プログラム自動生成支援による
「家族で学ぶパイソンプログラム入門」

株式会社ユーサイドシステム

第１章パイソンについて

１．はじめに

私は、１０年以上にわたってJavaプログラムの自動生成に取り組んできました。その背景にはシステム設計を上流、プログラム作業を下流と称し、プログラミングを軽視し非効率な人手に頼る開発体制での高コスト、その結果としての国際競争力低下への危機感があります。AI,IOT時代を迎え、益々進む産業の情報化に対してようやく先進的プログラム技術への重要性が認識され、２０２０年より小学校からのプログラミング教育という政策が打ち出されました。更に昨年より日本商工会議所ではプログラミング検定が始まり、今年よりパイソン言語検定も行われます。

ITに３０年以上関わりプログラム技術の重要性を認識し、自動生成技術による競争力向上を目指す者として、折角取り上げられた小学校からのプログラム教育をどうすれば効果あるものにできるか思案しました。結果AI,IOT言語としてまたインタプリタで学習言語として評価されるパイソン入門書を手に取り、素人の私がＩＴ業界に踏み込む契機になったシャープのポケコンBASIC言語との出会いを思い出しました。当時のBASIC言語は今のようなプログラム作りをはじめるのに多くの準備作業が要求されるのと違い、電卓同様コマンドを入力すれば即動いてくれました。素人の私がITに参入することが出来たのも、そうしたBASIC言語の環境があったからと言えます。しかしその後の経過は、コンパイル言語のCOBOL、インターネット出現によりｈｔｍｌJavascript,Perl,PHP等のインターネット言語習得が要求され、これでもかと思う技術的変遷による技術習得を強いられてきました。最後のJavaもコンパイル言語で甚だ厄介で、素人を遠ざける今のIT技術に疑問を抱いておりました。そうした折、小学校からのプログラ教育開始とパイソン言語との出会に、何か宿命を感じています。かねてより、プログラムの自動生成によってIT参入への高い垣根を取り払らわなくては、日本のIT分野の遅れを取り戻せないと思っておりました。

１０余年のJavaによる自動生成ですが、パイソン言語に出会ってそれも捨てパイソン言語の自動生成に取り組むことにしました。この入門編での無償公開自動生成プログラムは単体システムですが、現在更なるネットワークシステム、Webシステムの開発を進めています。自動生成技術とVPNネットによるテレワーク開発により、東京一極集中でのシステム開発を地方分散できれば、過度な残業依存からゆとりある働き方改革とIT人材の地方での活躍によって地方活性化が期待出来ます。

グローバルなIT技術競争の一方、働き方改革、人生１００年時代が叫ばれ、学校でのプログラム教育はもとより、これまでITに関わりを持たなかった方々のIT技術への関心が高まっています。今回掲げた「家族で学ぶパイソンプログラム入門」は、AI技術の進展により大きな社会的変革が予想され、これまでのようなIT技術者に閉じたITの世界をオープン化する必要があります。小学校でのプログラム教育に合わせて親子ともにプログラムを学ぶ契機とすべきです。その支援システムとして、これまで培ってきたプログラムの自動生成技術を活かすべきと思うに至りました。特に学校におけるプログラム教育の成否は、まず先生方がプログラムに興味を持ち、これからのAI時代を生きていかなくてはならない子供達に情熱を持ってプログラム教育を施すことが大切で、そうした先生の支援には保護者の理解が必要です。またこれからの教育は、全国レベルでの教材共有化、先生、保護者、生徒との連絡ネットワーク網構築等IT技術利用による教育のありようの大きな変革期を迎えます。

日本は少子高齢化、財政赤字、年金問題、更には温暖化と環境問題も深刻化していますが、問題先送りによる次世代への付け回しは目に余ります。老プログラマーの私に何ができるか甚だ心もとないのですが、敢えて花咲か爺さんの役を演じようと思います。誰もが身近な身の回りの課題をIT技術で解決しようとするチャレンジャー輩出こそ、日本停滞を打破し将来の希望へとつながります。そうしたチャレンジャーへのお役に立てれば、この上ない喜びです。

皆様のご支援、ご指導をお願いします。

令和２年５月吉日

株式会社ユーサイドシステム
代表取締役　佐久間　邦雄

２．パイソンの特徴

あらためて、パイソン言語の紹介をします。

①１９９１年オランダ人によって開発された歴史あるインタープリター言語

②欧米では高い言語シェアを誇り、Java とともに２大言語

③インタープリター言語はコンパイルの必要がなく、プログラムを即実行可

④データ処理に力点を置いてきた言語で、特に AI 開発では主流言語
　また、Web システム開発ライブラリー等の広範な開発ライブラリーが存在

⑤Java 同様サーバーサイド、クライアントサイドでの開発が可能
　アンドロイド携帯用パイソン版もあり

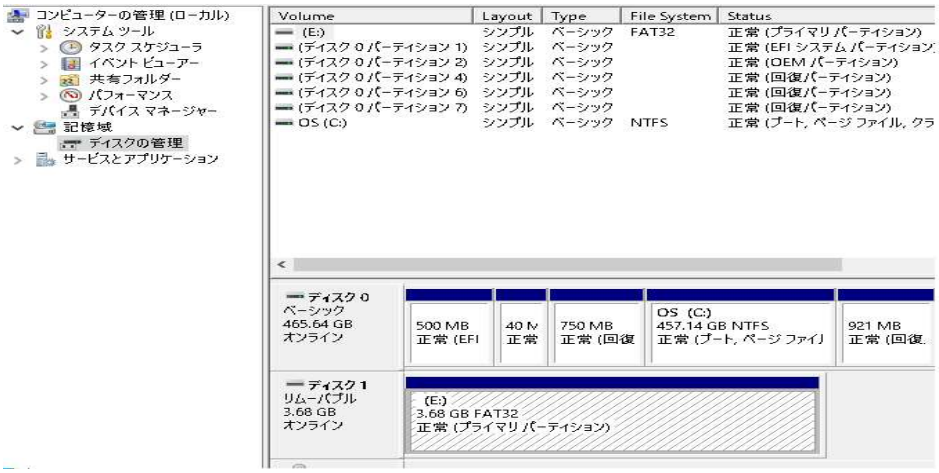
⑥ラズベリーパイと言われるボードコンピュータでも搭載可能
　ドローン、ロボット OS 等の広範な場面で利用

３．パイソン言語のインストール

絶えずパイソン言語及び各種ライブラリーはバージョンアップがなされます。
バージョンアップ情報には、注意を払う必要があります。
本来はパイソン言語のインストールを自ら行い、環境を用意する必要がありますが、ここでは弊社サイトより入門編環境すべてをダウンロードするか、ＵＳＢないし SD カードで提供された入門編環境を利用していただきます。
尚、ダウンロードする場合でも、ダウンロード先は USB ないし SD カードの外部リムーバルデスクを G で統一し、デスクの環境には影響のないようにします。また、外部リムーバルデスクに作成したパイソン入門編は自由にコピーし配布しても結構です。
外部リムーバルデスクの容量は 4GB あれば OK です。

（１）外部リムーバルデスクをＧに変更

Window 外部装置は各自異なっていますが、以下の操作でＧにします。
Windows 管理ツールのコンピュータの管理をクリックします。



上図例では外部リムーバルデスクはＥになっています。

リムーバルデスクＥにカーソルにあて、右クリックします。
更に、ドライブ文字とパスの変更をクリックします。

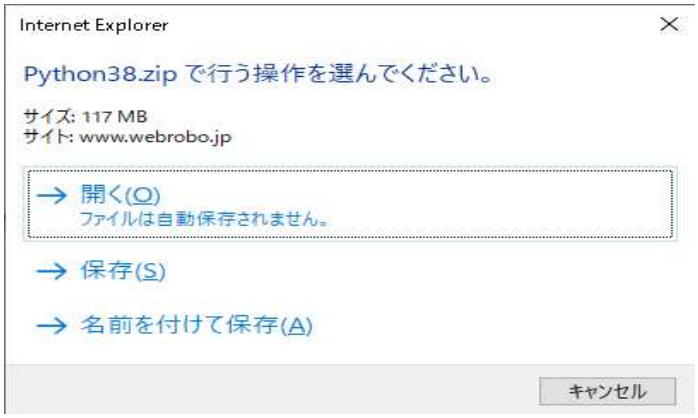
変更ボタンをクリックし、ドライブ文字をＧに変えてＯＫボタンをクリックします。

（２）パイソン言語のバージョン

パイソン言語には、２系と３系がありますが、３系を使用します。
パソコンＯＳは、Windows10で６４ビット用での使用とします。
パイソンのバージョンはPython3.8.2を入れています。

（３）入門編環境ダウンロード

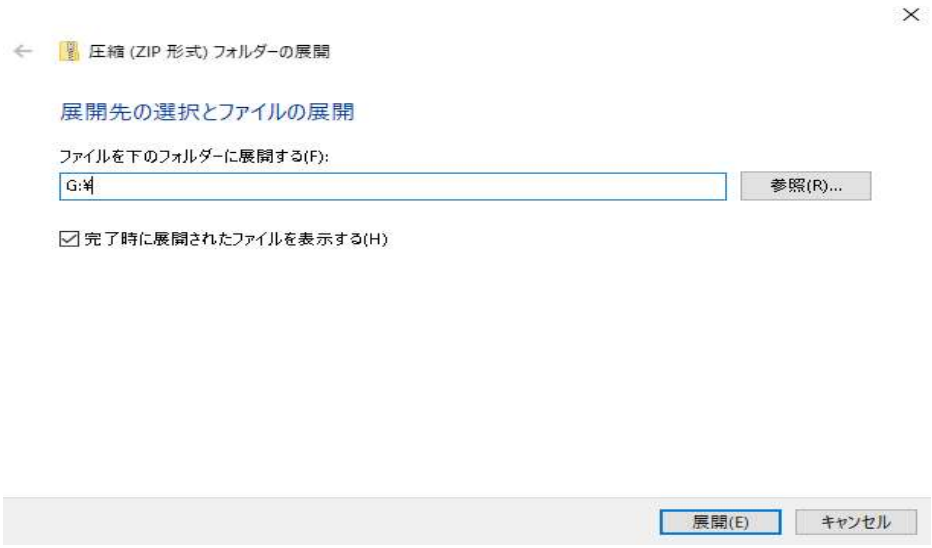
https://www.webrobo.jp/siryou/Python38.zip
をクリックします。下図画面が出てきますので



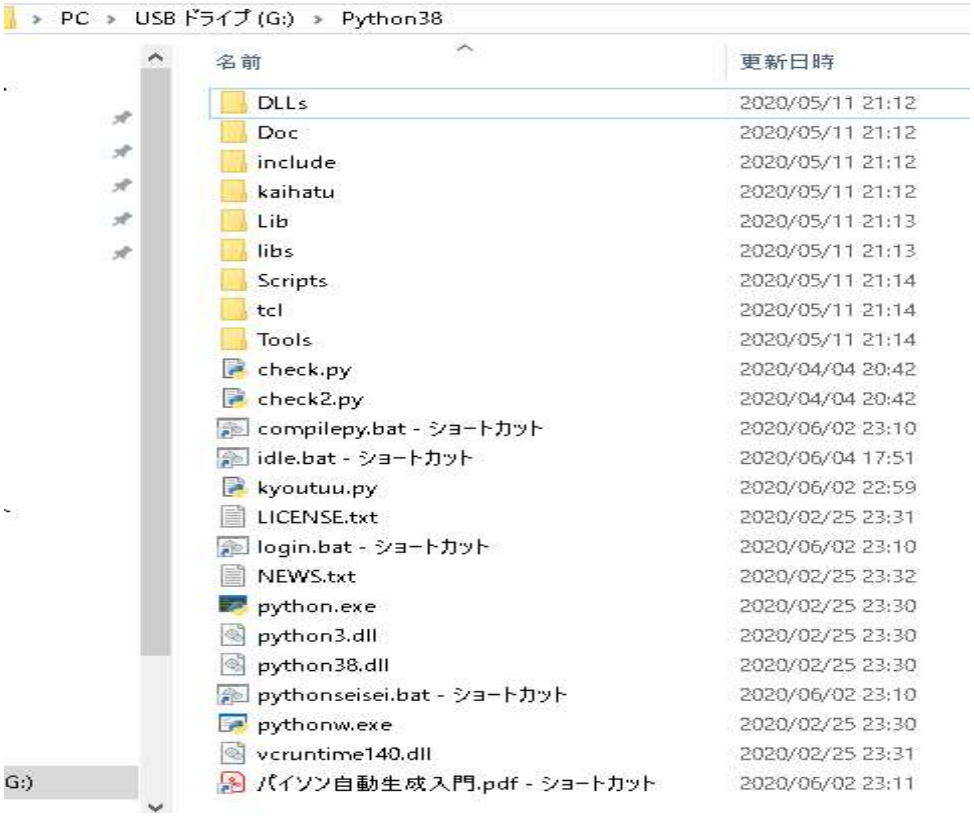
保存をクリックします。ダウンロードが終了しますと以下が表示されますのでフォルダーを開くをクリックします。

下図画面が表示されますので、Python38(1).zipカーソルを当て、右クリックをします。

すべて展開をクリックしますと下図画面がでますので、ダウンロードファイルの展開先をG:¥にして展開ボタンをクリックします。尚、Gにはフォーマットしたリムーバルディスクを挿入しておきます。



g:¥python38をクリックしますと以下が表示されます。

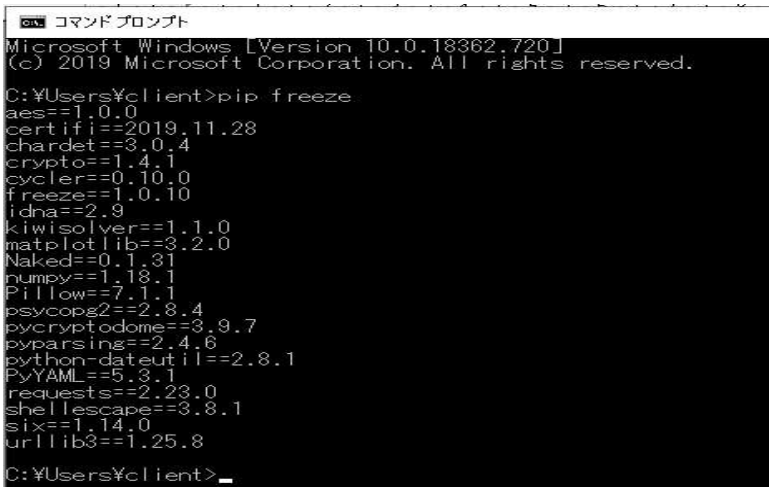


上図のフォルダーのidle.bat, login.bat, pythonseisei.bat, パイソン自動生成入門.ＰＤＦの４つのショートカットをデスクトップにコピーします。
上図ではいろいろなフォルダー、ファイルがありますが、特にkaihatuを使うことになります。２章では、kaihatu￥sampleフォルダーに入っているプログラムを使い、3章ではkaihatuに入っているプログラム等を使います。

（４）ｐｉｐコマンドについて

パイソンは極めて多くのパッケージが存在し、インターネットに接続した状態で
pip install パッケージ名で簡単に必要なパッケージをインストールしてくれます。

但し、注意はバージョンの整合性がとれない場合は、バージョンがあわないとのメッセージが出ます。その場合はバージョンをあわせて下さい。
pip install freeze と入力しますと、下図のようにインストールされているパッケージの一覧が表示されます。



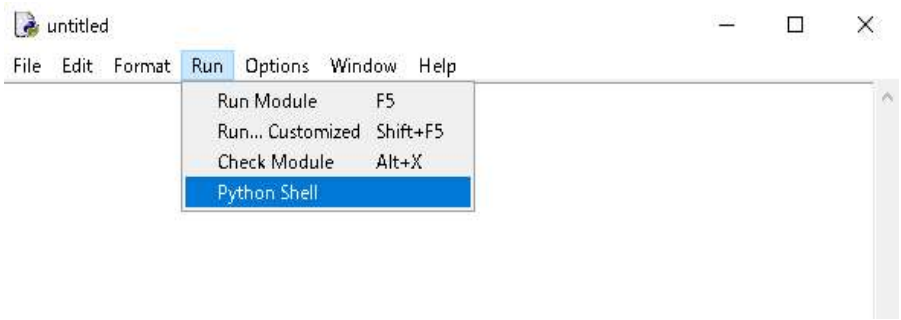
インストールしたパッケージの削除は、pip uninstall パッケージ名で削除します。
パイソンは本体のみならず、各種のパッケージもバージョンアップしています。
インストールがうまくいかない場合が出てきますので、インターネットでお調いただき対応をとって下さい。

第２章パイソンによるシンプルプログラミング

１．変数と計算

では、早速パイソン言語を動かしてみましょう。
先ほど作成したデスクトップにある IDLE アイコンを起動しますと下図の画面が表示されます。但し、下図画面が表示されるのに多少時間がかかりますのでお待ちください。
尚、IDLE の詳細な操作マニュアルは以下にあります。
https://docs.python.org/ja/3/library/idle.html

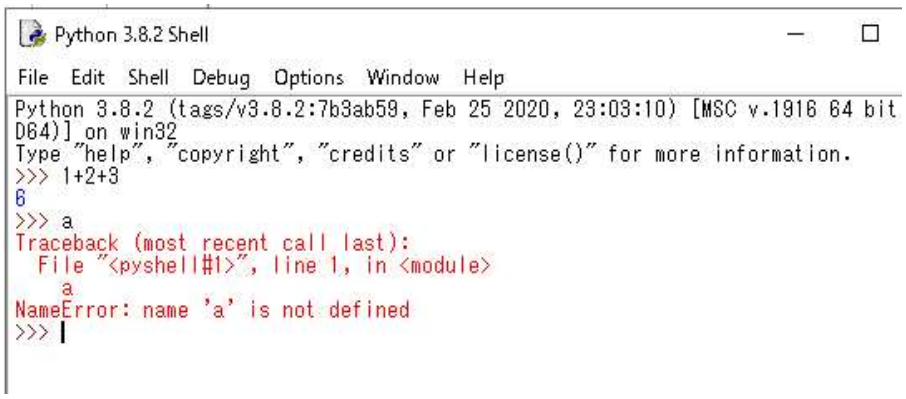
次に、Run の PythonShell をクリックします。



上図の>>>は、パイソンのコマンド待ち状態です。
早速電卓的操作を実行してみましょう。
上記画面で、1+2+3 と入力しエンターキーを押します。

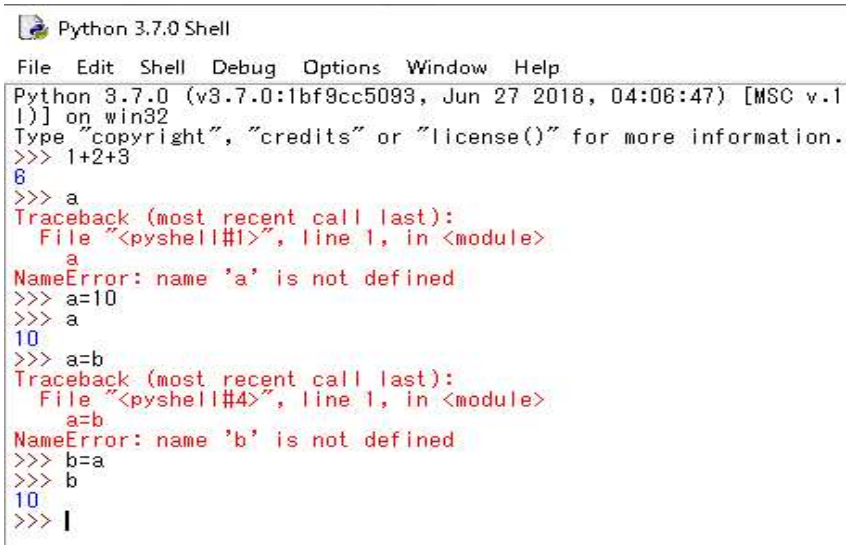
```
>>>1+2+3
6
>>>
```

計算結果６が表示されます。
同様に、いろいろ試してください。
次に、文字ａを入力します。



NameError：name a is not defined とエラー表示されました。
これは何もないａが指定されたということです。
さて、ここでプログラミングの基本である変数について説明します。
コンピューターは、よく人間の脳に例えられます。
脳は、目や耳などの外部よりの情報を記憶し、計算、判断をして外部身体にその反応を返す働きをします。コンピューターもキーボードの値を一旦記憶し、計算し、その結果を外部装置の画面に表示するということをします。
上の操作でａと入力しただけでは、ａは何なのか判断できずエラーを返しました。
脳は外部情報を記憶すると申しましたが、コンピュータも外部情報を記憶します。その記憶の入れ物が変数と言われるものです。
では、a=10 と入力してみます。エラーは表示されず、>>>が表示されコンピュータは指示待ちになりました。
次に、再度ａと入力してみます。
今度はエラーは出ずに、10 が表示され指示待ちになりました。
つまりは、a=10 はａという入れ物（変数）に 10 を入れるという命令です。
次のａの入力は、ａに入っている値を表示しなさいという命令で、その結果 10 が表示されました。
a=10 はコンピューターでは数学式のａと 10 は等しいという意味ではなく、ａという変

