品C	1200	50	60000	
9	2026/4/27	BB企画 有限会社	商品A	7200	8	57600	

5月 売上シート

	A	B	C	D	E	F	G
1	売上データ						
2							
3	売上日	顧客名称	商品名	単価	数量	小計	
4	2026/5/7	BB企画 有限会社	商品B	3800	10	38000	
5	2026/5/8	CC商事 株式会社	商品A	7200	7	50400	

6月 売上シート

	A	B	C	D	E	F	G
1	売上データ						
2							
3	売上日	顧客名称	商品名	単価	数量	小計	
4	2026/6/1	株式会社 AA商店	商品A	7200	30	216000	
5	2026/6/3	BB企画 有限会社	商品A	7200	15	108000	
6	2026/6/9	kotaniシステムラボ	商品C	1200	20	24000	

第一四半期売上シート

	A	B	C	D	E	F	G
1	売上データ						
2							
3	売上日	顧客名称	商品名	単価	数量	小計	
4	2026/4/1	株式会社 AA商店	商品C	1200	20	24000	
5	2026/4/8	BB企画 有限会社	商品A	7200	5	36000	
6	2026/4/14	株式会社 AA商店	商品A	7200	3	21600	
7	2026/4/17	CC商事 株式会社	商品B	3800	10	38000	
8	2026/4/23	CC商事 株式会社	商品C	1200	50	60000	
9	2026/4/27	BB企画 有限会社	商品A	7200	8	57600	
10	2026/5/7	BB企画 有限会社	商品B	3800	10	38000	
11	2026/5/8	CC商事 株式会社	商品A	7200	7	50400	
12	2026/5/11	株式会社 AA商店	商品C	1200	100	120000	
13	2026/5/15	CC商事 株式会社	商品B	3800	10	38000	
14	2026/5/20	CC商事 株式会社	商品B	3800	30	114000	
15	2026/5/26	BB企画 有限会社	商品A	7200	5	36000	
16	2026/5/29	CC商事 株式会社	商品B	3800	15	57000	
17	2026/6/1	株式会社 AA商店	商品A	7200	30	216000	
18	2026/6/3	BB企画 有限会社	商品A	7200	15	108000	
19	2026/6/9	kotaniシステムラボ	商品C	1200	20	24000	
20							

複数のシートをまとめる
XN10 複数のシートを一つにする

```
import openpyxl
import datetime
from pathlib import Path

parent = Path(__file__).resolve().parent
wb = openpyxl.load_workbook(parent.joinpath('data/売上データ_6月入力.xlsx'))
first_sheet = True
row_list = []

for ws in wb.worksheets:
    if first_sheet:
        start_row = 1
        first_sheet = False
    else:
        start_row = 4

    for row in ws.iter_rows(min_row=start_row):
        if row[0].row > 3 and row[0].value is None:
            break
        value_list = []
        for c in row:
            value_list.append(c.value)
        row_list.append(value_list)

ws_new = wb.create_sheet(title = '第一四半期売上')
num = 1
for row in row_list:
    ws_new.append(row)
    if num > 3:
        ws_new.cell(num,1).number_format = 'yyyy/m/d'
        ws_new.cell(num,6).value = '=D' + str(num) + '*E' + str(num)
    num += 1

top = 1 - len(wb.worksheets)
wb.move_sheet(ws_new, top)

wb.save(parent.joinpath('XN10_第一四半期売上.xlsx'))
```