数に 10 を入れるということです。数学的イコールは、コンピュータープログラムでは a==10 という=が２個続く式で表します。
次に、a=b と入力してみてください。
また、NameError が出ましたね。では、逆に b=a と入力してみてください。
今度は、エラーは出ませんでした。では、b と入力してみます。
10 の値が表示され、エラーはでませんでした。
a=b では、何もない変数 b を a に入れるということでエラーになりました。
逆に b=a では、b に a の値を入れるということで、a には 10 が入っており、b にその値が入った結果、b を入力して 10 の値が表示されました。
これまでの操作は、既にコンピュータに命令をし、プログラミングをしていると言っても間違いではありません。実に簡単ですね。これがインタープリタ言語のいいところです。



変数はデータの入れ物で脳の記憶の役割をすると言いました。しからばいつまで記憶しているのでしょうか。パイソン言語を終了するか、パソコン電源を切れば記憶はなくなります。折角入力したのに消えてしまうのでは、何かに残しておきたいと思うでしょう。
その方法は後にお話しします。もう少しこのままで話を進めます。
次に、変数という入れ物にはいくつかの種類があります。
例えば、a に姓を、b に名前を入れたいとします。a=佐藤、次に b=一郎と入力してみましょう。結果は、NameError になります。
しからば、a と入力しますと 10 と表示され、a には先ほど入れた値が記憶されています。
変数には、数字の変数と文字の変数があり、一度数字で入力した変数には、文字が入らないのです。であれば、a=20 と入力してみましょう。エラーなく入ります。a は数値の変数になっているのです。
であれば別の変数 c と d に佐藤、一郎を入れてみましょう。
c=佐藤では NameError になってしまいます。文字変数の場合は、c='佐藤'のように文字の前後を'で閉じる必要があります。
c と d に姓と名前を入れて e=c+d と入力してみます。
e と入力すると、'佐藤一郎'と表示されます。
文字の場合の＋は文字をそのまま結合する命令になります。

```
>>> a=10
>>> a
10
>>> a=b
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    a=b
NameError: name 'b' is not defined
>>> b=a
>>> b
10
>>> a=佐藤
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    a=佐藤
NameError: name '佐藤' is not defined
>>> b=一郎
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    b=一郎
NameError: name '一郎' is not defined
>>> a
10
>>> a=20
>>> a
20
>>> c=佐藤
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    c=佐藤
NameError: name '佐藤' is not defined
>>> c='佐藤'
>>> d='一郎'
>>> e=c+d
>>> e
'佐藤一郎'
>>>
```

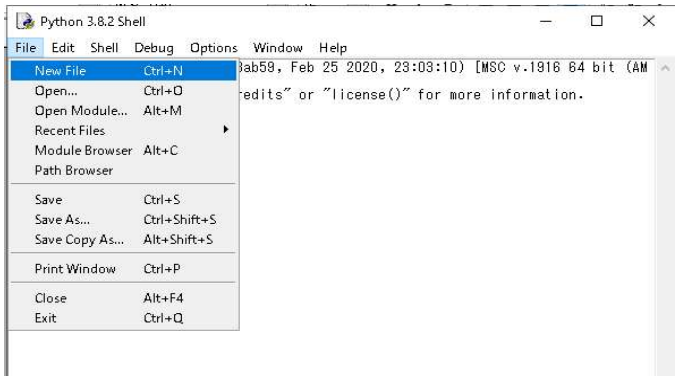
変数は、何も値を入れない場合は変数の型が決まらず、何らかの値を入れた時に変数の型が決まります。異なる型の値を入れた場合、エラーが出ます。
文字型になったｅにe+10と入力しますと、TypeError: can only concatenate str (not "int") to strという型が違うというエラーが出ます。
文字型になった変数ｅに文字としての10を結合する場合は、e+'10'と入力するか、e+str(10)のように10を文字化する式を使います。
ここで、ｅと入力しますと佐藤一郎のままですが、文字の10を結合してｅの変数に入れるのであれば、e=e+'10'もしくはe=e+str(10)と入力する必要があります。

```
>>> e+10
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    e+10
TypeError: can only concatenate str (not "int") to str
>>> e+'10'
'佐藤一郎10'
>>> e+str(10)
'佐藤一郎10'
>>> e=e+'10'
>>> e
'佐藤一郎10'
>>> e=e+str(10)
>>> e
'佐藤一郎1010'
>>>
```

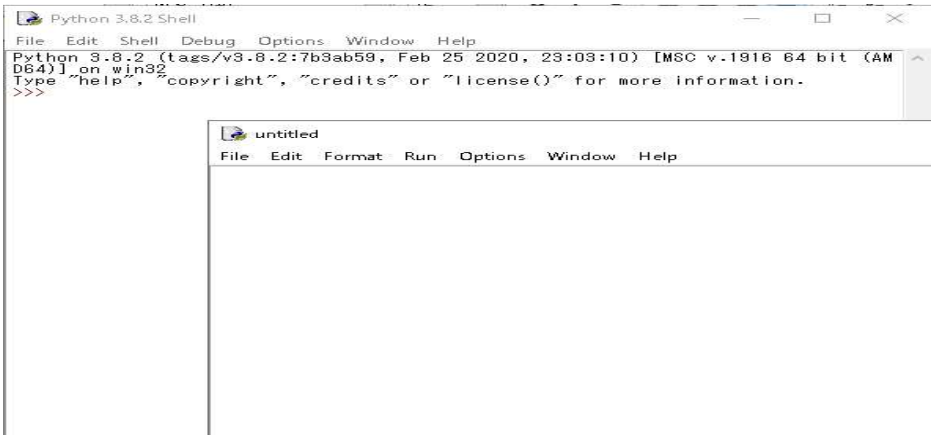
パイソンの変数には厳密には、整数型数値、ｆｌｏａｔ型数値、文字列型、ブール型があります。より詳しい説明は後に譲ります。

2. Input、Printコマンドによる四則計算

上記では、電卓的操作で四則計算が簡単にできました。しかし、1+2ないしa=10のように直接的命令を使ってきましたが、a=10ではａが要求する値が何かがわかりません。要求する値が何かを分かるようにプログラムしたいものです。この要求に応える命令がinputコマンドです。上記の変数をクリアするため、一度IDLEを終了して再度立ち上げます。
これからは、１行単位での入力ではなく、数行にわたるプログラム記述にします。
下図のようにIDLEで上段のFile のNew Fileをクリックします。



下図のように２画面になります。左の Python 3.8.2shell 画面は実行画面、右の untitled 画面は編集画面です。尚、Python 3.8.2shell 画面で New File を、あるいは untitled の編集画面でも New File をクリックしますと新たに untitled の編集画面が出ます。
あくまでも Python 3.8.2shell の実行画面と編集画面の２画面で操作下さい。
最初の Python 3.8.2shell 画面で New File ではなく、Open で既にあるプログラムを読み込んだ場合は、編集画面にそのプログラムが表示されます。
編集画面の Run の Run Module で Python 3.8.2shell の実行画面に移動してプログラムが実行されます。
両画面の Window をクリックしますと、開いている実行画面、編集画面がでますので画面の切り替えが出来ます。
この辺の操作は、いろいろいじってご理解いただく必要があります。



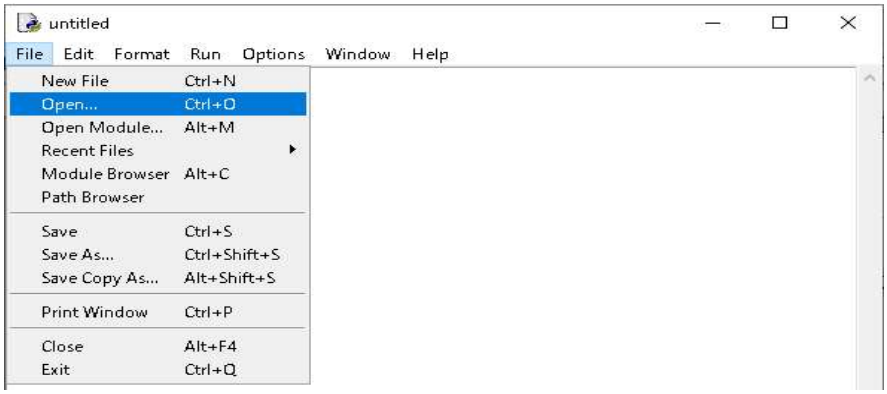
（1）足し算

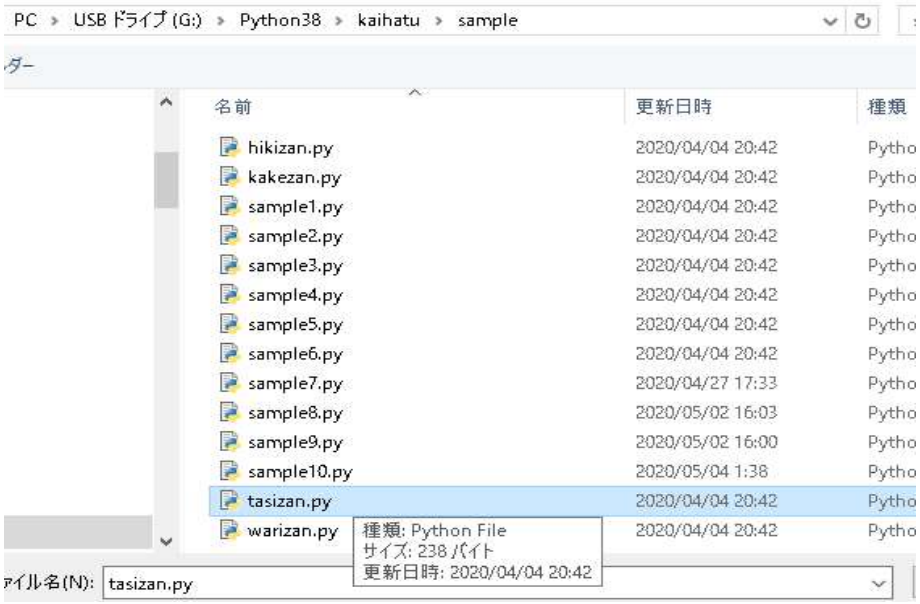
簡単な２項目の足し算計算プログラムを作ってみましょう。
以下のプログラムリストを上図右の編集画面に入力します。
input コマンドは、input('表示文')のように()の中に入力要求の文字を'で括って入力します。以下リストの#はパイソンではコメントを意味し、プログラムコマンドとして実行しません。理解の助けのために、自由に好きな場所に入れることができます。

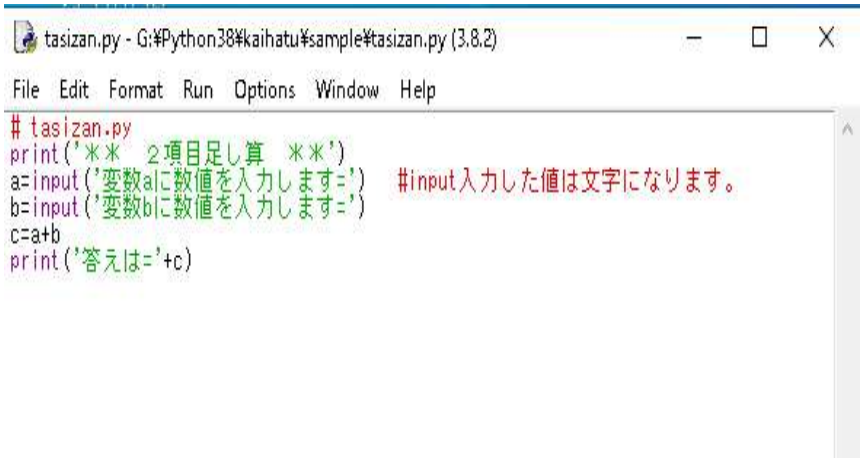
```
# tasizan.py
  print('＊＊　２項目足し算　＊＊')
  a=input('変数 a に数値を入力します=')  #input 入力した値は文字になります。
  b=input('変数 b に数値を入力します=')
  c=a+b
  print('答えは='+c)
```

尚、一からプログラムを入力するのが面倒であれば、既に下図のように G:¥Python38¥kaihatu¥sample フォルダーに tasizan.py を用意していますので、IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より tasizan.py を読込

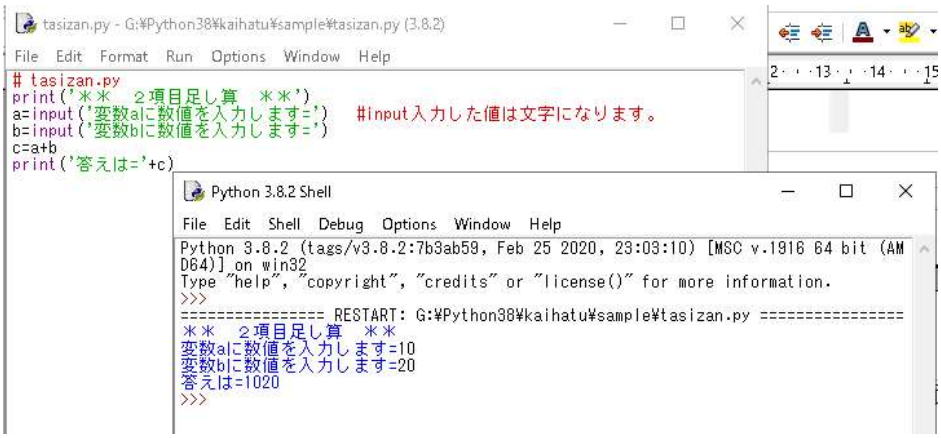
みます。







上図画面の Run の Run Module をクリックしますと下図のように実行画面が表示され値を入力します。

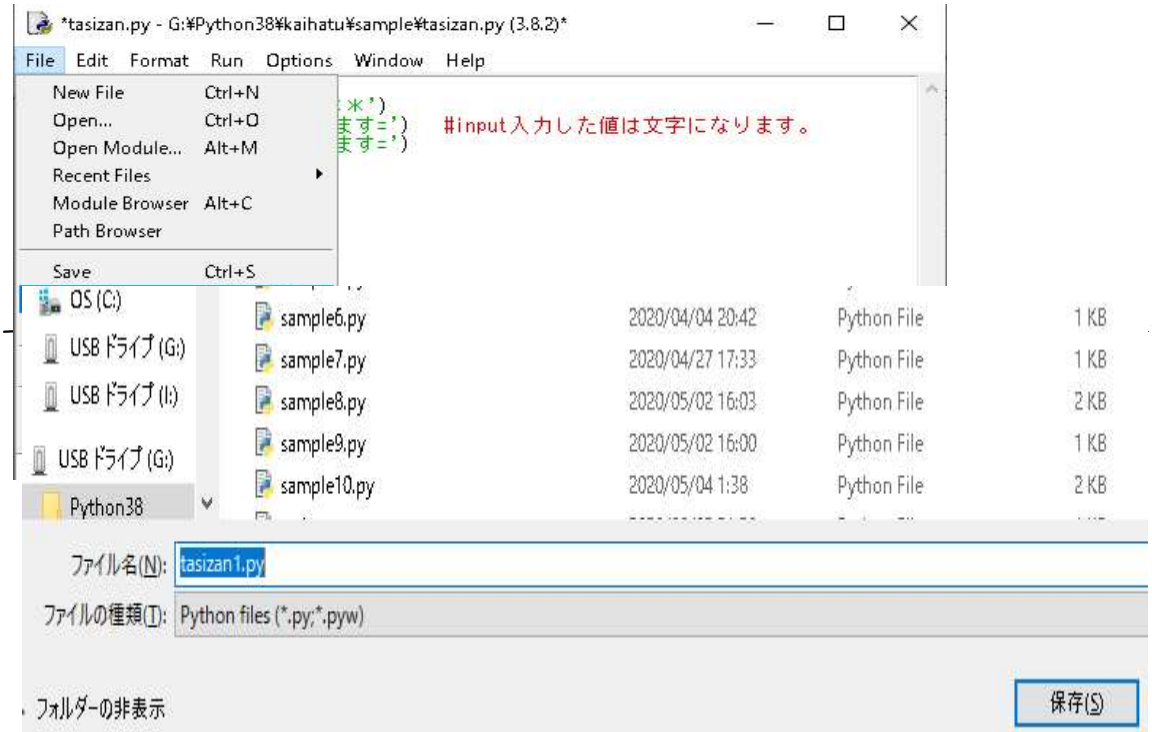


ところで上記プログラムでは数字の足し算結果になりませんでした。入力した２項目値を文字列として足しています。変数には文字と数値があると申し上げましたが、input で入力した値は文字変数に受け取りますので、文字変数を数値変数に変換する必要があります。
数値変数には、整数の int 変数と小数点を含む float 変数があります。
とりあえず、分かりやすい int 変数での足し算をしてみます。
上記プログラムの５行、６行を以下のように変えて下さい。

```
c=int(a)+int(b)
print('答えは='+str(c))
```

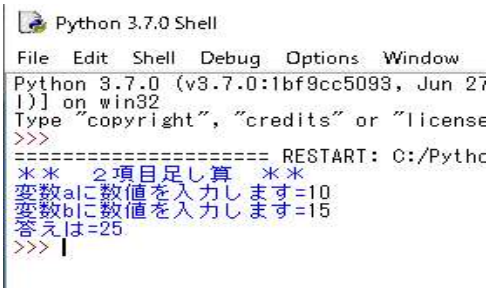
この変更で Run の Run Module で実行した場合、下図の変更していいか聞いてきますのでキャンセルします。

変更プログラムを tasizan1.py の別名を付けて登録することにします。
1行の#tasizan.py を#tasizan1.py に修正します。
その上で、File の SaveAs をクリックします。



存しま

す。その上で Run の Run Module を実行し様々な数値を入力してみます。マイナス値も入力してみて下さい。

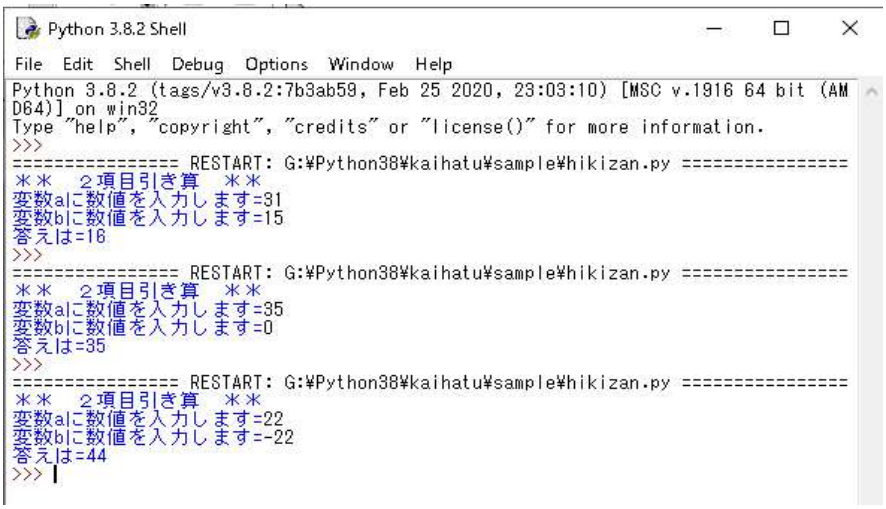


尚、プログラムの実行を続ける場合、画面はプログラム編集画面とプログラム実行画面の２画面が立ち上がっていますので、少し迷いますが Run の Run Module があるプログラム編集画面に切り替えて Run の Run Module をクリックして続けて実行します。

（２）引き算

```
#hikizan.py
  print('＊＊　２項目引き算　＊＊')
  a=input('変数 a に数値を入力します=')
  b=input('変数 b に数値を入力します=')
  c=int(a)-int(b)
  print('答えは='+str(c))
```