dataフォルダーから売上データ_6月入力読み込み

空のリストを作成

・最初のシートはヘッダーを含め1行目から読む
・2番目のシートはヘッダーを除外して4行目から読む
・row_listに順次値を格納

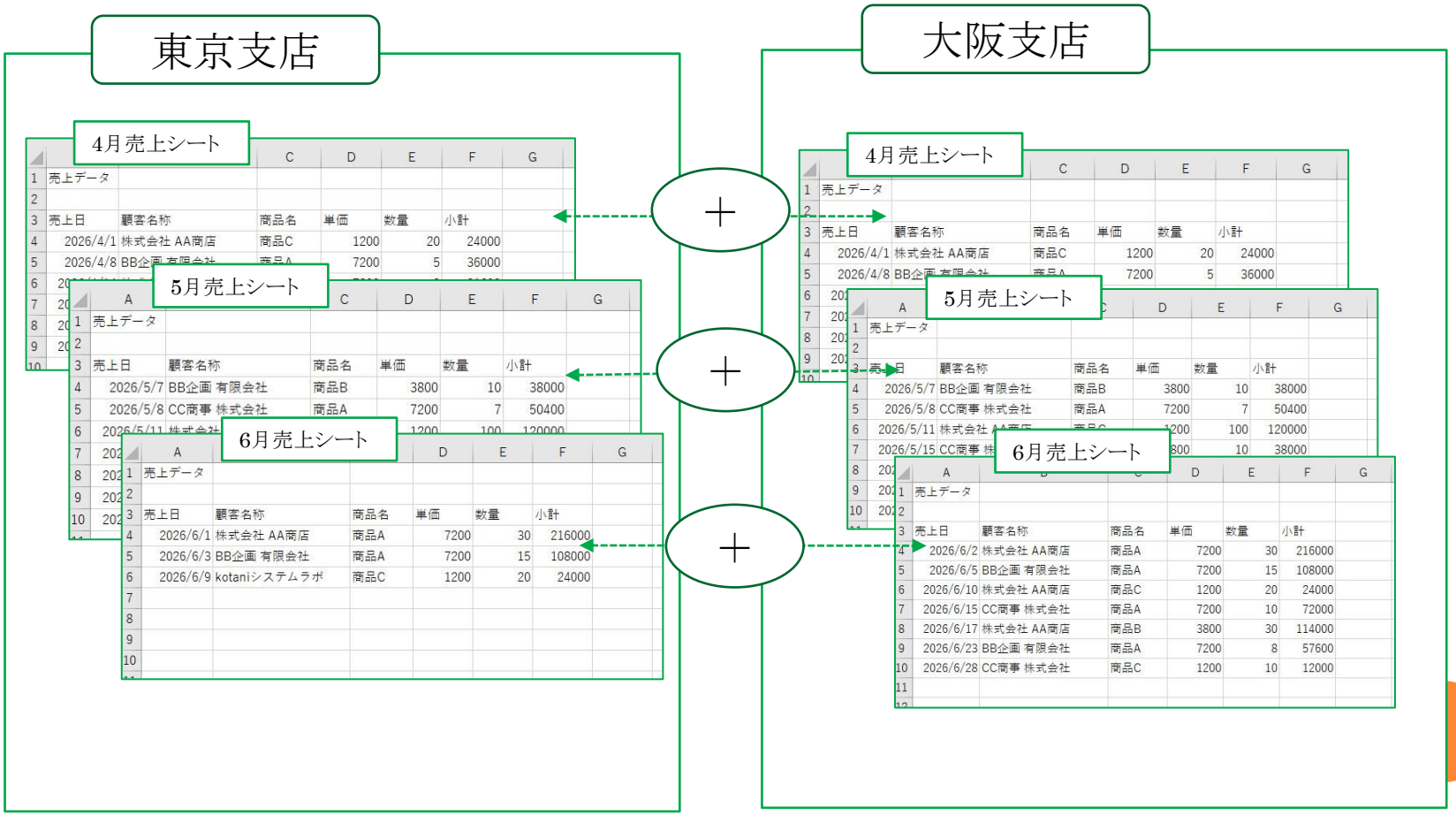
第一四半期売上シートを末尾に作成

・第一四半期売上シートにrow_listの内容を順次格納
・日付の表示形式設定、計算式設定

・先頭シート位置の移動量を計算し、先頭シート位置に移動
・新規ブック名を指定して格納

複数のシートをまとめる
XN11　複数のブックをシート名ごとにまとめる

- data/売上全支店フォルダーにある複数のブックを同名シートごとにまとめて、新しいブックに保存する。



複数のシートをまとめる
XN11 複数のブックをシート名ごとにまとめる

```
import openpyxl
from pathlib import Path

parent = Path(__file__).resolve().parent
wb_list = []

for file in Path(parent.joinpath("data/売上全支店")).glob("*.xlsx"):
    wb = openpyxl.load_workbook(file)
    wb_list.append(wb)
print(wb_list)
wb_new = openpyxl.Workbook()

sheet_names = wb_list[0].sheetnames
for sheet_name in sheet_names:
    ws_new = wb_new.create_sheet(sheet_name)
    is_first_book = True
    row_list = []

    for wb in wb_list:
        ws = wb[sheet_name]
        if is_first_book:
            start_row = 1
            is_first_book = False
        else:
            start_row = 4

        for row in ws.iter_rows(min_row=start_row):
            if row[0].row > 3 and row[0].value is None:
                break
            value_list = []
            for c in row:
                value_list.append(c.value)
            row_list.append(value_list)

    row_num = 1
    for row in row_list:
        ws_new.append(row)
        if row_num > 3:
            ws_new.cell(row_num, 1).number_format = "yyyy/m/d"
            ws_new.cell(row_num, 6).value = "=D" + str(row_num) + "*E" + str(row_num)
        row_num += 1
ws_first = wb_new.worksheets[0]
wb_new.remove(ws_first)
wb_new.save(parent.joinpath("XN11_売上データ全国.xlsx"))
```