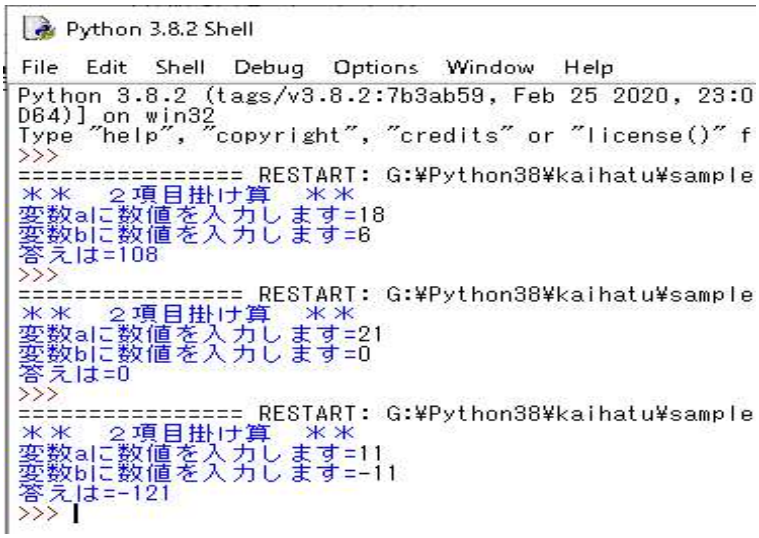
G:¥Python38¥kaihatu¥sample フォルダーに hikizan.py を用意していますので、IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より hikizan.py を読込みます。種々の値を入力してみて下さい。例えば、0 とかマイナス値も。



（３）掛け算

```
#kakezan.py
  print('＊＊　２項目掛け算　＊＊')
  a=input('変数 a に数値を入力します=')
  b=input('変数 b に数値を入力します=')
  c=int(a)*int(b)
  print('答えは='+str(c))
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より kakezan.py を読込みます。



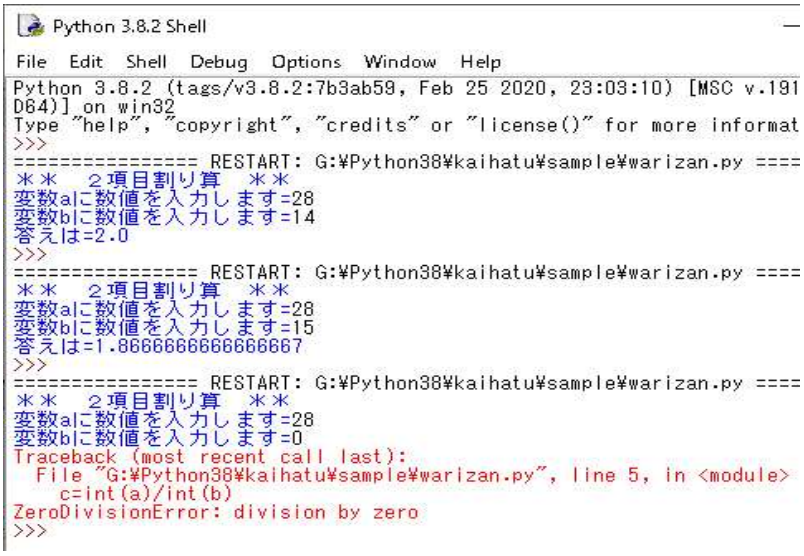
（4）割り算

```
#warizan.py
  print('＊＊　２項目割り算　＊＊')
  a=input('変数 a に数値を入力します=')
  b=input('変数 b に数値を入力します=')
  c=int(a)/int(b)
  print('答えは='+str(c))
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より warizan.py を読込みます。

種々の値を入力してみて下さい。例えば、0 とかマイナス値も。

但し、変数 b の値が 0 の場合、ゼロ除算エラーになります。プログラムを作るとき気を付けることとして 0 除算が起こらないように配慮する必要があります。



また、b に割り切れない値を入力した場合、例えば 28/15 の答えは 1.8666666666666667 になります。

割り切れる 28/14 で入力した場合、答えが 2 ではなく 2.0 と表示されます。

何故なのか、ここで上記プログラムの c=int(a)/int(b)の次の行に以下を加えてみます。

```
print(type(c))
```

type コマンドは、変数の型を表示します。

その上で実行しますと、変数が float になっていることがわかります。

その結果、割り切れても.0 が表示され、割り切れない場合は float 変数の最大格納値を表

示しています。
答えを int 変数にする場合は、以下のように c を int にします。

```
print('答えは='+str(int(c)))
```

値は少数以下が切り捨てられます。
特に割り算の場合、ゼロ除算エラー、切り上げ、切り捨て、四捨五入の丸め等の細かな処理記述が必要になります。ここでは、こうした細かなプログラムでの配慮は少し難しくなりますので課題として残して先に進めます。

（５）プログラム的思考について

ところで、小学校で習う掛け算、割り算は、紙の上で計算しなくてはならず、上記の計算式のようには簡単ではありませんね。ここでは小学校で習う四則計算をその計算手順でプログラム記述してみることにしてみます。
仮に、２桁同士の掛け算とし a=15,b=43 とでもしましょうか。
学校で習う掛け算は以下のような手順になります。

① a と b の各々を 1 の位の値と 10 の位の値に分解します。
② b の 3 と a の 5 を掛けて　15 を求めます。
③ b の 3 と a の 10 を掛けて　30 を求めます。
④ b の 40 と a の 5 を掛けて 200 を求めます。
⑤ b の 40 と a の 10 を掛けて 400 を求めます。
⑥ 2 から 5 までを合計して　645 を求めます。

以上をプログラミングすると以下のようなプログラムになります。

```
#sample1.py
a=int(input('二桁の数字 a '))
b=int(input('二桁の数字 b '))
#a の 1 の位を求め、その値を a1 とし、10 の位を a10 とします。
a10=int(a/10)*10
a1=a-a10
#b の 1 の位を求め、その値を b1 とし、10 の位を b10 とします。
b10=int(b/10)*10
b1=b-b10
#合計を c とし以下の計算式で求めます。
c=b1 *a1+b1*a10+b10*a1+b10*a10
print('答えは'+str(c))
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より sample1.py を読込みます。

如何でしょうか。我々大人は上記の掛け算が出れば、電卓をたたくか、無意識に紙の上で計算しています。しかし上記手順を記述しようとすると結構難儀します。もし桁数が２桁でなくより多くの桁数の場合はまた厄介です。
何故ここでプログラム的思考を取り上げたかと言いますと、文科省が小学校からのプログラム教育の意義をプログラム的思考の醸成としています。しからば、プログラム的思考とは何ぞやの疑問を抱きます。巷の本屋には、小学校からのプログラム教育に合わせて多くの関連本が並んでいます。その多くは、スクラッチ等の完成した部品を用意し、その組み合

わせで意図した動きをパソコン画面にシュミレートすることで、プログラム的思考を育むとの触れ込みです。プログラム言語によらず、スクラッチのような部品の貼り込みで果たして小学校で習う上記掛け算のプログラム記述ができるのでしょうか。少なくとも学校で習う掛け算程度のプログラミングはさせたいものです。

私の思うプログラム的思考とは、小学校で習ったこうした掛け算のような手順なのです。折角の小学校からのプログラム教育を成果がないものにしてはいけない、そうした思いから昔習った掛け算の計算手順を思い出しました。プログラム記述を見ながら、プログラムの実行結果を見ることが出来るのは、パイソンがインタープリター言語であるからで、プログラム教育言語として評価される所以です。

更に、今回掛け算に取り組んで改めて気付いた点があります。それは、掛け算は足し算の繰り返し処理であること、割り算は引き算の繰り返し処理であることです。コンピュータの特徴は、同じことを間違わず嫌がらずやってくれます。コンピュータのない時代だからこそ掛け算が生まれたのかと思います。コンピュータを使いますと、上記の掛け算はいとも簡単になります。繰り返し処理を使ったａをｂ回足すか、逆にｂをａ回足すことで掛け算の値を求めるのことが出来ます。

プログラム的には、小学校で習う上記方法より極めて単純でわかり易いですね。また、桁が何桁であっても構わない利点があります。但し、繰り返しコマンドの for コマンドの説明については後に譲ります。

```
#sample2.py
print('＊＊　足し算による掛け算　＊＊')
a=int(input('二桁の数字ａ'))
b=int(input('二桁の数字ｂ'))
c=0
for i in range(b):  #ｉは０から始まり、回数をカウントしている変数
                    #range は()の中の b-1 になるまで繰り返し
    c=c+a
print('答えは'+str(c))
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より sample2.py を読込み実行してみます。

またここで考えるべきことがあります。ａをｂ回足す場合とｂをａ回足す場合では、計算時間が異なることです。繰り返し数が少ない方が時間が短くなります。

a=15,b=43 であれば、上記プログラムは以下にした方がいいプログラムです。

```
for i in range(a):
    c=c+b
```

少ない桁であればそうした配慮は無視できますが、とてつもない回数を繰り返すような計算の場合は、かかる時間が極めて重要になります。出来れば上記プログラムも、ａとｂを比較して少ない値を繰り返すプログラムが最善です。

ついでに割り算ですが、引き算の繰り返しにより商と余りを求めることが出来ます。

次に出てくるwhileの繰り返しコマンドを使いますが、次のようなプログラムになります。

```
#sample3.py
print('＊＊　引き算による割り算　＊＊')
a=int(input('二桁の割られる数字ａ'))
b=int(input('二桁の割る数字ｂ'))
syo=0
amari=a
while amari>=b:　#amari>=bの間繰り返します
        syo=syo+1
        amari=amari-b
        print(amari)
        print(syo)
print('商＝'+str(syo))
print('余り＝'+str(amari))
```

IDLEをクリックしてFileのOpenでG:¥Python38¥kaihatu¥sampleよりsample3.py
を読込み実行してみます。

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:
D64)] on win32
Type "help", "copyright", "credits" or "license()" for
>>>
========== RESTART: G:¥Python38¥kaihatu¥sample¥s
＊＊　引き算による割り算　＊＊
二桁の数字a 21
二桁の数字b 3
18
1
15
2
12
3
9
4
6
5
3
6
0
7
商＝7
余り＝0
>>>
========== RESTART: G:¥Python38¥kaihatu¥sample¥s
＊＊　引き算による割り算　＊＊
二桁の数字a 21
二桁の数字b 11
10
1
商＝1
余り＝10
>>>
```

プログラム的思考とは、コンピュータを利用した問題解決の最適な解法を導きだす思考方法と言えるのではないでしょうか。決して問題解決の解法はひとつではなくより合理的な解法を発想できるか、その能力を磨くことがこれからのAI時代に求められています。そのためのツールとして何を利用するのがいいか、思い立った解法を即試せるプログラム言語がこれからのAI,IOT時代の言語と言えます。

3. for、whileコマンドによる繰り返しとifコマンドによる判断処理

（１）エラー回避処理

これまでの四則計算では計算式記述でのプログラムは難しいことはなく、楽勝と思われたかと思います。しかしプログラムで厄介なことは、自分で使うプログラムなら多少途中でエラーが出ても値を入れ直せばいいのですが、他人には途中でエラーが出るようなプログラムでは、計算結果自体にも不信感を持たれます。

四則計算のプログラムで配慮しなくていけないことは、数値以外の文字を入力した場合、割り算の０で割る場合のエラー回避処理です。

人間が操作するプログラムの場合、人間はミスを犯すものという前提でプログラミングする必要があります。実はここに、プログラム負荷がかかることが多いのです。

まず入力した値が数字なのか文字なのかの判断が必要ですが、どのような方法でプログラミングしたらいいでしょうか。もっともシンプルな考えは、入力値に０から９以外の文字

がないことです。
判断コマンドとして、if コマンドを使用します。
先に、コンピュータの機能は、記憶、計算、判断と申し上げましたが、プログラムにおいては頻繁に if コマンドを使い、いろいろ判断処理を作ります。
if コマンドは以下のような記述になります。

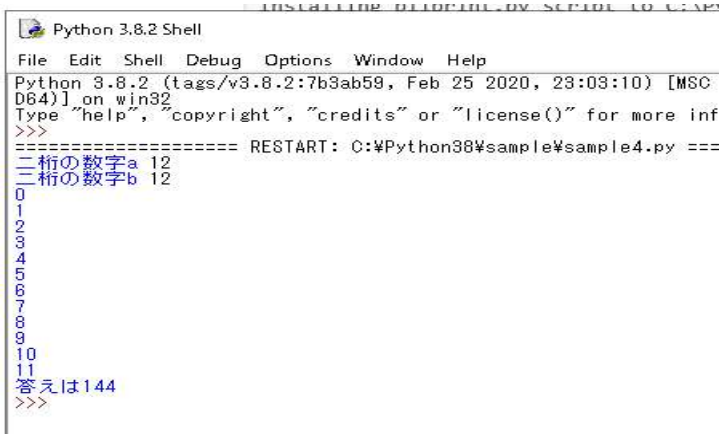
```
if 条件式 ：
   上記条件に合致した場合の処理を記述
else ：
   条件に合致しない場合の処理を記述
```

ここで、パイソン言語の特徴である、プログラム記述の字下げによるプログラム実行順序の制御について説明します。
上記例では、if と else は必ず左余白の字下げで同じ位置からの記述をします。
更に、if 条件に合致した場合の処理記述は、if より右に字下げされた位置に記述します。
このルールに反する場合は、エラーになります。
次に、入力した値に空白ないし文字が含まれていないかの判断プログラムはどのように記述するかを考えます。こうした判断のプログラムは、一般的には正規表現と言われる記述で判断しますが、ここではエラーがあった場合の対応コマンド try 文を使って
#sample2.py にエラー回避処理を施し、#sample4.py とします。

```
#sample4.py
err =1              #エラーを判断する変数 err を用意し、予めエラー判断の１をセット
while err==1:    #エラーの間繰り返します。
                    #try は except の対で使用します。入力値に文字が含まれ int 変換で
                    #エラーになった場合に except にいきます。エラー表示し err は１の
                    #ままですので、while err==1 により元に戻り a の再入力になります。
        try：
          a=int(input('二桁の数字 a '))
          if a>=-99 and a<=99:          #a は-99 から 99 の値を許容します。
             err=0                        #この条件により入力範囲を設定します。
          else:
             print('入力値が２桁を超えています。a を再入力します。')
        except:
             print('入力値が空値か文字が含まれています。a を再入力します。')
err =1
while err==1:
        try：
          b=int(input('二桁の数字 b '))
          if b>=-99 and b<=99:
             err=0
          else:
             print('入力値が２桁を超えています。b を再入力します。')
        except:
             print('入力値が空値か文字が含まれています。b を再入力します。')
c=0
for i in range(int(b)):   # i は 0 から始まり、回数をカウントしている変数
                          #range は()の中の値－１になるまで繰り返します
    print(str(i))          #念のため i の値をプリントしてみます
    c=c+int(a)
 print('答えは'+str(c))
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より sample4.py

を読込み実行してみます。



上図のように、確かにｉの値は０からｂ-１の１１まで繰り返しています。
様々な場面で、forとwhileの繰り返し処理が使われますので覚えておいて下さい。

（２）関数記述

さて、上記プログラムではエラー回避もでき、実用的プログラムになってきました。
しかし､ａの入力記述とｂの入力記述がａ,ｂの違いのみで同じ記述になっており､ひと工夫があっていいように思われたかと思います。処理を１個作り、それを呼び出して使う一般的にはサブルーチン、パイソンでは関数を使って記述してみます。
関数記述は、以下の記述形式になります。

```
def　関数名(引数):　　#引数がない場合は():複数の引数を渡す場合は、(引数,引数..)です。
    ここに関数処理を記述します。
    最後に呼出側に値を返す場合は、以下の記述をします。
    return 戻し変数
```

呼出側プログラムでは、戻り値がある場合は　x = 関数名(引数,引数..)
ない場合は　関数名(引数,引数..)
引数もない時は　関数名()　と記述します。
#sample4.pyに関数記述を入れて#sample5.pyで登録します。

```
#sample5.py
def suujinyuryoku(message,hensu):
    err =1
    while err==1:
        try :
            x=int(input(message+hensu))
            if  x>=-99 and x<=99:
                err=0
                return(x)
            else:
                print('入力値が２桁を超えています。'+hensu+'を再入力します。')
        except:
            print('入力値が空値か文字があります。'+hensu+'を再入力します。')
#メイン
a=suujinyuryoku('二桁の数字' ,'a')
b=suujinyuryoku('二桁の数字' ,'b')
c=0
for i in range(int(b)):
    c=c+int(a)
```

```
print('答えは'+str(c))
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より sample5.py を読込み実行してみます。

関数を使うとプログラムがすっきりし、メインはメイン、関数は関数で別個にプログラムを考えればいいので非常に考えやすくなります。尚、関数はメインより前に記述します。

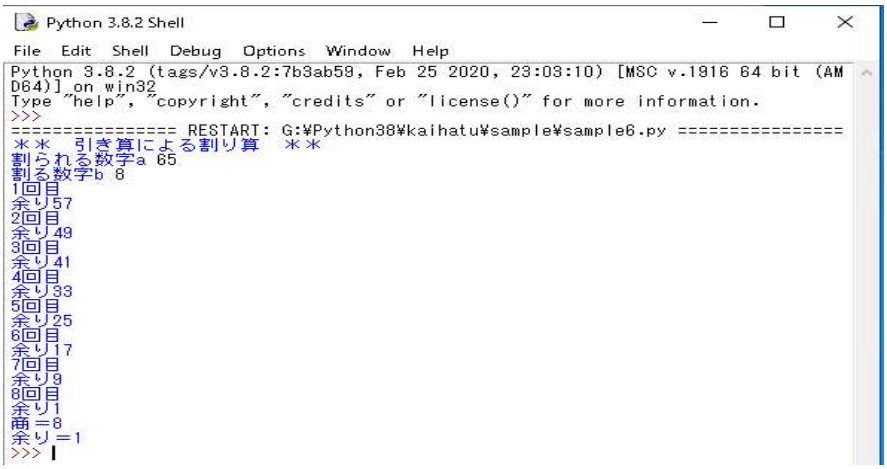
4. プログラムチェック記述とデバック作業

プログラム作りに必ずつきまとい、厄介な作業にデバックと言われるプログラムミスのチェック作業があります。この作業をどのように考えるかは、プログラマーの適性に関わる重要なことです。ひとつは、車の運転で自らハンドルを握って運転した場合は、道路の道筋を覚えているものですが、助手席にいた場合は覚えていないものです。プログラムも思ったように動かず、いろいろ試行錯誤してプログラムのロジックを覚えるものです。また、何度も試行錯誤した場合は、エラーの起こらないようなプログラムを作り、部品化して皆で利用しようとの発想が生まれます。苦労をすることで、それを回避する知恵が生まれます。それを厭うようであれば、プログラマーとしての適性に欠けます。私がパイソン言語を小学校からのプログラミング言語として取り上げたのは、試行錯誤に向いたこの点にあります。

プログラムを理解する上で大切なことは、最初に申し上げましたように、コンピューターの機能は、記憶、計算、判断であり、コンピューターの中でどのような動きをしているのかを把握できなければ、プログラムを作ることはできません。小学生と言えども、上記で説明したことは理解できるはずで、スクラッチのようにミスを起こさない用意されたコンピュータ部品を張り付ける教育には、大きな疑問を抱きます。人間はミスするもので、そのミスから何を学ぶかが大切です。

少し前置きが長くなりましたが、#sample3.py にデバックのためにコンピュータの動きを把握するコマンドを差し込んでみます。要は、動きを把握するためにプログラムがどこを実行したのか、また変数の値がどのように変わったのかを print 文を各所に差し込んで、画面に表示されたコンピュータの動きを見て、意図通りのプログラムかどうかを判断します。#sample3.py を#sample6.py で登録しています

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より sample6.py を読込み実行してみます。



上図のように経過とその時の変数の値がわかります。

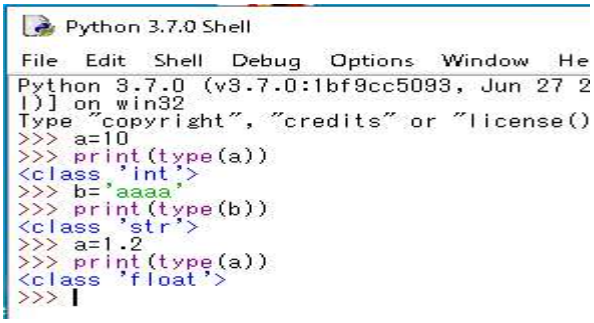
5. 変数の型

上記の sample4.py プログラムにはもう一つ問題があります。

それは、長さがあくまで整数の値の場合で、少数点を入れた場合エラーになります。

小数点での計算程度はさせたいことと、もうひとつここで少数点問題を扱うのは、変数の

型についての説明の必要性があります。
整数型変数(int)は、一般には１０進数ですが、２進数、８進数、１６進数の数値を扱います。小数点を含んだ数値は、ｆｌｏａｔ型数値に当たります。一般的な固定小数点と浮動小数点を扱います。更に、ブール（ｂｏｏｌ）型変数は、比較演算する場合の結果値、True(真)とFalse(偽)という特殊があります。この変数も、時折出てきます。
ここでは、変数として文字、int 数値変数、float 数値変数、ブール（ｂｏｏｌ）型変数程度を知っていれば結構です。
IDLE の shell 実行画面で >>10>1 と入力してみて下さい。
結果値として、True が返ってきます。逆に、>>10<1 の入力
にはFalseが返ってきます。更には、特殊な値としてNoneと言うものがあります。



変数の型として、更に複数のデータをグループとして取り扱うリスト型、変更不可のタプル型、キーと値を対に持つ辞書型があります。殊に、辞書型はパイソン特有なグループ変数で、パイソン言語では使い方を習熟する必要があります。尚、変数の型を知りたい場合は、上図のように変数ａ、変数ｂの type を print すれば型が表示されます。
変数型についてはこの程度の理解で、具体な必要性が出た時、ネットで調べるか文法書を紐解いて下さい。

６、演算子

ここで、改めてパイソンでの演算子を列記します。

演算子

＋	足す
－	引く
／	割る
＊	掛ける
％	割り算の余り
**	べき乗

比較演算子

ａ == ｂ	ａとｂは等しい
ａ != ｂ	ａとｂは等しくない
ａ < ｂ	ａはｂより小さい
ａ <= ｂ	ａはｂより小さいか等しい
ａ > ｂ	ａはｂより大きい
ａ >= ｂ	ａはｂより大きいか等しい

ブルー演算子

ａ and ｂ	ａとｂの時
ａ or ｂ	ａ またはｂの時
ａ not	ａでない時

７. インターネットデータによるグラフ作成

さて、ここまでは何となくパイソンの文法も理解し、少し退屈になってこられた方もおられるかと思います。そこで、少し話題を変えて温度データのグラフを書いてみたいと思います。

(1) 気温データの折れ線グラフ

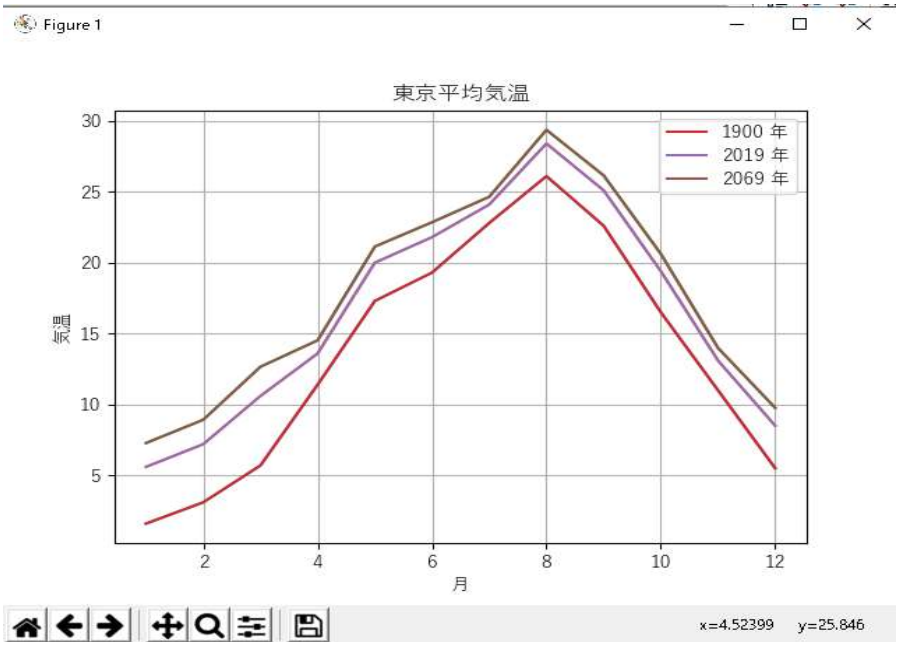
インターネットには、極めて多くのデータがあります。例えば、気象庁のサイトでは過去の気温等のデータを簡単に調べることができます。地区ごとの特に東京は1872年よりのデータがあり、そうしたデータをグラフ化することで温暖化問題を可視化して考えることが可能になります。パイソンには、簡単にグラフ化するツールが用意されています。

ここでは、ひとつの試みとして東京の1900年と2019年の気温データの年平均気温グラフを描いてみます。