data/売上全支店フォルダー内のexcelブック取得

最初のブックからシート名取得

格納用の新規シート作成

各ブック名の同一シート名のデータをrow_listに読み込む

row_list を順次読んで書式設定を行い書き込む

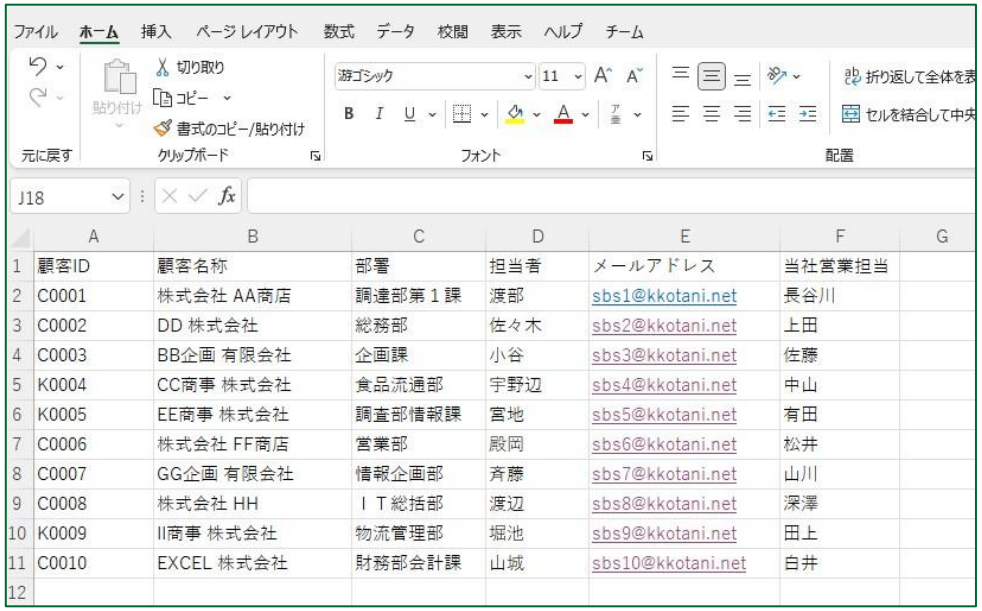
不要シートを削除して保存

python™

EXCELデータをデータベースのように扱う

XN15 顧客ごとの請求書作成

- dataフォルダーに顧客マスタEXCELがあります。
- 顧客マスタを使用して顧客ごとの請求書を作成する。



	A	B	C	D	E	F	G
1	顧客ID	顧客名称	部署	担当者	メールアドレス	当社営業担当	
2	C0001	株式会社 AA商店	調達部第１課	渡部	sbs1@kkotani.net	長谷川	
3	C0002	DD 株式会社	総務部	佐々木	sbs2@kkotani.net	上田	
4	C0003	BB企画 有限会社	企画課	小谷	sbs3@kkotani.net	佐藤	
5	K0004	CC商事 株式会社	食品流通部	宇野辺	sbs4@kkotani.net	中山	
6	K0005	EE商事 株式会社	調査部情報課	宮地	sbs5@kkotani.net	有田	
7	C0006	株式会社 FF商店	営業部	殿岡	sbs6@kkotani.net	松井	
8	C0007	GG企画 有限会社	情報企画部	斉藤	sbs7@kkotani.net	山川	
9	C0008	株式会社 HH	ＩＴ総括部	渡辺	sbs8@kkotani.net	深澤	
10	K0009	II商事 株式会社	物流管理部	堀池	sbs9@kkotani.net	田上	
11	C0010	EXCEL 株式会社	財務部会計課	山城	sbs10@kkotani.net	白井	
12							

python

EXCELデータをデータベースのように扱う

XN15 顧客ごとの請求書作成

- 顧客マスタを読み込んで顧客リストを作成する。

```
import openpyxl
import master
import datetime
from pathlib import Path

def get_master(master):
    wb = openpyxl.load_workbook(master)          #dataフォルダーから顧客マスタの読み込み
    ws = wb.active
    list = []                                    #空の顧客マスタリスト作成
    for row in ws.iter_rows(min_row=2):          #顧客マスタを2行目から全て読込む
        if row[0].value is None:                 #空行の場合、forループを抜ける
            break
        value_list = []
        for c in row:                            #該当行の顧客データを読込む
            value_list.append(c.value)
        list.append(value_list)                  #リストに顧客データ追加
    return list

parent = Path(__file__).resolve().parent
customer_list = get_master(parent.joinpath("data/顧客マスタ.xlsx"))
print(customer_list)
```

顧客マスタを読み込むget_master関数

顧客マスタを指定してget_master関数を呼び出す

帳票作成

XN15 顧客ごとの請求書作成

- 売上データ_202604から、請求書ひな型を使用して顧客ごとの請求書を生成

売上データ

売上日	顧客ID	顧客名称	商品名	単価	数量	計	税率区分
2026/4/1	C0001	株式会社 AA商店	商品C	1200	20	24000	
2026/4/2	C0001	株式会社 AA商店	商品B	3800	15	57000	*
2026/4/5	K0004	CC商事 株式会社	商品A	7200	5	36000	
2026/4/8	C0003	BB企画 有限会社	商品C	1200	5	6000	
2026/4/12	K0004	CC商事 株式会社	商品A	7200	3	21600	
2026/4/14	C0003	BB企画 有限会社	商品A	7200	3	21600	
2026/4/17	K0004	CC商事 株式会社	商品B	3800	10	38000	*
2026/4/18	C0003	BB企画 有限会社	商品B	3800	10	38000	*
2026/4/22	C0001	株式会社 AA商店	商品C	1200	50	60000	
2026/4/23	K0004	CC商事					
2026/4/25	C0001	株式会					
2026/4/27	C0003	BB企画					
2026/4/29	C0003	BB企画					
2026/4/30	C0001	株式会					

kotaniシステムラボ
〒426-0034
静岡県藤枝市駅前3丁目4－4

下記の通りご請求申し上げます。

合計金額（税込）	¥0

日付	商品名	区分	単価	数量	金額

注）区分の※は軽減税率（8%）対象

小計（税率8%対象）	0
小計（税率10%対象）	0
消費税（税率8%対象）	0
消費税（税率10%対象）	0
合計	0

お振込み先	連絡欄
静岡銀行　藤枝駅前支店（店番号：〇〇）	
（普通）〇〇〇〇〇〇	

ご請求書

ご請求日　2026年4月30日

株式会社 AA商店　御中

kotaniシステムラボ
〒426-0034
静岡県藤枝市駅前3丁目4－4

下記の通りご請求申し上げます。

合計金額（税込）	¥266,568

日付	商品名	区分	単価	数量	金額
2026/4/1	商品C		1,200	20	24,000
2026/4/2	商品B	※	3,800	15	57,000
2026/4/22	商品C		1,200	50	60,000
2026/4/25	商品A		7,200	8	57,600
2026/4/30	商品B	※	3,800	12	45,600

注）区分の※は軽減税率（8%）対象

小計（税率8%対象）	102,600
小計（税率10%対象）	141,600
消費税（税率8%対象）	8,208

python

帳票作成

XN15 顧客ごとの請求書作成 ～ 単数シート ～

```
    〔顧客マスタを読み込む関数参照〕
parent = Path(__file__).resolve().parent
customer_list = get_master(parent.joinpath("data/顧客マスタ.xlsx"))

wb_data = openpyxl.load_workbook(parent.joinpath("data/売上データ_202604.xlsx"), data_only=True)
ws_data = wb_data.active

wb_inv = openpyxl.load_workbook(parent.joinpath("data/請求書ひな型.xlsx"))
ws_inv = wb_inv.active

for customer in customer_list:
    customer_id = customer[0]
    customer_name = customer[1]
    data_list = []
    for row in ws_data.iter_rows(min_row=4):
        value_list = []
        for c in row:
            value_list.append(c.value)
        if value_list[1] == customer_id:
            data_list.append(value_list)
    if len(data_list) > 0:
        ws_new = wb_inv.copy_worksheet(ws_inv)
        ws_new.title = customer_id
        ws_new.cell(3,1).value = customer_name
        ws_new['H2'] = datetime.datetime(2026,4,30)
        row_num = 12
        for data in data_list:
            ws_new.cell(row_num,1).value = data[0]
            ws_new.cell(row_num,2).value = data[3]
            if data[7] is not None:
                ws_new.cell(row_num,5).value = '※'
            ws_new.cell(row_num,6).value = data[4]
            ws_new.cell(row_num,7).value = data[5]
            ws_new.cell(row_num,8).value = data[6]
            row_num += 1
wb_inv.remove(ws_inv)
wb_inv.save('XN15_請求書.xlsx')
```