```
#sample8.py
# 折れ線グラフを出力
import numpy as np                #import 文で numpy パッケージを取込ます
import matplotlib.pyplot as plt     #import 文で matplotlib パッケージを取込ます
from matplotlib import rcParams

rcParams['font.family'] = 'sans-serif'  #漢字が使えるように設定
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro',\
            'Yu Gothic', 'Meirio', 'Takao', 'IPAexGothic', 'IPAPGothic',\
            'VL PGothic', 'Noto Sans CJK JP']
plt.title("東京平均気温")
plt.xlabel("月")
plt.ylabel("気温")
plt.grid(True)                    #格子を印字
h=float(50/119)
x=np.array([1, 2, 3, 4, 5 ,6 ,7 , 8 ,9 ,10 ,11 ,12])
y1=np.array([1.6,3.1,5.7,11.4,17.3,19.3,22.8,26.1,22.6,16.5,11.0,5.5])  #1900 年月別平均気温
y2=np.array([5.6,7.2,10.6,13.6,20.0,21.8,24.1,28.4,25.1,19.4,13.1,8.5]) #2019 年月別平均気温
y3=(y2-y1)*h+y2
plt.plot(x, y1)
plt.plot(x, y2)
plt.plot(x, y3)
plt.plot(x,y1,label="1900 年")     #y1 のラベル文字
plt.plot(x,y2,label="2019 年")     #y2 のラベル文字
plt.plot(x,y3,label="2069 年")     #y2 のラベル文字
plt.legend()                       #y1,y2 のラベル印字
plt.show()
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より sample8.py を読込み実行してみます。



matplotlib の詳細な使い方はネット上にいろいろ掲載されていますので、そちらをご覧ください。尚、こうした高度なプログラムですが、タイトル名、横軸、縦軸の指定と対応データをセットするだけでグラフを書くことが可能です。

さて、上記平均気温グラフは、この１００年で明らかに温暖化を示しています。過去の温暖化率を用いて、今後５０年後の予測グラフも表示しております。

ところで、上記プログラムでの技術的要素として、x = np.array、y1 = np.array、y2 = np.array の()の中の[]で括られたデータですが、配列変数と言われる複数データをまとめて管理するデータ形式が使われています。パイソンでは、こうしたまとまったデータを変数として管理するものとして、配列変数、辞書変数、タプル変数があります。

少ないデータであれば、配列変数に直接値を入れることで済みますが、データが多量になる場合はエクセル形式のＣＳＶファイルにして、ファイルからデータを読むというプログラムにします。ここでは、ファイルからのデータの読み込みについては触れません。因みに気象庁データはファイルでダウンロードが可能です。

(2) 体格指数 BMI 円グラフ

学校では、恐らく毎年身体検査を行い、身長．体重のデータをお持ちと思います。

子供の肥満問題もあり、円グラフによる肥満構成プログラムを作成してみます。

ネットでいろいろ調べますとＣＡＳＩＯ社の下記サイトに子供の肥満のページがあり、子供の場合はローレル指数による５段階判定の説明があります。

https://keisan.casio.jp/exec/system/1161228729

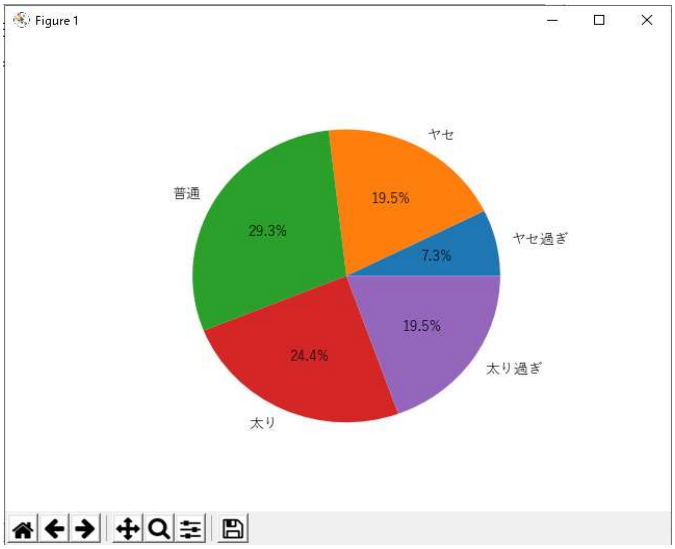
一方、下図の政府統計データに学校での男女別平均身長．体重データがあります。

こうしたデータと比較しながら、生徒の体格分析をなさってみて下さい。
国の統計データで全国的な学生の身長．体重のデータセットを探しましたが、適当で解析に向くデータがありませんでした。
まず、簡単な肥満５分類での人数による円グラフ作成プログラムを以下に表示します。

```
#sample9.py
# 肥満円グラフを描画
import matplotlib.pyplot as plt
from matplotlib import rcParams

rcParams['font.family'] = 'sans-serif'  #漢字が使えるように設定
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro',\
            'Yu Gothic', 'Meirio', 'Takao', 'IPAexGothic', 'IPAPGothic',\
            'VL PGothic', 'Noto Sans CJK JP']
labels = ['ヤセ過ぎ','ヤセ','普通','太り','太り過ぎ']
sizes = [3, 8, 12, 10,8]  #人数
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct="%1.1f%%")
plt.show()
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より sample9.py を読込み実行してみます。



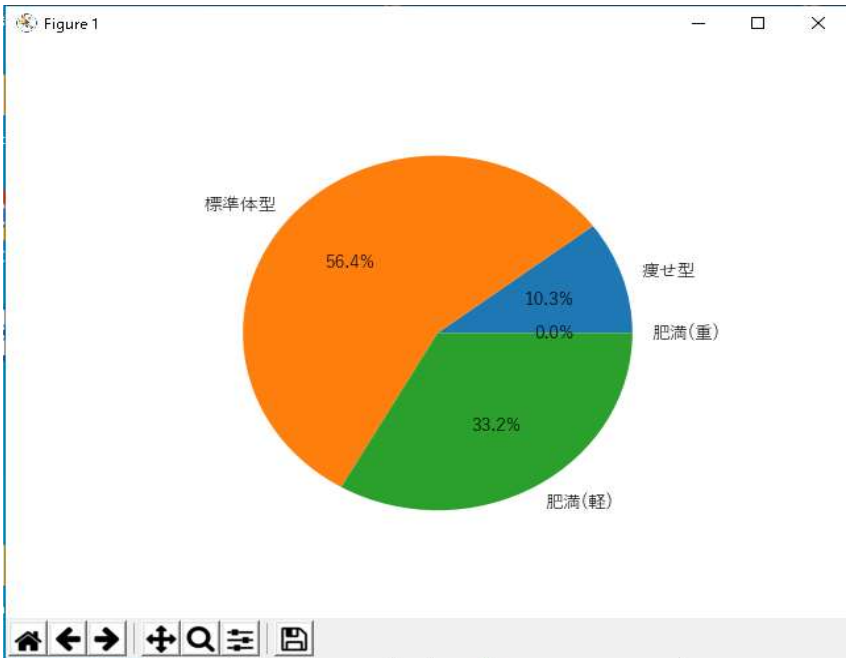
次に、今回のサンプルプログラムでは余り適当でなサンプルのＣＳＶデータを利用して円グラフを書くプログラムを以下に表示します。敢えて、このプログラムを用意しましたのは、実際の身長．体重データをエクセルで作成し、そのデータで肥満度の評価をしてみようと言う場合を想定しました。
サンプルＣＳＶは G:¥Python38¥kaihatu¥sample¥sintyotaijyu.csv です。

```
#sample10.py
# ｃｓｖ読込円グラフを描画
import matplotlib.pyplot as plt
from matplotlib import rcParams
import csv
import kyoutuu
import sys
import os
initial_dir=os.getcwd() #カレントデレクトリー
```

```
dirname = os.path.dirname(__file__) #ディレクトリ
#データ読込
def dataread():
    sizes = []
    size1 = 0
    size2 = 0
    size3 = 0
    size4 = 0
    try:
        with open(initial_dir+'/sintyotaijyu.csv') as f:     #データは sample10.py と同じ所
                reader = csv.reader(f,skipinitialspace=True)
                reader_list = [e for e in reader]
                for i in range(len(reader_list)):
                    record = reader_list[i]
                    sintyo=float(record[0])/100 # m
                    taijyu=float(record[1])     #kg
                    bmi = round(taijyu / (sintyo **2))
                    if bmi < 18.5:
                        size1 = size1 + 1
                    if bmi >= 18.5 and bmi < 25:
                        size2 = size2 + 1
                    if bmi >= 25 and bmi < 30:
                        size3 = size3 + 1
                    if bmi > 30:
                        size4 = size4 + 1
                sizes = [size1,size2,size3,size4]
                return sizes
    except csv.Error as e:
            kyoutuu.message(str(e))
    except Exception as e:
            kyoutuu.message(str(e))

#メイン
rcParams['font.family'] = 'sans-serif'     #漢字が使えるように設定
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro',\
        'Yu Gothic', 'Meirio', 'Takao', 'IPAexGothic', 'IPAPGothic',\
        'VL PGothic', 'Noto Sans CJK JP']
labels = ['痩せ型','標準体型','肥満(軽)','肥満(重)']
sizes=dataread()
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct="%1.1f%%")
plt.show()
```

IDLE をクリックして File の Open で G:¥Python38¥kaihatu¥sample より sample10.py を読込み実行してみます。



ここで敢えてグラフ作成をとりあげましたのは、学校でのプログラム教育では中学、高校、大学と先に進むほど、データサイエンスと言われるデータ解析と予測でのＡＩ技術が課題になります。
プログラム記述のルールさえ知れば、容易にグラフ化できますので、ぜひ身の回りにある各種データを使ってこうした取り組みにチャレンジいただければと思います。

8. 宿題

（１）これまでの四則計算はプログラムで四則が固定されておりました。これを計算式の+, -, *, /の区分を入力させて、足し算、引き算、掛け算、割り算のどれでもいい問題を作るプログラムを作りなさい。
（２）西暦での誕生日（８桁の数字）と本日の西暦日付（８桁数字）を入力して、年齢を計算するプログラムを作りなさい。
（３）例えば１０問の問題を自動的に作り、採点結果を表示するプログラムを作りなさい。ネットで python と乱数で検索すれば、乱数の使い方がいろいろ出てきます。
（４）元金と金利を入力して、10 年後の利息を単利、複利で計算しなさい。
（５）現在の日付と時間、分数を入力して 200000 分後の日付、時間、分数を求めなさい。

プログラム授業を進めるには、どのような身近な課題を取り上げるかが重要です。
プログラミングに相応しい各種問題を生徒さんと一緒に考え、プログラム訓練をすべきです。そうすることで、プログラムへの興味とプログラム的思考がアップします。

9. 2章を終えて

ここまでお読みいただきプログラミングに興味を持たれれば、間違いなく次章も読みこなし十分皆様はパイソンプログラマーとしての一歩を踏み出すはずです。私も先に述べましたように、ポケコン BASIC につられてプログラム自動生成と言う難儀な課題に挑戦するはめになってしまいました。

ところではじめにで申し上げましたように、私がパイソン言語に取り組むことになった契機は小学校でのプログラム教育開始でした。なんとか小学校での算数問題とプログラムをリンクできないかを考え、今回簡単な四則計算のプログラム作成に取り組みました。これまで何気なく掛け算、割り算のプログラム記述をしていましたが、小学校で習う位取りでの計算をプログラミングして今更ながら厄介な手順であったこと、掛け算は足し算の繰り返しであ

ること、割り算は引き算の繰り返しであることを認識することになりました。コンピュータの間違えず、また飽きずに行う繰り返し機能によって、人間の能力が飛躍的にアップすることを思い知りました。更に、小学生が位取りでの計算を習うなら、ここまでの四則計算でのプログラミングに十分ついてこれると思うようになりました。

プログラム教育をどのように進めるのか、その下駄を預けられた小学校の先生の戸惑いを耳にします。プログラム言語教育が難しいからか、言語教育を回避しあるいは子供の興味を引くためにゲーム的要素を取り入れようとする傾向があります。それでなくとも子供はゲームに夢中になりますので、そうした方向は慎むべきと思います。

私は、できれば親御さんも一緒に入門編でのプログラムを一緒に取り組んでいただき、これからのAI時代の「誰もがプログラマー」にチャレンジしていただきたいと思っています。今回は四則計算にとどめますが、中学校、高校ともなれば、まだまだ多くの数学的課題があります。また、入力した値をデータとして保存して統計的分析をしたい等々、今後そうした課題をストーリー化、プログラム化に取り組み２章をより充実したいと思っています。

次章からはプログラムは自動生成で出来ますが、そのプログラムを理解するにはこれまでと違って難儀すると思います。また、小学生の授業でのプログラムとしては向いておりません。ただし、次のステップの課題として、どのようなプログラムが出来るのかを見せていただければと思います。

更に申し上げたいことは、プログラムを作る事以上に大事なことはプログラミングは手段でしかなく、目的ではありません。AIのように、ターゲットとする専門的分野、IOTであればセンサーの知識と利用分野の知識、ゲーマーを目指すなら面白いゲームの発想等、プログラムの次にはより高い各々目的の壁が待っています。目的への夢と希望を持ち続け、その実現への１００人力、１０００人力パワーを持つコンピュータの力を信じれば必ず夢は実現します。次章に取り組んでいただき手段を理解し、目的へ向かわれることを願っています。

第３章. 自動生成によるパイソンプログラム

１．プログラム作成環境の変化と技術的変化

前章でのプログラムは、input 文でデータを入力する極めてシンプルな記述でした。しかし画面は甚だ貧弱で、今皆さんが日頃見ている画面とは大きく異なっています。input 文プログラムでは、文字のフォント、大きさ、色、表示座標を全く指定していません。今様の画面表示には、当然そうした記述がなければ出来ません。コンピュータの機能アップとインターネットの進展は、デザインでの見栄えが優先されプログラム技術は大きく変わりました。またデザイン性のみならず複雑化するプログラムの解決策として、プログラムの部品化、オブジェクト指向と言われる技術変革の波を受けました。

一方日本のＩＴ業界では、それまで技術者であれば少なからずＣＯＢＯＬ等のプログラム言語は習得していたものですが、それまでの技術とのギャップから新たな技術は若い者、下請けに任せ、システム設計を上流、プログラミングを下流と称し、プログラミングを避けるようになり今日の IT 技術力の低下を招きました。

前章で進めてきた四則計算プログラムですが、これを単純な input 文でなくそれなりの画面でのプログラムを作ろうとしますと、Web システムでの HTML で作成するか、GUI（グラフィカルインターフェイス）と言われる結構厄介なプログラミングが要求されます。私は極めて大きな技術変遷での負担を強いられてきて、ようやくパイソンに辿り着きましたが、皆様にはそうした経験をしていただきたくないとの思いがあります。プログラムに取り組むには技術的潮流と取り組むプログラム言語で何ができるのかの見極めが必要です。一度取り組んだものは簡単には捨てられず引きずるものです。

現在様々なプログラム言語がありますが、大まかに言えばクライアントサイド言語なのか、サーバーサイド言語か、あるいは両方の開発が可能な言語かに分類されます。サーバーサイドと言う意味はインターネットサーバーを意味します。インターネット回線を使った LAN でのネットワークと混同しがちですが、Web サーバーシステムはブラウザがあればプログラムをダウロードしなくても簡単に不特定多数とネットワーク接続できるメリットがあります。サーバーサイド、クライアントサイド両システム開発できる言語には、Java とパイソンがあり、更にコンパイルせずにプログラム記述をすれば即実行できるのは、パイソンに限られます。しかし、パイソンもサーバーサイドとクライアントサイドでは、利用する要素技術が異なります。特にサーバーサイドプログラムでは、データーベース、HTML、Javascript、Web フレームワーク等の知識が要求されます。

この章では、そうした知識なしで開発できるクライアントサイドプログラムに限定したプログラム作成をします。パイソンクライアントには、各種センサーでの IOT、ドローン、ロボット等の取り組みへの可能性があり楽しみが多い言語です。

２．自動生成とプログラムパターン

Java での自動生成に１０余年費やしてきましたが、なかなか自動生成技術への評価を受けられませんでした。曰く、プログラム技術が落ちる、曰く１００％の自動性はありえないので結局プログラム修正をするなら最初から自分で作った方がいいなどなど。

しかし、IT 業界の技術者でもパイソン習熟者は少なくこれからの言語で、また素人のプログラマー参入者が増えるとすれば、自動生成により厄介なプログラミングのハードル越えの大きな役割を果たせると思います。前章の四則計算プログラムを GUI 技術で自動生成しそのプログラムをお読みいただければ、自動生成技術に興味を持っていただけると思います。

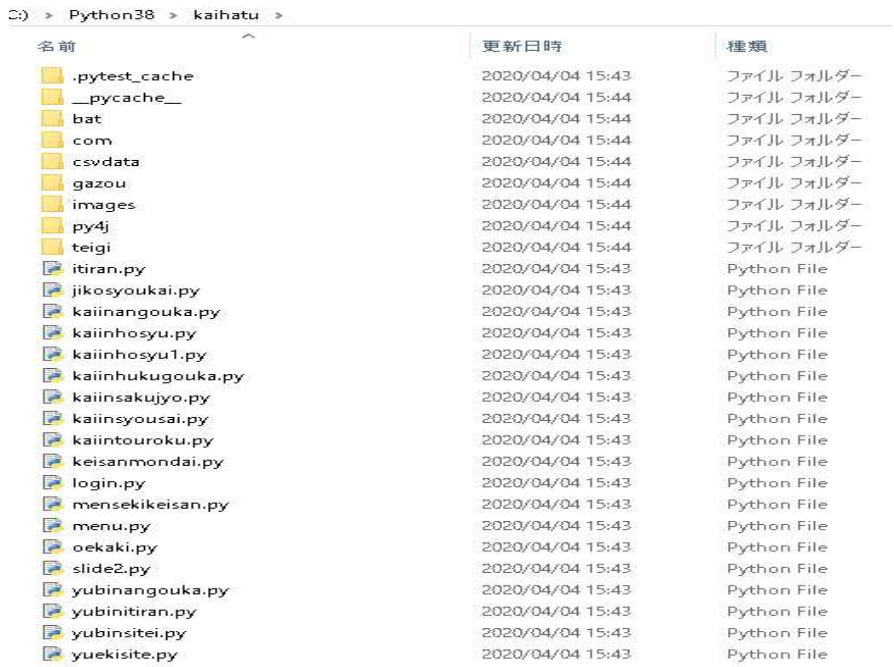
ただ、自動生成はどんなプログラムにも適用できるものではなく、プログラム機能はパターン化でき、そのパターンに用意した自動作成プログラムがある場合に限ります。

例えば、データーを入力するパターン、登録されたデータを読んで、画面に表示するパターンそれも、一個のデータの表示の場合、複数データーの一覧表示、登録したデータを修正、削除するパターン等にパターン化できます。パターンにより、プログラムのロジックは決まってきます。また、パターンにより要求される要素は決まってきます。例えば、登録パタ

ーンの場合であれば、画面に配置する登録項目数は何個あるか、その項目は何桁か、数字か文字か入力したデータをどこに登録するか等々。こうしたプログラムパターンごとに必要な最低の条件設定することで、自動生成プログラムがプログラムを書き出すのです。
無償公開していますスタンドアローン版では、条件入力、メニュー、一覧表示、詳細表示、データ登録の５パターンの自動生成をします。
なお、生成されたプログラムについては、自己責任のもとでご利用ください。一切、弊社では責任を負わないものとします。

３．自動生成システム環境

①自動生成システムで使用するホルダー g:\paython38\kaihatu を確認します。



__pycache__　　パイソンプログラムの翻訳プログラム
bat フォルダー　　各種の BAT ファイル
csvdata フォルダー　　csv ファイル
gazou フォルダー　　画像ファイル
teigi フォルダー　　プログラム定義ファイル
プログラムファイル　識別子.py

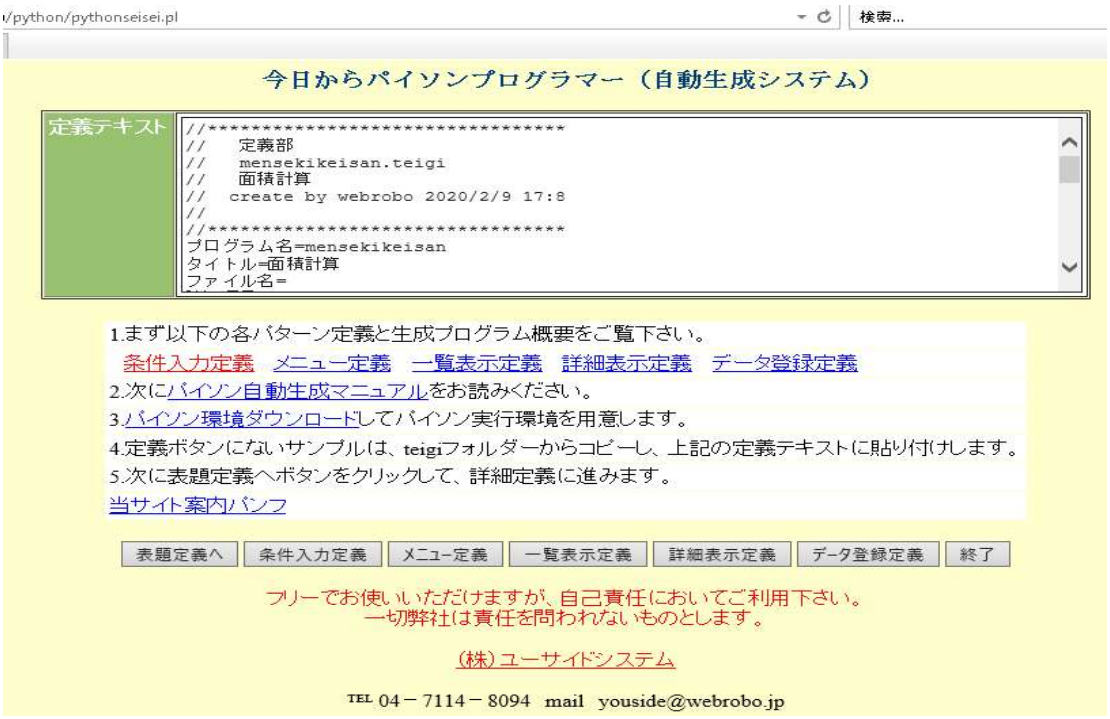
４．面積計算プログラムの自動生成

ここでは、前章の掛け算プログラムを面積計算に変え以下の GUI 画面で作成します。

＊＊　面積計算　＊＊

縦長さｃｍ □□　　2 桁の文字入力項目
横長さｃｍ □□　　2 桁の文字入力項目
答え　　□□□□ 4 桁の文字入力項目
実行 取消 終了

①上記でデスクトップに作った pythonwebseisei をクリックし、インターネットのパイソン自動生成サイトを表示します。
②ｇ:¥Python38¥kaihatu¥teigi にある mensekikeisan.teigi をエデターで開きそのソースを下図画面の定義テキスト欄にコピー、ペーストします。
③表題定義へボタンをクリックします。



④以下の画面が表示されますので、詳細定義ボタンをクリックします。

⑤以下の画面が表示されますので生成ボタンをクリックします。
尚、印刷ボタンをクリックしますと定義詳細を印刷できますので、仕様書として活用できます。
下図の詳細定義画面での操作説明は、次の自己紹介プログラムに回します。ここでは、プログラムを作成、実行してどのようなプログラムが出来るかを優先します。

最初へ 終了

```
//**********************************
//    定義部
//    mensekikeisan.teigi
//    面積計算
//   create by webrobo 2020/2/10 11:52
//
//**********************************
プログラム名=mensekikeisan
タイトル=面積計算
ファイル名=
読込項目ＮＯ=
画面サイズ=900*400
ページ行数=
パターン=1
作成行数=11
文字ポイント=16
背景色=frame1&darkcyan,header&lightcyan,button&skyblue
@11@1@＊＊　面積計算　＊＊@taitol@@@@@@
@31@2@縦長ｃｍ@tate@@@2@2@2@
@51@2@横長ｃｍ@yoko@@@2@2@2@
@71@2@答え@kotae@@@2@4@2@
@91@6@実行@jikou@@@@@@
@92@6@取消@torikesi@@@@@@
@93@6@終了@syuryou@@@@@@
%def
#必ず def jikkou(): の記述で始めます。
#半角項目名が項目名になりますが、その項目に入力された値は、var項目名で受け取ります。
#ただし、文字変数になっていますので、数値として扱う場合は、int(var項目名)で数値変換します。

def jikkou():
    if int(varkotae)==int(varyoko) * int(vartate):
        kyoutuu.kakunin_message('正解')
        field_clear()
        texttate.focus_set()
    else :
        kyoutuu.message('不正解')
        field_clear()
        texttate.focus_set()



#**********************************
#    メインソース
#   mensekikeisan.py
#   面積計算
#   create by webrobo 2020/2/10 11:52
#
#**********************************

from tkinter import *
from tkinter import ttk
import tkinter as ttk
```

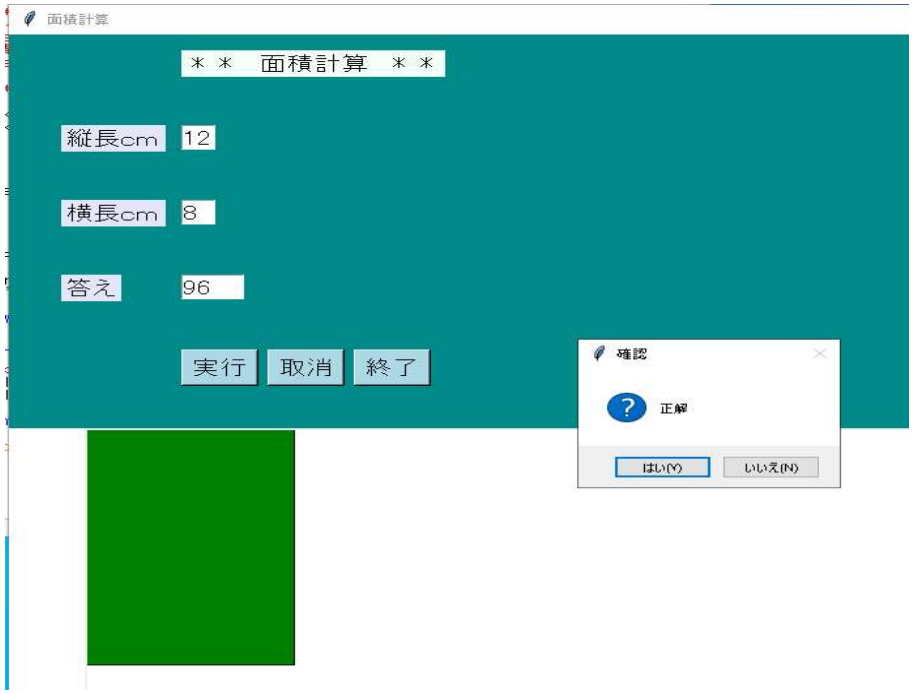
⑥上記画面のように定義部とメインソースが表示されます。
すでに、定義部は C:¥Python38¥kaihatu¥teigi に、ソースは g:¥Python38¥kaihatu フォルダーに入れてありますが、自分で作るプログラムでの操作でも同様の操作をしますので以下の操作で入替してみて下さい。

a. 定義部からメインソースの手前までをコピーして、何らかのエデターを開き貼り付けします。それを g:¥Python38¥kaihatu¥teigi フォルダーに mennsekikeisan.teigi で保存します。

b. メインソースを同様にコピーしてエデターに張り付け、 g:¥Python38¥kaihatu フォルダーに mensekikeisan.py で保存します。

⑦ IDLE を起動し、File の Open をクリックし、 g:¥Python38¥kaihatu¥mensekikeisan.py を開きます。

⑧ Run の Run Module をクリックし実行します。以下画面が表示され整数の値を入力し、実行ボタンをクリックして正解、不正解の判断をさせます。尚、終了ボタンは IDLE では機能しません。右上の×で終了します。



面積計算プログラム定義の説明

定義部は次のように記述されています。

```
//**********************************
//  定義部
//  mensekikeisan.teigi
//  面積計算
// create by webrobo 2020/2/10 11:34
//
//**********************************
プログラム名=mensekikeisan
タイトル=面積計算
ファイル名=
読込項目ＮＯ=
画面サイズ=900*400
```

```
ページ行数=
パターン=1            #条件入力パターン
作成行数=11
文字ポイント=16
背景色=frame1&darkcyan,header&lightcyan,button&skyblue      #frame,header,button の背
                                                                景色
@11@1@＊＊　面積計算　＊＊@taitol@@@@@@　#以下は詳細定義の明細
@31@2@縦長ｃｍ@tate@@@2@2@2@
@51@2@横長ｃｍ@yoko@@@2@2@2@
@71@2@答え@kotae@@@2@4@2@
@91@6@実行@jikou@@@@@@
@92@6@取消@torikesi@@@@@@
@93@6@終了@syuryou@@@@@@
%def
#必ず def jikkou(): の記述で始めます。
#半角項目名が項目名になりますが、その項目に入力された値は、var 項目名で受け取ります。
#ただし、文字変数になっていますので、数値として扱う場合は、int(var 項目名)で数値変換します。

def jikkou():
  henkanti=20 #倍率
  if int(varkotae)==int(varyoko) * int(vartate):
    canvas.create_rectangle(0,0,henkanti*int(varyoko),henkanti*int(vartate),\
                fill = 'green',tag='canvashyouji')
    kyoutuu.kakunin_message('正解')
    canvas.delete('canvashyouji')
    field_clear()
    texttate.focus_set()
  else：
    canvas.create_rectangle(0,0,henkanti*int(varyoko),henkanti*int(vartate),\
          fill = 'green',tag='canvashyouji')
    kyoutuu.message('不正解')
    canvas.delete('canvashyouji')
    field_clear()
    texttate.focus_set()
```

プログラム定義部ソースを保存していれば、定義部ソースを自動生成の定義テキスト欄に貼り付けることで、簡単にプログラム生成することが出来ます。
当パイソンクライアント生成では GUI に Tkinter なる部品を使っています。当部品の Canvas 画面に画像、図形表示も可能です。
算数の図形問題では、ぜひ図形描画を使って理解を深めることをおすすめします。
尚、パイソンでの図形表示ライブラリには、他に matplotlib や pillow があります。
前章での面積計算と比べると、生成されたプログラムは長く難しい印象を持たれたと思います。しかしプログラム定義はご理解いただけるものと思います。

次に生成プログラムについて説明しようと思いますが、聞き流しで受け止めていただければと思います。

```
#********************************
#  メインソース
#  mensekikeisan.py
#  面積計算
#  create by webrobo 2020/2/10 16:35
#
#********************************
from tkinter import *
from tkinter import ttk
import tkinter as ttk
import os, tkinter, tkinter.filedialog, tkinter.messagebox
import webbrowser
import kyoutuu
from datetime import date, timedelta
import subprocess
import csv
import sys
```

上記 import 文は、必要なパッケージ部品を読み込んでいます。

syokisyori(args)では、プログラムリンク変数である args に渡ってきた id、password を当プログラムに取り込んでいます。

```
def syokisyori(args):
    global id
    global password
    global recordno
    for i in range(len(args)):
        if args[i].find('id=')>-1:
            id=args[i].replace('id=','')
        if args[i].find('pass=')>-1:
            password=args[i].replace('pass=','')
texttate.focus_set()
```

jikkou()は実行ボタンをクリックした際の処理を記述しています。
縦長と横長の入力値より答えを求め、入力した答えの値と一致する場合は正解を、不一致の場合は不正解を表示しています。
尚、kyoutuu.kakunin_message という関数にメッセージを表示する記述があります。

```
def jikkou():
    henkanti=20 # ｃｍよりポイント変換値
    if int(varkotae)==int(varyoko) * int(vartate):
        canvas.create_rectangle(0,0,henkanti*int(varyoko),henkanti*int(vartate),\
                                fill = 'green',tag="canvashyouji")
        kyoutuu.kakunin_message('正解')
        canvas.delete('canvashyouji')
        field_clear()
        texttate.focus_set()
    else :
        canvas.create_rectangle(0,0,henkanti*int(varyoko),henkanti*int(vartate),\
```

```
                                fill = 'green',tag="canvashyouji")
            kyoutuu.message('不正解')
            canvas.delete('canvashyouji')
            field_clear()
            texttate.focus_set()

    def jikou_clicked():
        err=datacheck()
        if err!='':
            kyoutuu.message(err)
        else:
            jikkou()
```

#データチェック処理では、すべての入力項目に対して入力値が妥当かどうかチェックします。 g:¥Python38¥LIB¥kyoutuu には弊社が用意した共通関数を記述しており、hyoujyuncheck で入力値の妥当性をチェックしています。

```
    #データチェック処理
    def datacheck():
            errmsg=''
        #tate
            msg=''
            global vartate
            vartate=tate.get()
            zokusei=2
            kubun=2
            hissu=2
            koumokuname="tate"
            koumokutaitol="縦長ｃｍ"
            keta=2
            msg=kyoutuu.hyoujyuncheck(vartate,zokusei,koumokutaitol,koumokuname,kubun,\
                    hissu,keta)
            msg=msg+kobetucheck(vartate,zokusei,koumokutaitol,koumokuname,kubun,hissu,\
                    keta)
            if len(vartate)>keta:
                texttate.delete(0,len(vartate))
            errmsg=errmsg+msg
```

以下は縦項目を入力してエンターキーを押した時、次の横項目 textyoko にカーソル移動します。尚、入力値によって移動先を変えることで、リッチクライアントと言われるカーソル制御が可能になります。

```
    def texttate_enter(event):
        textyoko.focus_set()
```

#ボタン取消処理定義では取消ボタンを押したとき、キャンバスに表示した四角画像を消去し、又入力項目値を消去します。

```
    def torikesi_clicked():
        canvas.delete("canvashyouji")
        field_clear()
```

```
        texttate.focus_set()
    #画面クリアー
    def field_clear():
        radionengou.setvar('nengou',nengouitem[0])
        textyy.delete(0,2)
        textmm.delete(0,2)
        textdd.delete(0,2)
        textsimei.delete(0,20)
        lbgakureki.selection_clear(0,len(gakurekiitem))
        radiodanjyo.setvar('danjyo',danjyoitem[0])
        for i in range(len(syumiitem)):
            syumi='syumi'+str(i)
            checksyumi.setvar(syumi,False)
        textsyasin.delete(0,50)
        texthomepage.delete(0,50)
```

#ボタン syuryou 処理定義でプログラムを終了します。尚、IDLE からは機能せず、メニューから実行した場合プログラムを閉じて、メニューに戻ります。

```
    def syuryou_clicked():
         sys.exit()
```

以下の# bgcolor 指定から、画面の生成プログラムを記述しています。
frame1 と frame2 を定義していますが、frame1 は通常の画面、frame2 は画像表示の Canbas 画面を指定しています。

```
    # bgcolor 指定
    framebg1='darkcyan'
    framebg2='white'
    headerbg='mintcream'
    labelbg='Lavender'
    buttonbg='lightblue'
    # Frame
    frame1 = ttk.Frame(self,bg=framebg1,width=900,height=400,padx=40, pady=20)
    frame1.pack(fill=ttk.X)
    # Frame2
    frame2 = ttk.Frame(self,bg=framebg2,width=900,height=400)
    frame2.pack(fill=ttk.X)
```

行 1 では labeltaitol=ttk.Label で固定項目を labeltaitol.place(x=92,y=-4)の座標に表示します。

```
    #行 1
    labeltaitol=ttk.Label(frame1,text='＊＊　面積計算　＊＊',\
                background=headerbg,font=("",16))
    labeltaitol.place(x=92,y=-4)
```

# 行 3 では texttate=ttk.Entry は縦入力項目を texttate.place(x=92,y=72)の座標に配置します。
tate=StringVar()で入力値を文字変数にうけます。
尚、入力値を取り出すには tate.get()命令で取り出しできます。
texttate.bind('<Key-Return>', texttate_enter)は当項目でエンターキーをクリックした場合、texttate_enter 関数を呼出ます。

```
    #行 3
```

```
    labeltate=ttk.Label(frame1,text='縦長ｃｍ',background=labelbg,font=("",16))
    labeltate.place(x=0,y=72)
    tate=StringVar()
    texttate=ttk.Entry(frame1,width=2,textvariable=tate,font=("",16))
    texttate.place(x=92,y=72)
    texttate.bind('<Key-Return>', texttate_enter)
```

行 9では buttonjikou=ttk.Button は実行ボタンを縦入力項目を
buttonjikou.place(x=92,y=300)の座標に配置します。
buttonjikou.bind('<Key-Return>', buttonjikou_enter)で実行ボタンをクリックした場合、buttonjikou_enter を呼び出します。

```
  #行 9;
  buttonjikou=ttk.Button(frame1,text='実行',background=buttonbg,/
              font=("",16),command=jikou_clicked)
  buttonjikou.place(x=92,y=300)
  buttonjikou.bind('<Key-Return>', buttonjikou_enter)
```

#キャンバスでは、キャンバス画面を表示します。