・get_masterによりcustomer_listに顧客データ読込み

・処理対象の売上データを指定
・請求書ひな型データを指定

・顧客マスタのcustomer_idに一致するデータの取り出し

・一致するデータが存在する場合にひな型ブックに新規シートをcustomer_id名で作成
・顧客名と日付を設定

・取得したデータを順次新規シートに書き込む
・消費税が8%対象のマークを設定

・不要シートを削除し保存

XN16　WORD処理を学ぼう

- Python-docxは、Pythonでwordファイル（.docx形式）を作成、編集、保存が簡単に行えます。

- ターミナルから以下のpipコマンドを入力してインストール

  py -m pip install python-docx

【pipは、Pythonのパッケージを簡単にインストール、アップグレード、アンインストールするツール】

python

XN16_READDOC

WORD処理　　文書内のテキストを取り出す

- wordファイルを読込み、段落(paragraph)プロパティを使って内容を表示する。

```
import docx

# 既存Wordファイルを読み込む
doc = docx.Document('data/letter.docx')

# 段落を順次表示
for p in doc.paragraphs:
    print(p.text)
```

for文で段落(paragraph)を順次取り出す

XN17_WRITEDOC

WORD処理　　WORDテンプレートを活用する

- wordファイル(letter.docx)を開いて日付、宛名等を書き換える

```
import docx
import datetime
from pathlib import Path

parent = Path(__file__).resolve().parent
template_file = parent.joinpath('data/letter.docx')
save_file = parent.joinpath('XN17_Wrletter.docx')
now = datetime.datetime.now()

# 書き換える内容を指定
card = {
   '2026年4月1日': now.strftime('%Y年%m月%d日'),
   '●●●御中': 'システムソリューション御中',
   '■■■様': '小谷様',
   '★★★の送付について': '契約書の送付について'
}

# Wordファイルを開く
doc = docx.Document(template_file)

# 内容を書き換える
for p in doc.paragraphs:
    # テキストを書き換え
    for k,v in card.items():
        if p.text.find(k) >= 0:
            p.text = p.text.replace(k, v)

# Wordファイルへ保存
doc.save(save_file)
```

書き換える内容を辞書型データで指定する

辞書型データに沿って一つずつ置換処理を行う

XN18_WRITEDOC_FORMAT

WORD処理　　WORD書式を設定する

- 強調表示の書式設定を行う。

```
import docx
import datetime
from pathlib import Path

parent = Path(__file__).resolve().parent
template_file = parent.joinpath('data/letter.docx')
save_file = 'XN18_Wrletter.docx'
now = datetime.datetime.now()

# 書き換える内容を指定
card = {
  '2026年4月1日': now.strftime('%Y年%m月%d日'),
  '●●●御中': 'システムソリューション御中',
  '■■■様': '小谷様',
  '★★★の送付について': '契約書の送付について'
}
# どの部分に書式を設定するか
cstyle = {
  '★★★の送付について': True
}

# Wordファイルを開く
doc = docx.Document(template_file)

# 内容を書き換える
for p in doc.paragraphs:
    # テキストを書き換え
    for k,v in card.items():
        if p.text.find(k) >= 0:
            p.text = p.text.replace(k, v)
            # 書式を設定するか？
            if k in cstyle:
                font = p.runs[0].font
                font.bold = True
                font.underline = True
                font.size = docx.shared.Pt(20)

# Wordファイルへ保存
doc.save(save_file)
```

p.runs[0].fontプロパティにより太字、下線、文字サイズの変更を行う。

XN19_WRITEDOC_EXCEL　　EXCELとWORDの連携

- Excelの顧客名簿を元にwordの案内状を作成

	A	B	C
1	顧客名	郵便番号	住所
2	小谷　和海	427-0011	島田市東町2290-6
3	渡辺　一二三	427-0011	島田市東町１０－５
4	奥田　一二実	426-0022	藤枝市稲川４４４
5	登呂　遺跡	422-8037	静岡市駿河区下島３３３


```
import openpyxl as excel, docx, os
from pathlib import Path

parent = Path(__file__).resolve().parent
in_file = parent.joinpath('data/meibo.xlsx')
template_file = parent.joinpath('data/template-letter.docx')

# Excelファイルを読んでリストで得る
def read_book():
    result = []
    sheet = excel.load_workbook(in_file).active
    for row in sheet.iter_rows(min_row=2):
        v = [c.value for c in row]
        if v[0] is None: break
        result.append(v)
    return result
```

```
# 顧客の数だけ案内状を生成
for person in read_book():
    name, zipno, addr = person
    card = {
        '住所:▲▲▲': addr,
        '●●様へ': name+'様へ'
    }
    # Wordテンプレートを読む
    doc = docx.Document(template_file)
    # 内容を書き換える
    for p in doc.paragraphs:
        # テキストを書き換え
        for k,v in card.items():
            if p.text.find(k) >= 0:
                p.text = p.text.replace(k, v)
                p.runs[0].font.bold = True
    # Wordファイルへ保存
    save_file = 'XN21_' + name+'様.docx'
    print('save:', save_file)
    doc.save(save_file)
```

ChatGPTによるAIプログラミング

XN20　ChatGPTによるAIプログラミング

ChatGPTは、自然言語処理の技術を使って会話文章を解析し、ディープラーニングを使って文章を生成するシステム

1. 長所

- ✓ **自然言語処理**: 非常に流暢な文章生成が可能で、多くの場合、人間らしい自然な会話ができます。
- ✓ **応用範囲の広さ**: 文書作成、プログラミングのアシスト、アイデア出し、教育サポートなど、多様な用途で利用されています。

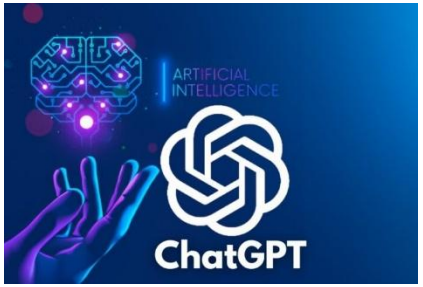
2. 短所・課題

- ✓ **誤った情報の生成**: 確信を持って誤りを提示することがあり、検証が重要です。
- ✓ **最新情報の制限**: 知識の更新に制限があり、最新の情報やイベントには対応できない場合がある。

⇒ **私が作成したプログラムは、この欠点を補う機能を持っている。**

ChatGPTを使ったAIによるプログラミング

日本語でプログラム生成の要求を行うとChatGPTがプログラムを自動生成します。



python™

ディープラーニングとは何か？
（深層学習）

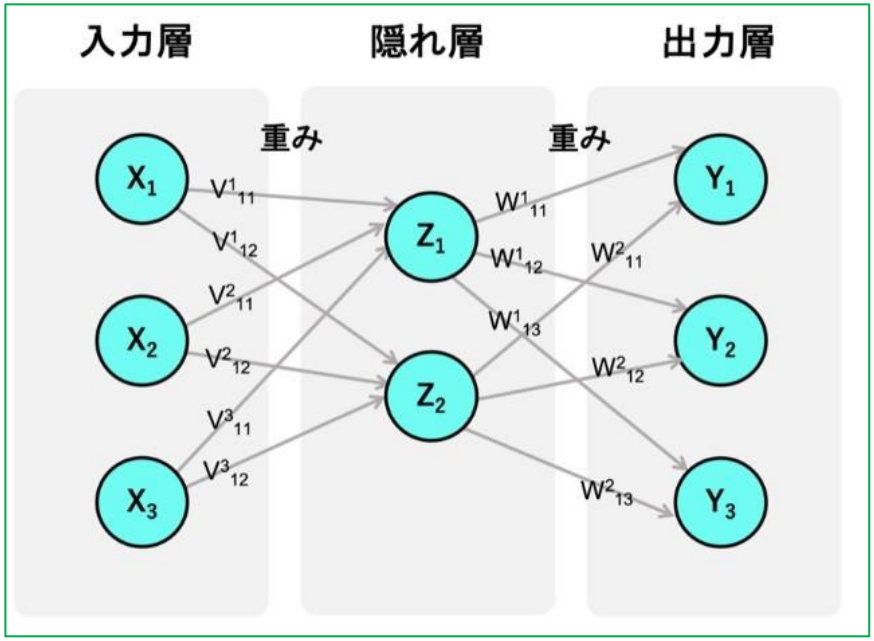
ニューラルネットワーク

ニューラルネットワークは、脳の神経回路から着想を得た数学モデル。
ニューラルネットワークは「層（Layer）」で構成されています。

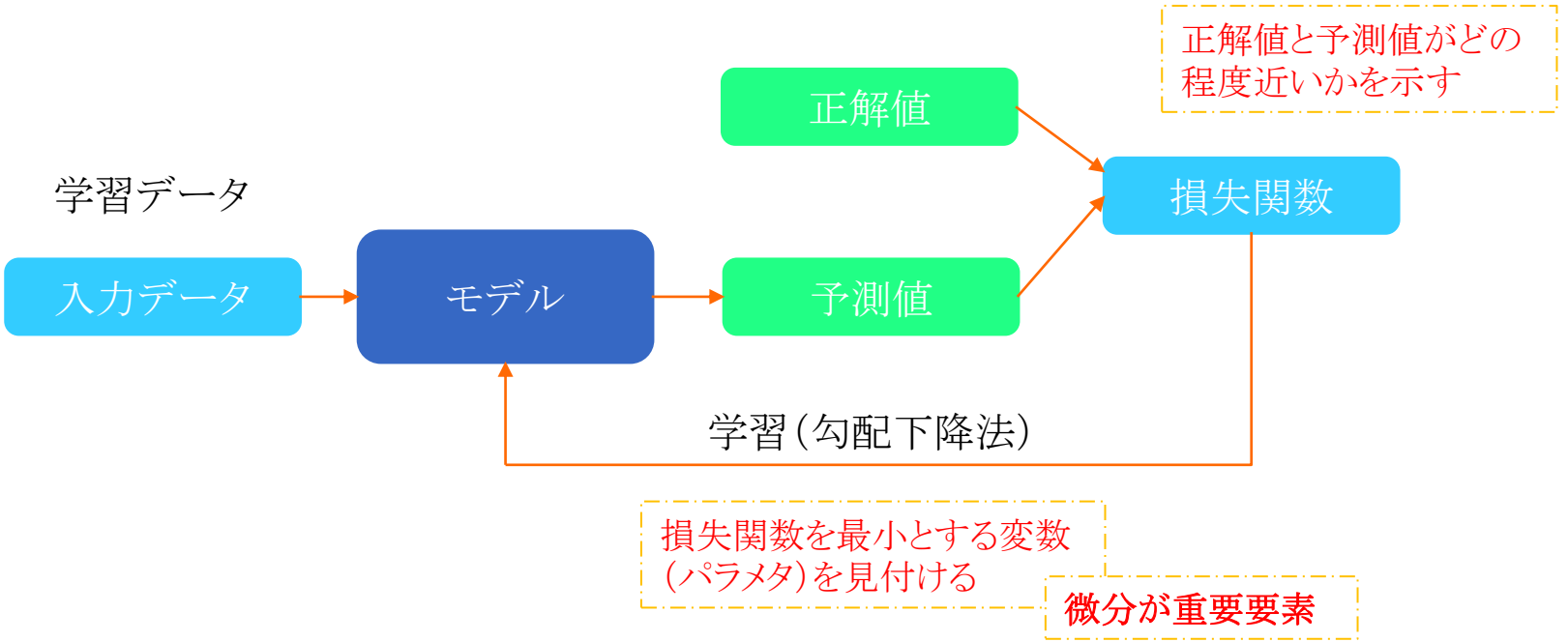
•入力層
　データが最初に入る場所です。たとえば、画像認識の場合、ピクセルの値が入力されます。