```
  #キャンバス
  canvas=ttk.Canvas(frame2,bg=framebg2,width=780,height=380)
  canvas.pack()
```

#syokisyori では、sys.argv で前プログラムから渡ってきた変数を syokisyori(args)に渡し、当プログラムの処理を実行します。

```
  #syokisyori
   args = sys.argv
   syokisyori(args)
```

以下の記述は、tkinter での慣用的記述と思っていただいて結構です。

```
  if __name__ == '__main__':
      root = ttk.Tk()
      root.title('面積計算')
      app=MyApp(master=root)
      root.mainloop()
```

前章までのプログラムとは異なり、複雑で厄介に思われたかと思います。一から文法書を読んで上記プログラムを記述するとなると極めてハードルが高く、特に GUI 画面の場合座標値が求められその値計算は厄介です。自動生成であれば、プログラム定義要件を間違わず定義すれば、こうした煩瑣なプログラムを作成してくれます。また、慣れてくれば難解なプログラム記述も徐々になじんできます。
自動生成でのメリットは、プログラム記述構成がプログラムパタンが同一であれば同じになりますので、誰が作っても同じような記述になりプログラムの保守性が飛躍的に高くなります。テレワークが叫ばれますが、これまでプログラム開発でテレワークが進まなかった原因は、仕様変更が避けられず密な打ち合わせの必要性がありました。しかし自動生成の場合、変更箇所を修正して再度自動生成することで、後戻り作業が軽微になります。
プログラム開発こそテレワークに向いた仕事であり、家庭を持った場合、田舎の自然の中で子育てしながら仕事ができ、また極めて高い生産性、品質、保守性によりこれからの日本の高度な産業を支える誇りある仕事になるものと考えます。ぜひ、自動生成技術に

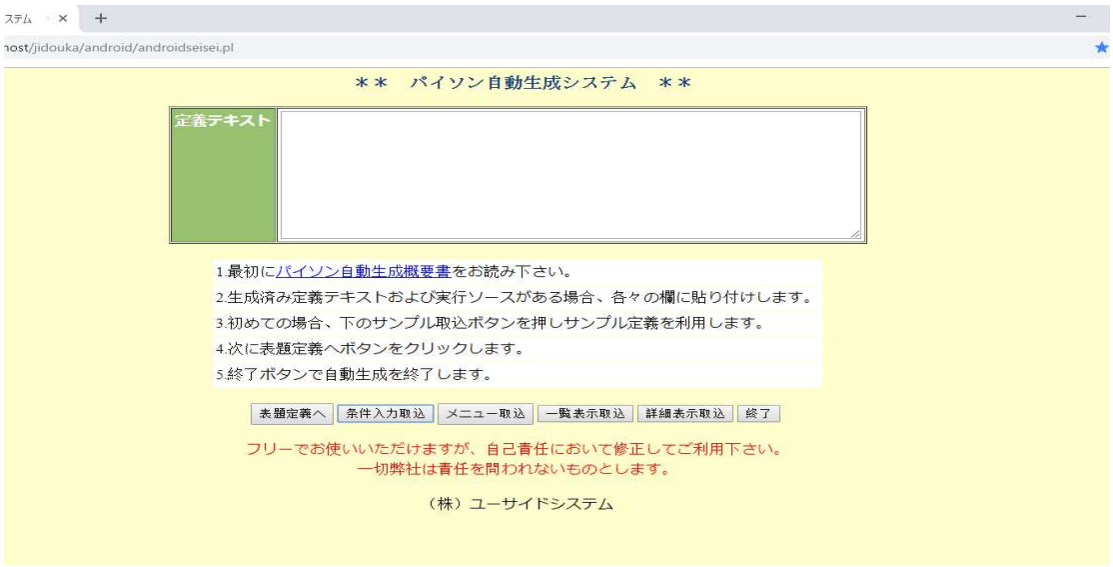
感心を持ち、自らの仕事の高率化を志向していただきたいと思います。
次よりは、主に操作説明に主眼を置くようにします。

５．自己紹介プログラム作成

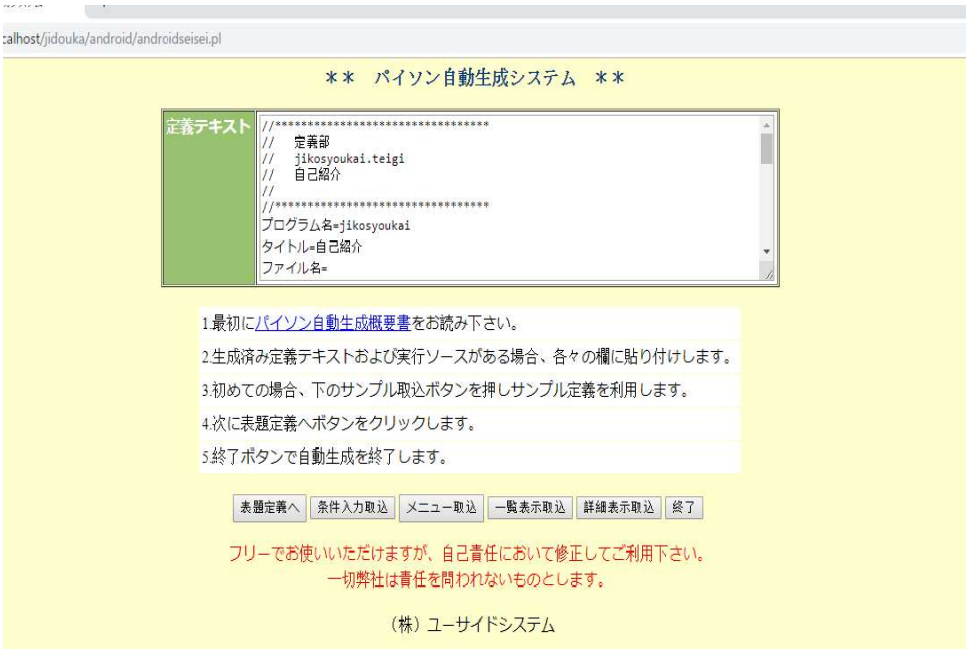
上記の面積計算では入力項目はすべてテキスト形式でした。ここでは、入力項目をいろいろな形式で構成したプログラムを自己紹介プログラムとして取り上げます。
ラベル、テキスト、リストＢＯＸ、ラジオボタン、チェックボックス、ボタン、画像の画面生成に必要な各種部品を使っており、GUI 画面生成機能が理解できます。
また、年齢計算のところで、プログラム記述があり、初めてにしては少し難しいかも知れませんが、よく読めば理解できると思います。

①パイソン自動生成システムの実行

デスクトップに貼り付けた pythonwebseisei のショートカットをクリックして、パイソン自動生成画面を表示し、条件入力定義ボタンをクリックします。



あらかじめプログラムに用意されている自己紹介プログラム定義が、定義テキスト欄に表示されます。



そのうえで表題定義へボタンをクリックします。

②表題定義

表題定義では、プログラム全体の定義をします。

定義内容をとして、

プログラム名：jikosyoukai　　半角ローマ字で指定します。

タイトル：自己紹介　　全角漢字で指定します。

パターン：入力条件

画面サイズ：900*400

作成行数：19　　項目数に応じて、最大の行数を指定します。
ただし、次の詳細定義で行数がたりなくなっても追加は可能です。

文字ポイント：16　　文字ポイント１６で固定です。

背景色：frame1&darkcyan,header&lightcyan,button&skyblue

背景色を空白にしますと、システムで用意している背景色が設定されます。
上記指定では、フレーム１画面はダークシアン
ヘッダータイトルはライトシアン
ボタンはスカイブルーの色づけされます。
Frame1&***,header&***,button&***のように指定します。
好きな色を指定してみてください。

③詳細定義

画面下の、詳細定義ボタンをクリックします。

詳細定義では、行単位でのプログラム詳細定義をします。

＊＊　パイソン詳細定義　＊＊

プログラム名 jikosyoukai　タイトル 自己紹介　ログインチェック しない

実行ソース

```
%def
#必ず def jikkou(): の記述で始めます。
#実行のまとまりは、インデント（字下げ）で行うのが、パイソンの基本です。
#文法的エラーがある場合、実行時に赤字でエラー箇所が生じされます。
#半角項目名が項目名になりますが、その項目に入力された値は、var項目名で受け取ります。
```

no	加除	行列	項目属性	項目タイトル	項目名	選択肢漢字	選択半角or初期値	項目区分	半角桁数	必須区分	onchange
1		11	ラベル	＊＊　自己紹介　＊	taitol						
2		21	ラベル	生年月日	seinengapi						
3		22	ラジオボタン	年号	nengou	大正/昭和/平成	taisyou/syouwa/heisei	文字		必須yes	
4		23	テキスト	年	yy			符号無整数	2	必須yes	
5		24	テキスト	月	mm			月	2	必須yes	
6		25	テキスト	日	dd			日	2	必須yes	
7		31	テキスト	氏名	simei			全角文字	20	必須yes	
8		41	リスト	学歴	gakureki	/中学/高校/大学/大学院	/tyugaku/koukou/daigaku			必須no	
9		51	ラジオボタン	男女	danjyo	男/女	otoko/onna			必須yes	
10		61	チェックボックス	趣味	syumi	音楽/読書/スポーツ/旅行	ongaku/dokusyo/suport/r			必須no	
11		71	画像	自己写真	syasin			文字	50	必須yes	
12		81	テキスト	ホームページ	homepage		http://www.webrobo.jp	ＵＲＬ	50	出力項目	
13		91	ボタン	実行	jikou						
14		92	ボタン	取消	torikesi						
15		93	ボタン	終了	syuryou						
16											
17											

サンプルの自己紹介プログラムでは、生年月日、氏名、学歴、男女、趣味、自己写真を入力し、入力された内容でキャンバス欄に自己紹介文と写真を表示します。
なお、入力条件項目でホームページの欄がありますが、ここは出力項目で定義されていて、上記の定義のままでは弊社ホームページアドレスがセットされます。
詳細定義画面の一番上の右端にログインチェック欄がありますが、ログインチェックするかしないかの指定ができます。プログラム公開時点には、ログインチェックをするにして、ログインしないプログラムの実行を禁止する必要があります。
上記のパイソン詳細定義画面の下段にある生成ボタンをクリックしますと、生成プログラムが実行され、プログラム定義とプログラムソースが生成されます。

ド生成ソース表示

localhost/jidouka/android/androidseisei3.pl

アンドロイド生成ソース表示

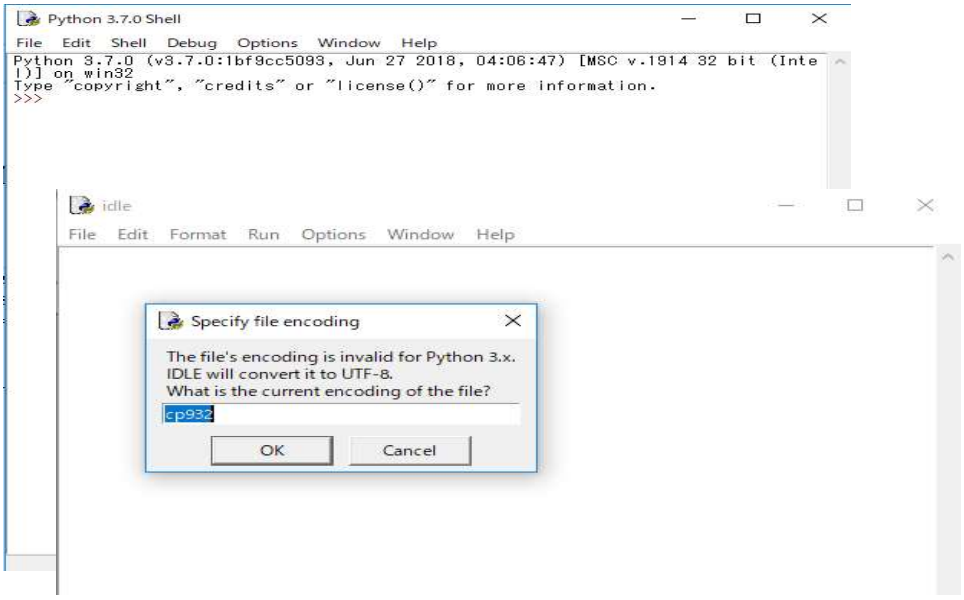
最初へ　終了

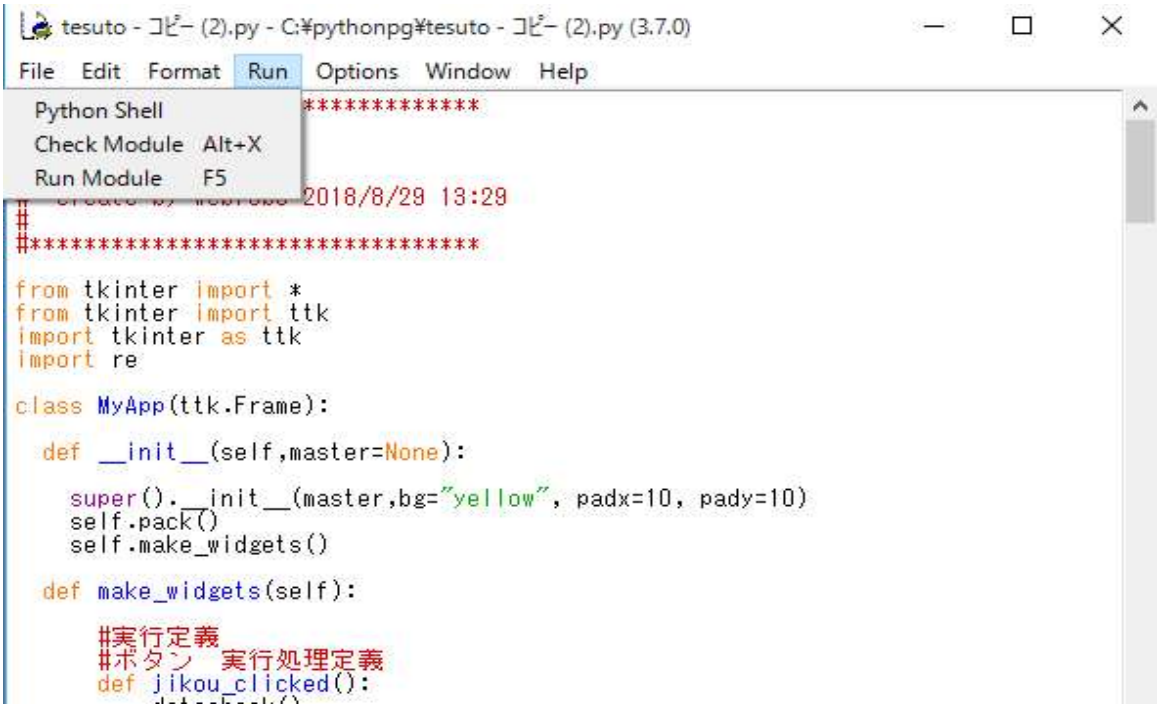
```
//**********************************
//    定義部
//    jikosyoukai.teigi
//    自己紹介
//   create by webrobo 2018/9/29 16:22
//
//**********************************
プログラム名=jikosyoukai
タイトル=自己紹介
ファイル名=
読込項目ＮＯ=
画面サイズ=800x500
ページ行数=
パターン=1
作成行数=19
文字ポイント=16
背景色=frame1&darkcyan,header&lightcyan,button&skyblue
@11@1@＊＊　自己紹介　＊＊@taitol@@@@@
@21@1@生年月日@seinengapi@@@@@
@22@4@年号@nengou@大正/昭和/平成@taisyou/syouwa/heisei@1@@2
@23@2@年@yy@@@2@2@2
@24@2@月@mm@@@@2@2@2
@25@2@日@dd@@@@7@2@2
@31@2@氏名@simei@@@20@20@2
@41@3@学歴@gakureki@/中学/高校/大学/大学院@/tyugaku/koukou/daigaku/daigakuin@@@1
@51@4@男女@danjyo@男/女@otoko/onna@@@2
@61@5@趣味@syumi@音楽/読書/スポーツ/旅行@ongaku/dokusyo/suport/ryokou@@@1
@71@7@自己写真@syasin@@@1@30@2
@81@2@ホームページ@homepage@@http://www.webrobo.jp@12@30@2
@91@6@実行@jikou@@@@@
@92@6@取消@torikesi@@@@@
@93@6@終了@syuryou@@@@@
#実行関数
#必ず def jikkou(): の記述で始めます。
```

生成表示されたソースは、プログラム定義とプログラムソースに分けてエディターにコピーペーストし、自分のパソコンフォルダーｇ:¥Python38¥kaihatu¥teigi に jikosyoukai.teigi、ｇ:¥Python38¥kaihatu に jikosyoukai.py を保存します。
すでにある場合は、書き直しします。
利用エディターによってソースの文字コードを指定できる場合は、ＵＴＦ－８を指定します。尚、保存に際して、定義体の拡張子は.text ないし.teigi 等で結構ですが、プログラムソースは.py がパイソンプログラムの拡張子になります。

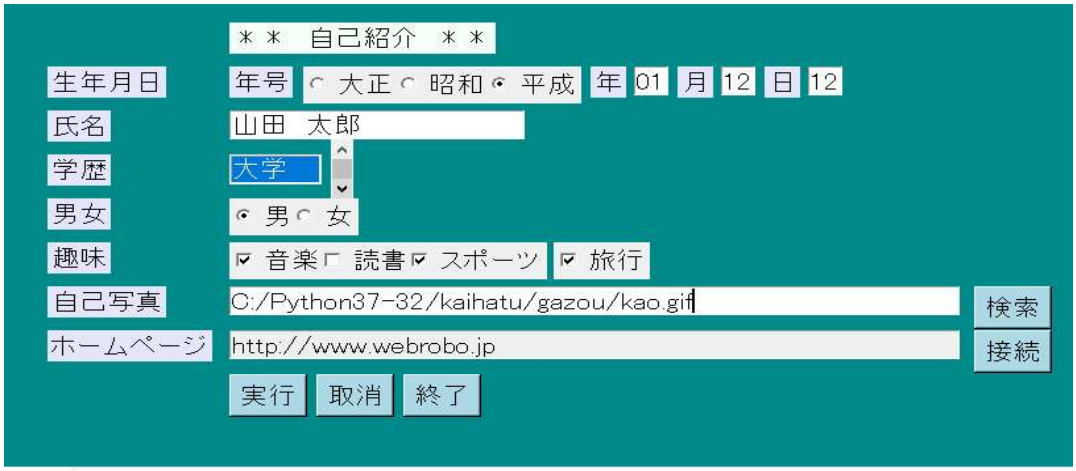
④パイソンプログラム実行

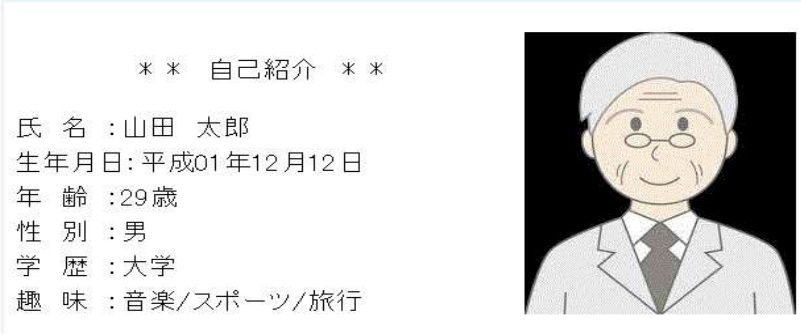
IDLE を起動します。
File から open を選択し、生成したソースｇ:¥Python38¥kaihatu¥jikosyoukai.py を開きます。
メモ帳に生成ソースを貼り付けて生成ソースを保存した場合、ソースの文字コードはＳＨＩＦＴ－ＪＩＳになります。その場合下図のように、Specify File encoding のウィンドウが表示されますので OK をクリックします。
自動生成後にエディターで保存したソースが、UTF-8 に自動変換され表示されます。





Specify File encoding が出てコード変換した場合は、上図のようにFile からSave を選択し、自動変換したソースを保存します。
次にRun をクリックし、Run Module クリックしプログラムを実行します。
今回のGUI プログラムでは、テキスト項目で値を入力後エンターキーで項目移動が可能です。更に、入力の値によってカーソルの移動先を変えて操作性をアップするプログラム変更は容易です。
自己写真の項目で検索ボタンをクリックしますと画像ファイルを選択してくださいのメセージがでますので、OK を押して画像の入っているフォルダ
ｇ:¥Python38¥kaihatu¥gazou から kao.gif をクリックします。
また、ホームページの欄で接続ボタンをクリックしますと、インターネットに接続された環境では、ホームページの値のホームページが表示されます。
実行ボタンをクリックすると、入力された値に問題なけれ、下図のようにキャンバスに自己紹介内容が表示されます。問題があれば、エラー表示されますので入力し直します。
尚、学歴のプルダウンはレイアウト編集上１行にしている関係で、プルダウンに表示された学歴では選択したことになりません。クリックすることで青色に変わり、それで選択したことになります。
消去ボタンをクリッしますとキャンバスに表示された内容は消去されます。
いろいろ入力して、試してください。
終了ボタンはIDLE から実行した場合、終了しません。右上の×で閉じます。

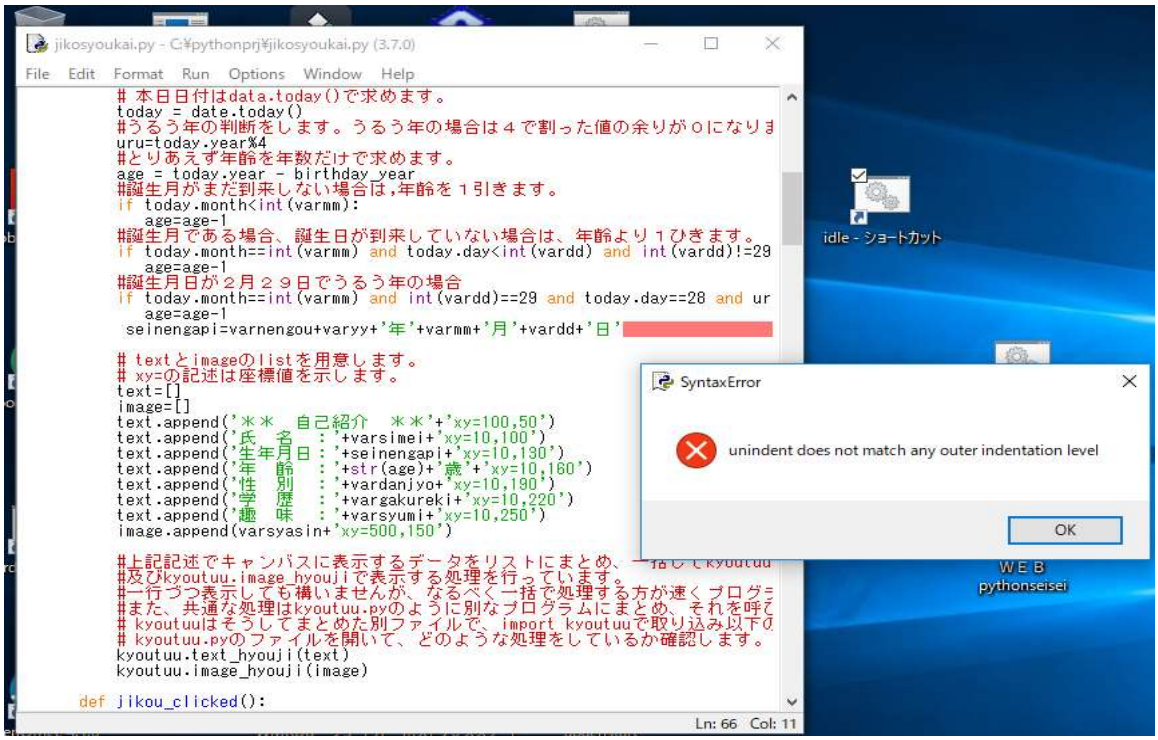




⑤デバック作業について

上記のプログラムが生成されるまでには何度もエラーが出て、その都度エラー内容を解析し、エラーを修正してプログラムを生成しています。自動生成と言えども、定義に間違いがあったり、インデントが適切でなくてエラーになる場合が多々あります。
プログラム開発においては、このデバック作業に多くの時間を割くことになります。
パイソン言語の良い点は、インタープリター言語でプログラムを記述すれば、即実行できることです。Java 等のコンパイラー言語では、まずコンパイルして文法エラーを取って、初めて実行することができます。修正の都度、コンパイル作業を強いられます。

さて、以下の画面では意図的に一文字文空白を挿入しました。インデントでの文法エラーが発生してエラーが表示されます。



実行時のエラーは、下図のように IDLE の画面に赤字でエラーが表示されます。
上図 IDLE 画面の赤字には、エラーが発生した行番号とエラー内容が表示されます。
エラーがある場合は、エラー行でプログラムの実行が停止します。
デバックには、文法的なエラーを取る作業と文法エラーはなくても、意図した動きをしないために、プログラムの動きを把握するため、実行途中での変数の動きをチェックする作業が出てきます。後者の場合は、プログラム途中にに print 文を挿入し、変数の値を上図 IDLE 画面に表示して確認します。尚、print 文は文字になりますので、数値の変数の場合は以下のように、文字化してプリントします。
print(str(varsuuti))
プログラムを自動生成しても、想定していない問題が出てきますので、このデバック作業を回避できないことはご承知ください。

⑥詳細定義の操作説明

ここでは、③詳細定義の操作の説明をします。詳細定義を少し変えて別のプログラムを作る場合は参照してください。

加除	この欄の下矢印をクリックしますと、挿入、削除の選択肢がでます。 上に行を挿入する場合は挿入を、削除する場合は削除をクリックします。
行列	画面の上から順に定義しますが、ひと桁目は行番号を、ふた桁目は列番号を意味します。行が２桁になる場合は、上２桁が行、下１桁が列になります。
項目属性	項目の属性を選択します。 属性として、ラベル、テキスト、リスト、ラジオボタン、チェックボックス、ボタン、画像の選択肢より指定します。
項目ﾀｲﾄﾙ	項目のタイトルを全角漢字で入力します。
項目名	ローマ字で入力します。 この指定値が、プログラム上の項目名として生成されます。 また、当項目で入力された値は、var+項目名で取り出し出来ます。 ただし、値は基本的に文字として扱われていますので、数値として扱う場

合は、int(varyy)のように数値化する必要があります。
選択肢漢字　画面に表示される選択肢を全角漢字で指定します。選択肢と選択肢の間の区切り文字は半角/で指定します。
空値を選択肢に含めたい場合は、自己紹介サンプルの学歴欄のように半角/ からはじめ、/中学/高校/大学/大学院のように指定します。
選択半角 or　選択肢の場合は、半角ローマ字がプログラムに使用されます。
初期値　テキスト項目で、予め初期値で値を設定したい場合、ここに指定した値が設定されます。
項目区分　入力される値がどのような項目なのかを指定します。
この指定により、入力された値が項目区分に合致するかどうかを自動的にチェックし、合致しないときエラーメッセージを表示します。
下矢印の選択肢を表示しますと、以下の選択肢が表示されます。
文字、符号無整数、符号付整数、符号少数数値、年月日、月、日、時刻、郵便番号、携帯含電話番号、携帯のみ番号、ホームページアドレス、メールアドレス、パスワード、英のみ文字、英数のみ文字、全角カタカナ、半角カタカナ、全角ひらがな、全角文字
半角桁数　項目属性がテキストないし画像の場合、桁数を入力します。
漢字項目の場合、１文字を２桁とします。
この値によって、画面でのレイアウト配置をするとともに、入力値の桁数チェックをします。
必須区分　ラベル、ボタン項目以外の項目で、空値を許すかどうかの指定をします。

詳細定義の１行目には、必ずプログラムのタイトルをラベルで指定します。
行番号を連番でなく登録した場合は、次の項目は開けただけの行間で項目配置されます。
入力した値の項目区分による妥当性チェックは、実行ボタンをクリック時に実行されます。
実行ソースの欄には、実行ボタンをクリックしたとき、どのような処理をするかのプログラムソースを記述します。表題定義時に、定義テキストに定義ファイルを貼り付け、その定義ファイルに実行関数の処理記述がある場合は、実行ソースが表示されます。
実行ソースの記述ルールは以下の通りです。
最初の一行目は、%def か%global か%include の%記述ではじまります。
#はコメント行で、説明文を入れたい場合は#の後に記述します。
入力条件パターンの場合、実行ボタンをクリックした時の処理を記述する def jikkou():は必須です。def jikkou()は左端から記述し、空白は入れないようにします。
実行、取消、前進、後退、終了、接続、詳細ボタン以外のボタンを用意した場合は、そのボタンに対応する関数を定義する必要があります。その関数名は、後退ボタンではボタンのプログラム定義では,command=koutai_clicked で関数を呼び出しします。新たなボタンで項目名を sinki とした場合、def sinki_clicked():　の関数処理記述する必要があります。
次に、パイソンでは、プログラム全体で値を保持するグローバル変数があります。下図では、import　文と class MyApp の間に変数が定義されています。
ここで、記述された変数は、グローバル変数として扱われ、各関数処理のなかで例えば global offset として記述することで、グローバル変数の offset 変数を使うことができます。global を指定しない場合は、その関数の中だけのローカル変数と見なされます。

```
from tkinter import *
from tkinter import ttk
import tkinter as ttk
import re
import os, tkinter, tkinter.filedialog, tkinter.messagebox
import webbrowser
from datetime import date, timedelta
import subprocess
import os,csv
import kyoutuu

offset=1
startno=1
readover=0
endno=9999
pagegyo=3


class MyApp(ttk.Frame):

  def __init__(self,master=None):
```

自動生成プログラムでは、自動的にグローバル変数を定義するようにしていますが、自動では対応できない場合は、実行ソースの中で%global と記述し、その下に変数を記述します。例えば、

```
%global
  hensu1=''
  hensu2=''
  hensu3_list=[]
```

のようにします

⑦自己紹介プログラムの実行関数説明

以下に自己紹介プログラムの実行関数を記述し、詳細説明をします。

```
%def
 #実行ボタン処理定義
  def jikkou():
      #年号より西暦年に変換します。
      if varnengou=='大正':
        birthday_year=1911+int(varyy)
      if varnengou=='昭和':
        birthday_year=1925+int(varyy)
      if varnengou=='平成':
        birthday_year=1988+int(varyy)
      #本日日付は data.today()で求めます。
      today= date.today()
      #うるう年の判断をします。うるう年の場合は４で割った値の余りが０になります。
      #%は余りを求めます。
      uru=today.year%4
      #とりあえず年齢を年数だけで求めます。
      age = today.year - birthday_year
      #誕生月がまだ到来しない場合は,年齢を１引きます。
      if today.month<int(varmm):
        age=age-1
      #誕生月である場合、誕生日が到来していない場合は、年齢より１ひきます。
      if today.month==int(varmm) and today.day<int(vardd) and int(vardd)!=29:
        age=age-1
      #誕生月日が２月２９日でうるう年の場合
      if today.month==int(varmm) and int(vardd)==29 and today.day==28 and uru==0:
        age=age-1
```

```
seinengapi=varnengou+varyy+'年'+varmm+'月'+vardd+'日'

#xy=の記述は座標値を示します。
text=[]          #text 文用のｌｉｓｔを用意します。
image=[]         #image 用のｌｉｓｔを用意します。
text.append('＊＊　自己紹介　＊＊'+'xy=100,50')
text.append('氏　名　: '+varsimei+'xy=10,100')
text.append('生年月日 : '+seinengapi+'xy=10,130')
text.append('年　齢　: '+str(age)+'歳'+'xy=10,160')
text.append('性　別　: '+vardanjyo+'xy=10,190')
text.append('学　歴　: '+vargakureki+'xy=10,220')
text.append('趣　味　: '+varsyumi+'xy=10,250')
image.append(varsyasin+'xy=500,150')
```

上記記述ではキャンバスに表示するデータをリストと呼ぶ配列にまとめ、一括して text_hyouji 及び.image_hyouji で表示処理を行っています。
一行づつ表示しても構いませんが、なるべく一括で処理する方が速く、プログラムがすっきりします。

６．ログインプログラム生成

単体プログラムではなく、システムとして特定のユーザーに対してプログラムを提供する場合、ログインプログラムを用意することになります。ここでは、ログイログラムを作成してみます。

①パイソン自動生成システムの実行

パイソン自動生成システムを立ち上げ、g:¥Python38¥kaihatu¥teigi に入っている login.teigi ファイルをエディターで開きます。エディターに表示される定義体すべてを選択コピーし、パイソン自動生成システムの定義テキスト欄に貼り付け表題定義へボタンをクリックします。

②表題定義

ログイン生成のパターンは、条件入力パターンになります。
背景色は空白での設定になっています。詳細定義ボタンをクリックします。

③詳細定義

ログインプログラムでの詳細定義では、パスワードの項目区分をパスワード指定をしています。この場合、入力時に入力した値は＊＊表示されます。
レイアウト上、行番号を１行飛びに指定しています。
条件入力パターンでは実行関数の定義が必要ですが、ここでは入力したＩＤとパスワードが用意された値とマッチしない場合、値と合致しないメッセージを出して、次のメニュープログラムに行かないようにしています。
合致した場合、メニュープログラムにリンクします。
実際のプログラムでは、データベースないしファイルにＩＤ，パスワードの値を持たせる場合が多いかと思いますが、今回のサンプルでは g:¥Python38¥kaihatu¥csvdata フォルダーの login.csv ファイルに値を直接持たせて、その値によりログインのチェックを行います。ＣＳＶファイルをＵＳＢメモリーに持たせ、ＩＤ、パスワードとＵＳＢメモリーの物理的セキュリティーを併用することで、よりセキュリティーを高めることが可能です。その場合、ログインプログラムがパイソンソースのままでは、セキュリティーが破られますが、パイソンにはコンパイル機能があり、ソースを隠蔽することが可能です。その方法は、メニュー作成のところで説明します。
実行ソースは、以下のように記述しています。