•隠れ層
　入力データを処理する場所です。ここで計算が行われ、データの特徴が抽出されます。
　複数の隠れ層があるものを「ディープニューラルネットワーク」と呼びます。

•出力層
　最終的な結果を出す部分です。
　たとえば、猫か犬かを判別するタスクでは「猫」「犬」の確率が出力されます。



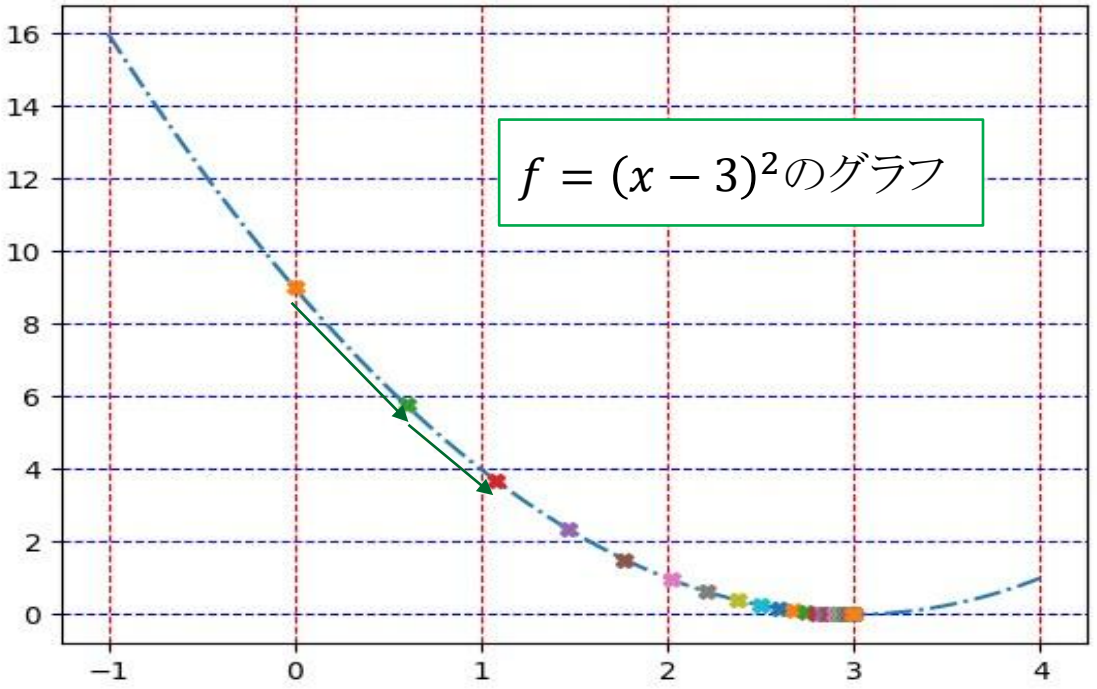
ニューラルネットワーク
学習の仕組



python™

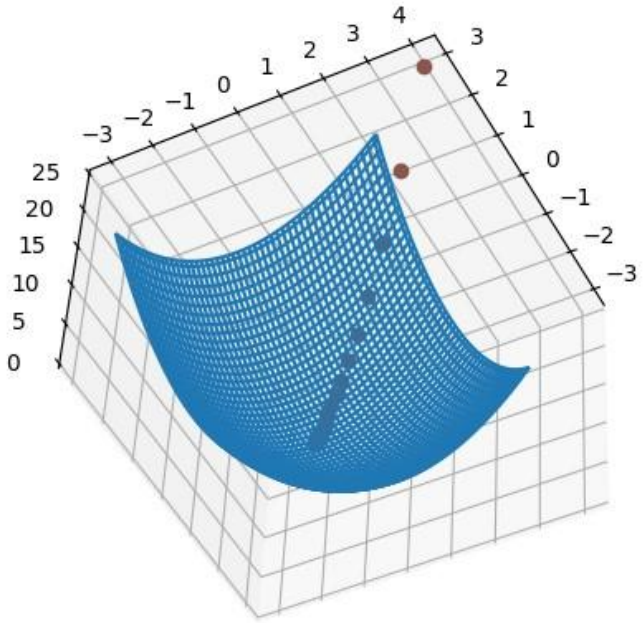
XN20_1　勾配下降法

- 勾配下降法（Gradient Descent）は、目的関数（損失関数）の最小値を求める方法
  - 関数の一番低いところ（最小値）を見つける方法
    - スタート地点: 開始地点を決める・
    - 傾きの計算: 今いる場所の傾きを微分により調べる。
    - 一歩進む: 傾きの急な方向（マイナス方向）に進む。
    - 繰り返す: これを何回も繰り返します。最終的に山の一番低い場所にたどり着くまで続けます。



X20_2　勾配下降法　曲面

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from matplotlib import animation

sp=3
x = y = np.linspace(-sp,sp,100)

fig = plt.figure()
ims = []
ax = fig.add_subplot(projection='3d')
X,Y = np.meshgrid(x,y)

z = X**2 + Y**2

ax.plot_wireframe(X,Y,z)

xl = []
yl = []
zl = []
xl.append(3)
yl.append(4)
zl.append(xl[0]**2 + yl[0]**2)
r = 0.1

for i in range(99):
    xl.append(xl[i] - r*2*xl[i])
    yl.append(yl[i] - r*2*yl[i])
    zl.append(xl[i+1]**2 + yl[i+1]**2)
    ims.append(ax.plot(xl,yl,zl,marker='o',linestyle='none'))

ani = animation.ArtistAnimation(fig, ims, interval=2000)

plt.show()
```



$z = x^2 + y^2$の関数
をワイヤフレームで描画

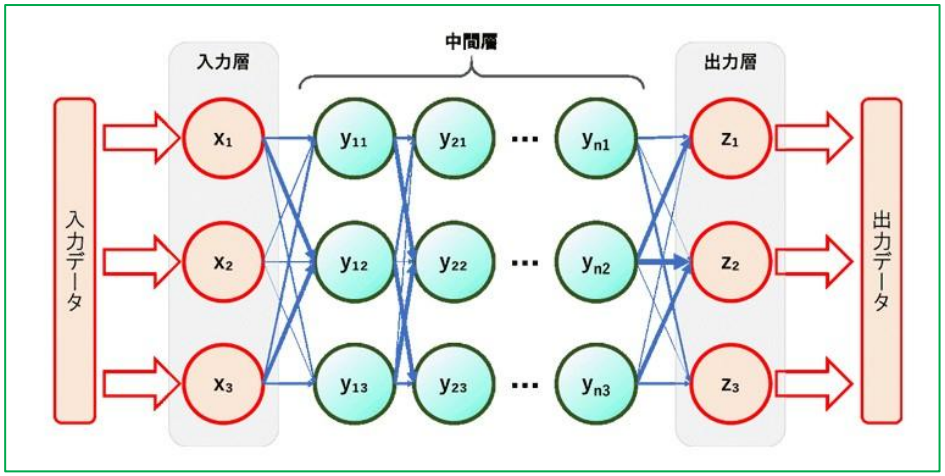
開始点　(3, 4, 25)　に●を描画

・偏微分値で次の点を求める
・次の座標位置に●を描画

2秒ごとにアニメーション描画

X20　ディープラーニング（層が複数）が強力な理由

- 層が深いほど情報を段階的に処理できるようになる。
  （複雑な問題を解けるようになる）
  - 画像なら
    - 第1層：エッジ
    - 第2層：角、模様
    - 第3層：パーツ（目・鼻）
    - 第4層：顔全体第
    - 5層：人物の種類
  - 言語なら
    - 第1層：文字・単語
    - 第2層：文法的関係
    - 第3層：意味的関係
    - 第4層：文脈
    - 第5層：意図・推論



深さが増えるほど、より抽象度の高い特徴を自動で学習できる。

XN20　ChatGPTによるAIプログラミング

- プログラム生成の入力例を以下に示す。
  - 素数を生成するプログラムを提案して下さい。
  - さらに高速な素数生成プログラムを提案して下さい。
  - 数当てゲームを提案して下さい。
  - カウントダウンプログラムを生成して下さい。
  - 可愛い猫の画像を取得するプログラムを生成して下さい。
  - openpyxlを使ってExcelデータを読んでword文書に変換するプログラム。

    など

- ChatGPTが自動生成したpythonプログラムを以下の名前で格納。

  XN20_prg数字.py

ChatGPTによるAIプログラミング

- ✓ このプログラムは、ChatGPTへの情報を与え、その情報をもとに会話を進め、会話を記録する。
- ✓ プログラム生成の要求を行った場合、自動生成されたPythonプログラムを新規Pythonプログラムとして格納する。
- ✓ excelに記載された情報を読込み、ChatGPTへ与える情報の作成を行う。
- ✓ 会話の記録をword文書として作成し保存する。

```
import openai,os
import datetime
import json
import openpyxl
from docx import Document

client = openai.OpenAI(api_key = "ChatGPTのキーを指定する")
messages = []
userInput = None
dt = datetime.datetime.now()
filex = 'chatgpt.xlsx'
wb = openpyxl.load_workbook(filex)
wsr = wb['chatgpt']
msg = []
num = 1
for row in wsr.iter_rows(min_row=2):
  c0 = wsr['A' + str(num)]
  if c0.value == 'S':

    c1 = wsr['B' + str(num)]
    num +=1
    msg.append(c1.value)
    messages.append(
      {
      "role": "system",
      "content": c1.value
    }
    )
doc = Document()
doc.add_heading('AIプログラミング', 0)
msg = ""
```

- ✓ ChatGPTのキーを指定

- ✓ ChatGPTへの情報を記載されたexcelを指定する

	A	B
1	S	あなたはチャッピーと言う名のアシスタントです。
2	S	大谷翔平さんはドジャースに移籍しました。
3	S	大谷翔平さんは2023年にMLBのホームラン王になりました。
4	S	大谷翔平さんは2023年に2度目のMVPを受賞しました。
5	S	大谷翔平さんは2024年にメジャーリーグ史上初の「50本塁打-50盗塁」を達成。
6	S	大谷翔平さんは2024年に史上初の3度目の満票でのMVP受賞。
7	S	大谷翔平さんはロサンゼルス・ドジャース移籍1年目で、ワールドシリーズ制覇に大きく貢献。

- ✓ excelのchatgptシートから情報を読込む
- ✓ Sの文字の情報のみをChatGPTに、role systemとして情報を与える。

- ✓ ChatGPTとの会話を記録するword文書を定義
- ✓ タイトルヘッダーを設定

ChatGPTによるAIプログラミング

- ✓ ChatGPTとの会話内容を蓄積し、毎回会話全体を与えることにより討議した内容を踏まえた会話ができる。
- ✓ ChatGPTからPythonプログラム生成の返答が有った場合、新規pythonプログラムとして格納する。
- ✓ endの入力で終了し、会話内容をword文書に保存する。

```
pno = 1
while True:
  userInput = input("聞きたいことを入力してください> ")
  messages.append({
    "role": "user",
    "content": userInput
  })
  msg += '>' + userInput + "¥r¥n"
  # docファイルにユーザ入力を書き込み
  doc.add_paragraph('Q: ' + userInput)

  if userInput != "end":
    res = client.chat.completions.create(
      model = "gpt-4o-mini",
      messages = messages
    )
    response = res.choices[0].message.content
    print(response + "¥r¥n")
    msg += response + "¥r¥n"
    if response.find('```python') > 0:
        source = response
        start_marker = '```python'
        end_marker = '```'
        sindex = source.find(start_marker)
        sindex += len(start_marker)
        eindex = source.find(end_marker,sindex)
        source = source[sindex:eindex].strip()
        pname = 'XN2O_prg' + str(pno) + '.py'
        with open(pname, 'w') as file:
            file.write(source)
        pno += 1
    messages.append({
      "role": "assistant",
      "content": response
    })
    # docファイルにユーザ入力を書き込み
    doc.add_paragraph(response)
  else:
    break
fname = 'chatgpt' + dt.strftime('%Y%m%d%H%M%S') + '.docx'
doc.save('XN2O_'+fname)       #word文書の格納
```

- ✓ 質問内容の入力
- ✓ roleでuserを指定し、質問内容をリストに追加

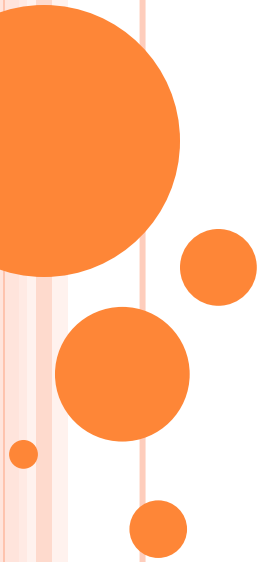
- ✓ endが入力されるまで継続
- ✓ modelでgpt-4o-miniを指定
- ✓ 質問をChatGPTに行う

- ✓ ChatGPTからの返答を印刷

- ✓ ChatGPTからのPthonプログラム生成から判断
- ✓ Pythonのプログラムを取出す
- ✓ プログラム番号を生成して新規Pythonプログラムとして格納

- ✓ chatgptの返答をassistantとしてリストに追加

- ✓ word文書の格納



終わり

経歴

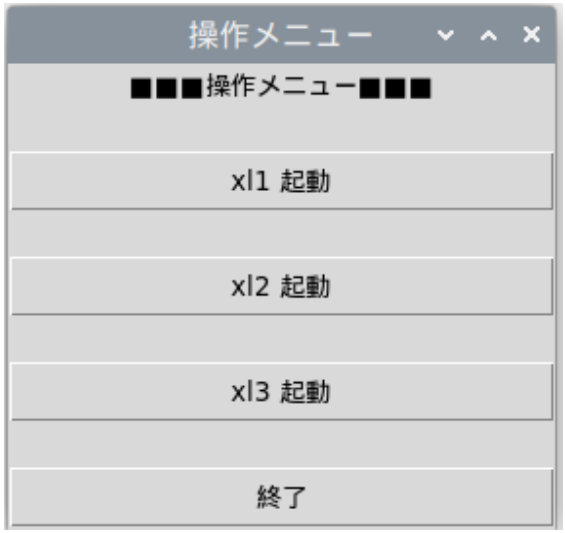
- 富士通で32年間ソフトウェア開発（大型OS、コンピュータグラフィックス、CAD等）を中心技術者として従事。
- 8年間トヨタ傘下の共和レザーで基幹情報システム開発、システム監査に従事。
- 2年間ソフトウェア会社（CSE)にて認証ソフト開発コンサル、その後言語変換業務開発コンサルに従事。
- 東海大学海洋学部非常勤講師(2001年～2006年）
- email
  - ksys@kkotani.net

付録

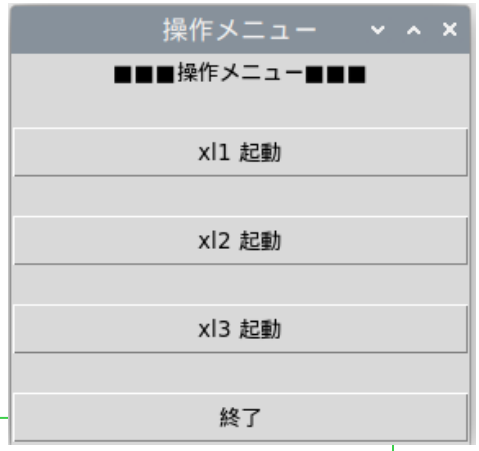
PYTHONの標準 GUI TKINTER

- Pythonプログラムの起動画面作成するのにどうするか？
  - Tkinter を使って構築する
    - Tkinterは、Python からGUIを構築・操作するための標準ライブラリ
    - 以下のウィジェットを組み合わせて画面を作る

      ラベル、 1行入力ボックス、 ボタン、 チェックボックス、 ラジオボタン、 リストボックス、 メニュー、 スクロールバー、 キャンバス など
- 以下の画面を作成



PYTHONの標準 GUI LAUNCHER PYTHON起動画面作成




```
import tkinter as tk
import sys
import subprocess
from pathlib import Path

parent = Path(__file__).resolve().parent
def prg_1():
    print('** program 1 is invoked.')
    path = parent.joinpath('XN4.py')
    subprocess.call('python %s' % path, shell=True)
def prg_2():
    print('** program 2 is invoked.')
    path = parent.joinpath('XN5.py')
    subprocess.call('python %s' % path, shell=True)
def prg_3():
    print('** program 3 is invoked.')
    path = parent.joinpath('XN6_1.py')
    subprocess.call('python %s' % path, shell=True)
def finish():
    sys.exit()
```

```
root = tk.Tk()
root.title('操作メニュー')
root.geometry('280x240')

labeltitle = tk.Label(root, text='■■■操作メニュー■■■')
labeltitle.pack()
label_blanc = tk.Label(root, text= '')
label_blanc.pack()
prg_1_button = tk.Button(root, text='XN4 起動', width=30)
prg_1_button['command'] = prg_1
prg_1_button.pack()

label_blanc = tk.Label(root, text= '')
label_blanc.pack()
prg_2_button = tk.Button(root, text='XN5 起動', width=30)
prg_2_button['command'] = prg_2
prg_2_button.pack()

label_blanc = tk.Label(root, text= '')
label_blanc.pack()
prg_3_button = tk.Button(root, text='XN6_1 起動', width=30)
prg_3_button['command'] = prg_3
prg_3_button.pack()

label_blanc = tk.Label(root, text= '')
label_blanc.pack()
finish_button = tk.Button(root, text='終了', width=30)
finish_button['command'] = finish
finish_button.pack()

root.mainloop()
```