```
%def
    def jikkou():
        try:
            with open('g:\Python38\kaihatu\csvdata\login.csv') as f:
            reader = csv.reader(f,skipinitialspace=True)
            reader_list = [e for e in reader]
            list=reader_list[0]
            id=list[0]
            password=list[1]
            if varid==id and varpassword==password:
               cmd = "python menu.py "+"id="+id+" "+"pass="+password
```

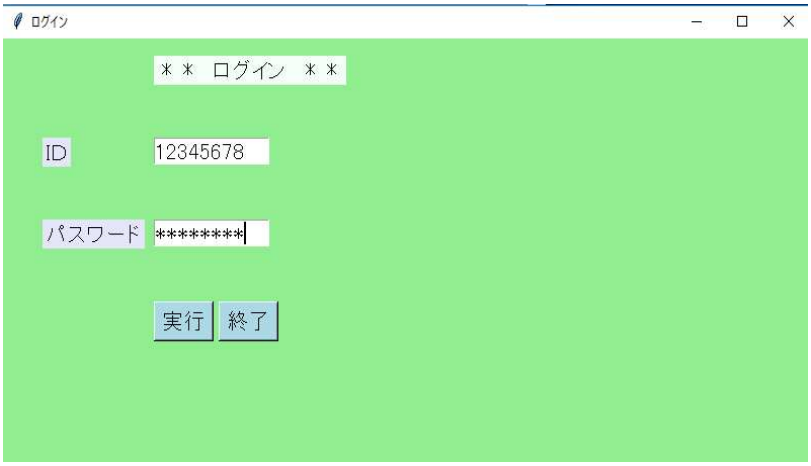
```
            pid = subprocess.call(cmd,shell=True)
            sys.exit()
        else:
            kyoutuu.message('ＩＤまたはパスワードが正しくありません！')
    except FileNotFoundError:
            kyoutuu.message('ログインファイルが読めません。')
    except csv.Error as e:
            kyoutuu.message(e)
```

生成ボタンをクリックしてプログラムを生成し、エディターで g:¥Python38¥kaihatu に login.py を、g:¥Python38¥kaihatu¥teigi に login.teigi を保存します。

④パイソンプログラム実行

デスクトップから IDLE 起動し、g:¥Python38¥kaihatu¥login.py を実行します。

実行ボタンをクリックしますと、ＩＤ、パスワードが合致すればメニューが表示されます。ＩＤに 12345678 を、パスワードに password を入力します。



７．メニュープログラム生成

当システムでは、容易にメニューができます。

自作プログラムのほか、利用がフリーな他者開発プログラム、開発ツールとしての各種バッチファイルをメニューに組み込んで利用できます。

①パイソン自動生成システムの実行

パイソン自動生成システムの画面からメニュー定義ボタンをクリックし、表題定義へボタンをクリックします。下図の表題定義画面が表示されます。

次に、詳細定義ボタンをクリックします。

②詳細定義

詳細定義画面では、以下が表示されます。

パイソンメニュー詳細定義

プログラム名	menu
タイトル	メニュー
ログインチェック	しない

no	加除	行列	項目タイトル	項目名	リンク先
1		11	自己開発	jikokaihatu	
2		12	他者開発	tasyakaihatu	
3		13	開発ツール	tool	
4		21	自己紹介	jikosyoukai	python jikosyoukai.py
5		22	お絵かき	oekaki	python oekaki.py
6		23	Ｐｙｔｈｏｎ自動生成	jidouseisei	C:/Python37-32/kaihatu/bat/pythonwebseisei.bat
7		31	会員一覧	hanisitei	python hanisitei.py
8		32	スライドショー	slide	python slide2.py
9		33	ＩＤＬＥ実行	idle	C:/Python37-32/kaihatu/bat/idle.bat
10		41	会員登録	kaiintouroku	python kaiintouroku.py
11		43	Ｐｙｔｈｏｎ翻訳	compile	C:/Python37-32/kaihatu/bat/compilepy.bat
12		51	有益サイト	yuekisite	python yuekisite.py
13		61	計算問題	keisanmondai	python keisanmondai.py
14		71	郵便番号一覧	yubinsitei	python yubinsitei.py
15					

メニュー定義も他の定義と同様、行列でレイアウトされます。
上記の指定では、１行目で自己開発、他者開発、開発ツールが指定されています。
３つのメニュータグが自動生成されることになります。
以下のレイアウトを参考にしてください。
最初の２桁が行、最期の１桁が列を意味します。定義画面では、行の値が 01~09 の場合に最初のゼロが表示されずに 1~9 で表示されます。

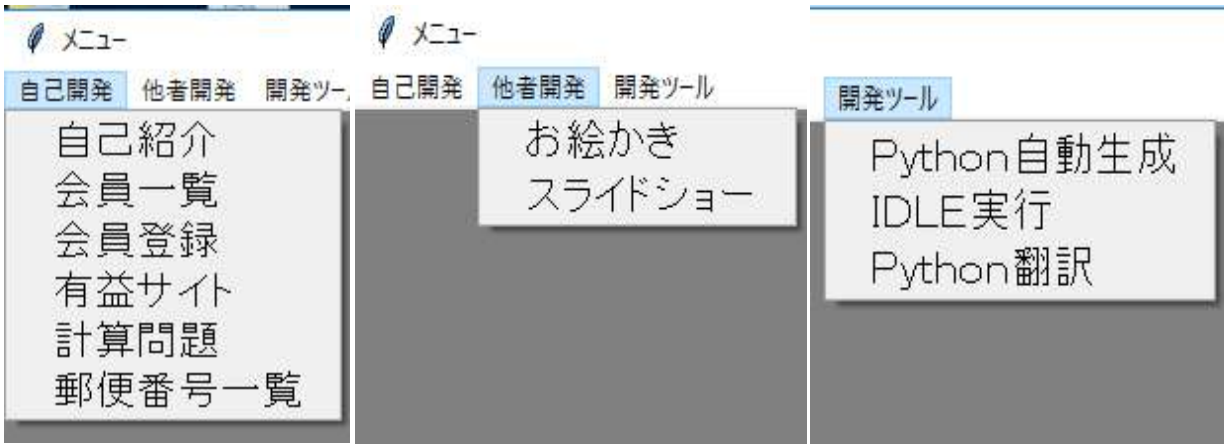
	列１	列２	列３
行 01	自己開発	他者開発	開発ツール
行 02	自己紹介	お絵描き	Python 自動生成
行 03	会員一覧	スライドショー	ＩＤＬＥ実行
行 04	会員登録		Ｐｙｔｈｏｎ翻訳
行 05	有益サイト		
行 06	郵便番号一覧		

リンク先には、実行されるプログラムを指定します。
パイソンプログラムは、python jikosyoukai.py
バッチファイルは、g:/Python38/kaihatu/bat/pythonwebseisei.bat
のように記述します。
生成ボタンをクリックしてプログラムを作成し、g:¥Python38¥kaihatu に menu.py を、g:¥Python38¥kaihatu¥teigi に menu.teigi を保存します。

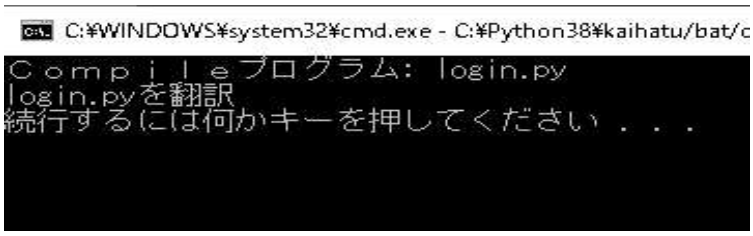
③プログラムの実行

デスクトップから IDLE を起動し g:¥Python38¥kaihatu¥login.py を実行します。
ログインしたにもかかわらずメニュー画面が表示されない場合は、メニューソースファ

イルmenu.pyがＳＨＩＦＴ－ＪＩＳコードになっていると思います。
その場合は、menu.pyoをオープンして、ＵＴＦ－８コードに変換、セーブしてあらためてログインして下さい。
また、リンク先プログラムにエラーがある場合、エラーが表示されませんので、IDLEで単独に実行してエラーを取り除きます。
メニューを実行しますと、今回作ったメニュープログラムが下図のように表示されます



自己開発メニューより、先に作った自己紹介プログラムを実行してみます。
次に、他社開発メニューをクリックして、お絵描きプログラム、スライドショーを実行してみてください。
尚、他者開発の２本は、ソシム株式会社出版の「ホンキで覚えるｐｙｔｈｏｎ」より取り込みました。当本は、パイソンのGUIに重点を置いて解説していて参考になります。
次に、開発ツールメニューよりＰｙｔｈｏｎ自動生成、IDLE実行、Ｐｙｔｈｏｎ翻訳をしてみます。開発ツールメニューより実行すれば、デスクトップに戻らずにプログラムの作成、実行も可能です。
先にログインプログラムで翻訳の話をしましたが、開発メニューのＰｙｔｈｏｎ翻訳をクリックし、Ｃｏｍｐｉｌｅプログラムでlogin.pyと入力し、続行でエンターキーをクリックして翻訳します。



g:¥Python38¥kaihatu¥__pycache__ホルダーにlogin.cpython-38.pycファイルが出来ます。ンパイルファイルは、ソース名.cpython-パイソンバージョン番号.pycで生成されます。
コンパイルファイルでプログラムを実行する場合は、このファイル名を指定します。
デスクトップに貼り付けたログインのショートカットは、翻訳ファイル
ｇ:¥Python38¥kaihatu¥__pycache__¥login.cpython-38.pycを実行するようにしています。
尚、Ｃｏｍｐｉｌｅプログラムでプログラム名を入れずエンターキーをクリックした場合は、　ｇ:¥Python38¥kaihatuに入っているプログラム全てを翻訳します。
これよりのログインは、デスクトップのショートカットをクリックします。

８．会員一覧プログラム生成

ＣＳＶファイル読んで、画面に一覧表示するプログラムを作ります。
一般的には、サーバー上にＳＱＬデータベースを置いて、そのデータベースとの読み書きをしますが、環境構築、ＳＱＬ記述の理解が必要ですので、広く知られたExcelと連携できるＣＳＶデータを読んで一覧表示するプログラムを作成します。

①パイソン自動生成システムの実行

パイソン自動生成システムの画面から一覧表示定義ボタンをクリックし、表題定義へボタンをクリックします。

②表題定義

上記定義で重要なことは、ページ行数とファイル名と読込項目ＮＯ、詳細プログラムです。ページ行数で、一画面に何行のレコード（データ数）を表示するかを指定します。
ファイル名は、読み込むＣＳＶファイル名を指定します。
読込項目ＮＯでは、画面に表示する項目がＣＳＶファイルの何番目の項目かを指定します。詳細プログラムは、一覧表示したレコードをクリックし詳細画面を表示する機能を持たせる場合その詳細プログラム名を指定します。
今回の一覧表示プログラムでは、kaiinsyousai.pyを指定していますので、詳細プログラムへのリンクボタンが生成されます。詳細定義ボタンをクリックして詳細定義に移ります。

③詳細定義

１行目には、ラベルでタイトル名を、２行目に画面に表示する項目をテキスト項目で定義します。２１から２５の５項目が一覧表示されます。３行目に前進，後退､終了ボタンで定義しています。
一覧表示パターンでは､条件入力パターンで記述した実行ソース記述は必要ありません。
作成ボタンをクリックしてプログラムを生成して、ｇ:¥Python38¥kaihatu¥に itiran.py を、ｇ:¥Python38¥kaihatu¥teigi に itiran.teigi を保存します。

④ＣＳＶファイルの確認

今回のプログラムで使用する、kaiin.csv ファイルを開いてデータ内容を確認します。
データは，ｇ:¥Python38¥kaihatu¥csvdata¥kaiin.csv です。
下図のファイルが確認されます。

⑤プログラムの実行

IDLE を起動し、会員一覧の itiran.py を開き実行します。以下の画面が表示されます。
前進、後退ボタンを押してレコードの表示状況を確認します。
今回はテストデータの用意が４件のみでしたので、行数を２行にしていますが、レコードを増やしプログラム定義の行数を増やしてテストしてみて下さい。

上図画面では、定義していない詳細ボタンが自動的に出てきました。
表題定義で、詳細表示プログラムを指定したことによります。

９．会員詳細表示プログラム生成

ＣＳＶファイルを読んで、レコードの詳細を表示するプログラムを作ります。

①パイソン自動生成システムの実行

パイソン自動生成システムの画面から詳細表示定義ボタンをクリックし､表題定義へボタンをクリックします。

②表題定義

詳細表示パターンで重要な定義項目は、ファイル名と読込項目ＮＯになります。

定義内容を確認して詳細定義ボタンをクリックします。

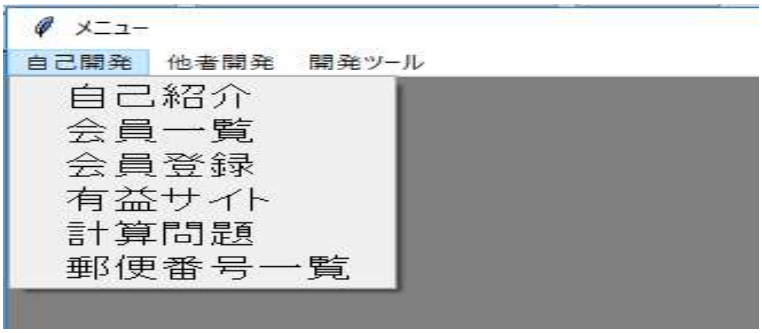
③詳細定義

表示項目はすべてテキストにして下さい。
上図定義でボタンに登録、削除、終了ボタンが定義されています。
詳細表示の機能の他、レコード修正、削除機能のあるプロラムが生成されます。
生成ボタンをクリックしてプログラムを生成し、ｇ:¥Python38¥kaihatu¥に kaiinsyousai.py を、ｇ:¥Python38¥kaihatu¥teigi に kaiinsyousai.teigi を保存します。

④プログラムの実行

今回のプログラム実行は login ショートカットより実行してみます。

ＩＤが12345678、パスワードは　password　です。
ログイン画面の実行ボタンをクリックすると下図のメニューが表示されます。
自己開発メニューの会員一覧を実行します。



会員一覧表示後、詳細ボタンをクリックしますと、指定レコードの詳細が下図のように表示されます。

登録、削除ボタンが表示され、クリックすると修正ないし削除ができます。
ＣＳＶファイルのデータ件数が少ない場合には現在のプログラムで問題はありませんが、データ件数が多い場合には該当レコードにたどり着くのが困難になります。
その解決策として、一覧表示の閲覧範囲を指定出来るようにしたいものです。
そこで、一覧表示の前に、閲覧範囲を指定するプログラムを用意することにします。

１０．会員閲覧範囲指定プログラムの生成

①パイソン自動生成システムの実行

エディターでｇ:¥Python38¥kaihatu¥teigi の hanisitei.teigi を開き、定義をすべて選択、コピーし、パイソン自動生成システムの定義テキスト欄に貼り付けます。
その上で、表題定義へボタンをクリックします。

②表題定義

定義内容を確認して、詳細定義ボタンをクリックします。

③詳細定義

開始番号と終了番号、２つの入力項目がテキストで定義されています。
必須ｎｏの指定になっています。
実行ソースは以下の通りです。

```
%def
 def jikkou():
     global varstartno
     global varendno
     varstartno=startno.get()
     varendno=endno.get()
     if varstartno=='':
        varstartno='0'
     if varendno=='':
```

```
        varendno='9999'
    if int(varstartno)>int(varendno):
        message("開始番号が終了番号より大きい")
    else:
    # 次の一覧閲覧プログラムに以下のように閲覧範囲条件を args 変数にセ
      ット
        para='start='+varstartno+'&end='+varendno
        cmd = ' python itiran.py '+para
        pid = subprocess.Popen(cmd,shell=True)
```

上記記述では、開始番号が入力されない場合は 0 をセット
終了番号が入力されない場合は 9999 がセットされています。
para='start='+varstartno+'/end='+varendno で次のプログラム itiran.py に開始番号と終了番号の値を渡す設定しています。生成ボタンをクリックして、
g:¥Python38¥kaihatu¥に hanisitei.py を g:¥Python38¥kaihatu¥teigi に hanisitei.teigi を保存します。
デスクトップから IDLE を起動し、hanisitei.py を開き単体実行します。
開始番号に 2、終了番号に 111 を入力し実行ボタンをクリックします。
下図のように、開始番号で 2 を指定していますので、レコード 1 はスキップされました。

次に、メニューを変更して、会員一覧を範囲指定に変更し、範囲指定、会員一覧会員詳細の順に実行するように変えてみます。

パイソン自動生成システムの画面からメニュー定義ボタン、表題定義へボタン、詳細定義へボタンの順にクリックします。
メニューの詳細定義の７行目のリンク先を python hanisitei.py に変えてプログラム生成保存します。
デスクトップから IDLE を起動し、menu.py を開き UTF-8 へのコード変換、save を実行します。デスクトップの login ショートカットより実行して範囲指定した条件で会員一覧表示し、更に会員詳細表示することを確認します。

１１．会員登録プログラムの生成

会員一覧、会員詳細閲覧、会員修正、削除まで上記で作成してきました。
ここまであれば、会員登録もあればと思われるかと思います。
ここに、会員登録プログラムも用意しました。

①パイソン自動生成システムの実行

パイソン自動生成システムの画面からデータ登録定義ボタンをクリックし、定義体を取り込み表題定義へボタンをクリックします。表題定義画面で、表題定義内容を確認し、詳細定義ボタンをクリックします。

②詳細定義

no	加除	行列	項目属性	項目タイトル	項目名	選択肢漢字	選択半角or初期値	項目区分	半角桁数	必須区分
1		11	ラベル	＊＊　会員登録　＊	taitol					
2		31	テキスト	年	yy			符号無整数	4	必須yes
3		41	テキスト	月	mm			月	2	必須yes
4		51	テキスト	日	dd			日	2	必須yes
5		61	テキスト	氏名	simei			全角文字	20	必須yes
6		91	ボタン	登録	touroku					
7		92	ボタン	終了	syuryou					
8										
9										
10										
11										
12										

上記内容を確認し生成ボタンをクリックして、 g:¥Python38¥kaihatu¥に kaiintouroku.py を、 g:¥Python38¥kaihatu¥teigi に kaiintouroku.teigi を保存します。
デスクトップの IDLE を起動し、kaiintouroku.py を開き UTF-8 への変換して save を実行します。
尚、メニューにすでに登録していますので、ログインより実行してみます。

自己開発メニューの会員登録を選択しますと上図画面が表示されます。
上記でデータを入力して、登録ボタンをクリックしますとデータがＣＳＶにそれまでのレコード番号に１追加した番号で登録されます。
一覧閲覧プログラムで追加データを確認するとともに、Excel 等でデータを確認します。
尚、Excel は利用者も多くまたいろいろな所で利用されていますが、個人情報を含む項目のデータ処理の場合、セキュリティーでの十分な配慮が必要です。
今回のサンプルでは、ＣＳＶデータの暗号化、復号化機能をつけておりませんが、当システムでは容易に付加することが可能です。

１２．有益サイト一覧プログラムの生成

ネットには、各種教育上有益なサイトが多々あります。
そうしたサイト情報を Excel で作成し、先の一覧表示パターンにより一覧プログラムを作れば、容易にサイト情報を提供することが可能です。
サンプルとして作ってみましたのでご紹介します。

①パイソン自動生成システムの実行

エディターで g:¥Python38¥kaihatu¥teigi の yuekisite.teigi を開き、パイソン自動生成システムの画面の定義テキスト欄に貼り付けます。その上で、表題定義ボタンをクリックします。
一覧に読み込むＣＳＶファイルには、 g:¥Python38¥kaihatu¥csvdata¥yuekisite.csv を指定しています。ネット上にある適当なサイトをＣＳＶファイルにしました。出来れば、実用で使えるサイト情報を集めてＣＳＶファイルに登録していただければと思います。副教材として活用することも重要です。但し、掲載者への了解をとられた方がいいと思います。
当プログラムでは、一覧表示機能にサイトへのリンクボタンを用意するための定義が必要です。表題定義の読込項目ＮＯ欄が 1,2,3url,4 の記述になっています。3url の指定は３項目目はホームページアドレスであることを意味しています。この記述でホームページのリンクボタンとリンクアドレスを認識できます。プログラムを生成し、インターネッと接続できる環境で、ぜひお試し下さい。尚、実行したプログラムは下図のとおりです。

１３．ランダム数値での自動問題作成プログラムの生成

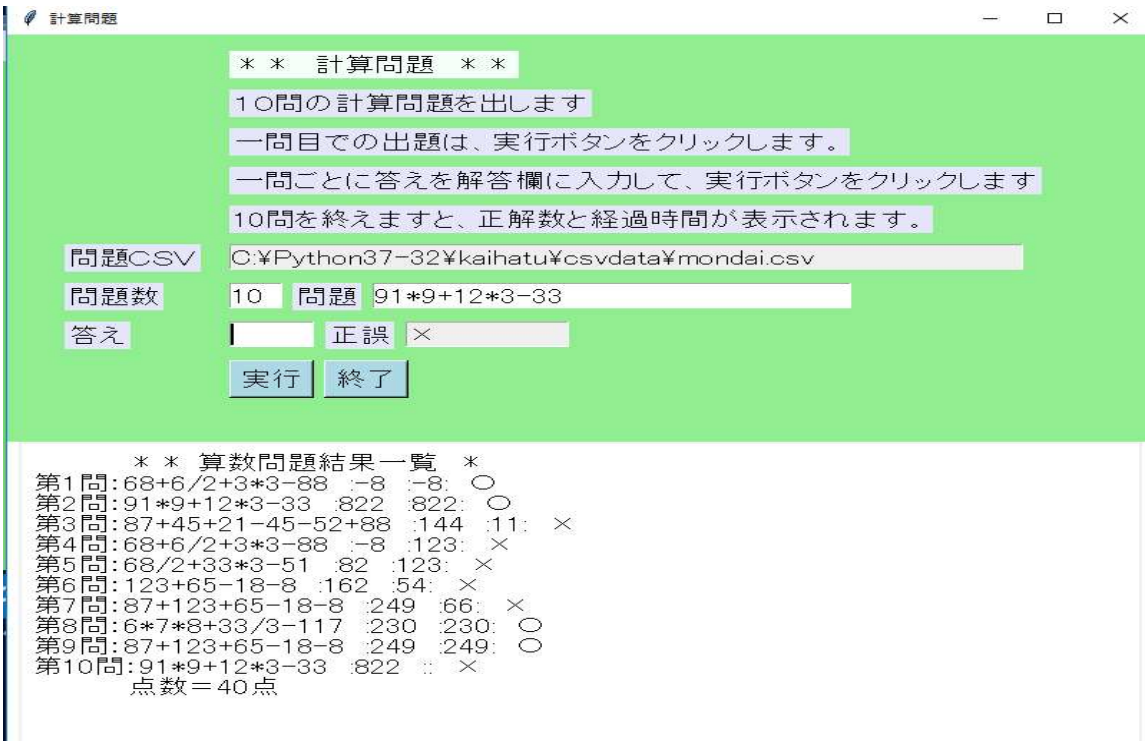
問題集を作る場合、ある範囲での値を自動的に生成して、問題を可変的に作る必要があります。例えば熟語問題の場合、熟語データに番号を付与し、存在する番号の範囲で乱数を発生させ、発生した番号の熟語を出題する必要があります。

今回の問題作成サンプルプログラムは、問題と答えをＣＳＶファイルに用意し、ランダムに画面上に問題をセットし、答えを入力してもらい、１０問繰返した上で成績を画面に表示します。

どのような教科であれ、問題と答えのＣＳＶファイルを用意すれば、ドリル出題が可能です。今回の汎用問題システムを改良し、その問題での試験データを成績管理へつなぐシステムが出来れば、先生の負担軽減が可能になると思います。また、音声認識と音声発音による語学問題作成の課題も見えてきました。

とりあえず、自己開発メニューの計算問題を実行します。

実行ボタンをクリックしますと出題され、回答を入力し実行ボタンをクリックすることで正誤を表示し、次の出題がされます。１０問が終わると成績結果が表示されます。



尚、問題のＣＳＶファイルは下図です。１項目は、番号、２項目が問題文、３項目が正解データです。

１４．郵便番号一覧プログラムの生成

これまでは少ないデータで、実用に耐えるのかと思われる方もあるかと思い、東京都の４０００件弱の郵便番号データの閲覧プログラムを作ります。
ｇ:¥Python38¥kaihatu¥teigi フォルダーに yubinitiran.teigi ファイルがあります。
この定義をパイソン自動生成システムの定義テキストに貼り付け、プログラム作成をします。そのままでは、一件目のデータからの表示になりますので、更に表示範囲指定のプログラムを、ｇ:¥Python38¥kaihatu¥teigi フォルダーの yubinsitei.teigi を定義テキストに貼り付け作成します。
閲覧範囲の開始を例えば２０００と指定した場合、２０００まで読み飛ばし、十分実用に耐えるスピードで表示されることが確認できます。

＊＊ 郵便番号一覧 ＊＊

連番	郵便番号	府県名	住所1	住所2
2000	1630426	東京都	新宿区	西新宿新宿三井ビル(26階)
2001	1630427	東京都	新宿区	西新宿新宿三井ビル(27階)
2002	1630428	東京都	新宿区	西新宿新宿三井ビル(28階)
2003	1630429	東京都	新宿区	西新宿新宿三井ビル(29階)
2004	1630430	東京都	新宿区	西新宿新宿三井ビル(30階)
2005	1630431	東京都	新宿区	西新宿新宿三井ビル(31階)
2006	1630432	東京都	新宿区	西新宿新宿三井ビル(32階)
2007	1630433	東京都	新宿区	西新宿新宿三井ビル(33階)
2008	1630434	東京都	新宿区	西新宿新宿三井ビル(34階)
2009	1630435	東京都	新宿区	西新宿新宿三井ビル(35階)
2010	1630436	東京都	新宿区	西新宿新宿三井ビル(36階)
2011	1630437	東京都	新宿区	西新宿新宿三井ビル(37階)
2012	1630438	東京都	新宿区	西新宿新宿三井ビル(38階)
2013	1630439	東京都	新宿区	西新宿新宿三井ビル(39階)
2014	1630440	東京都	新宿区	西新宿新宿三井ビル(40階)
2015	1630441	東京都	新宿区	西新宿新宿三井ビル(41階)
2016	1630442	東京都	新宿区	西新宿新宿三井ビル(42階)
2017	1630443	東京都	新宿区	西新宿新宿三井ビル(43階)

後退 前進 終了

１５．自動生成でのプログラムソースの要点

上記プログラム操作で、それなりにプログラムソースはご覧になられたと思いますが、なかなか読んでも理解できなかったと思います。
ここではぜひ知っておくべき点を解説をします。

①kyoutuu ソース

ｇ:¥Python38¥kaihatu フォルダーに kyoutuu.py というソースがあります。
当自動生成で共通に使う関数を用意しています。

hyoujyuncheck(各種入力項目値)
項目定義で桁、項目属性を指定した場合、自動的に入力値がチェックされました。この関数を使用して入力値をチェックしています。

message(msg)
エラーメッセージを表示します。

kakunin_message(msg)
確認のメッセージを表示します。

argsset(args)
引数値の配列設定

logincheck(args)
ログインしたかどうかチェックします。
標準ではログインプログラムで、id,password の値をすべてのプログラムで保持します。この値がない場合はログインエラーを表示しプログラムを終了します。
尚、自動生成時にログインチェックをしないで生成した場合は当チェックはされません。

②プログラムリンク

メニュープログラムソースを見ますと、メニューから別なパイソンプログラムを実行する場合、以下のような記述になっています。

```
def csvkaiintouroku():
    argstr=kyoutuu.argsset(args)
    cmd = 'python kaiintouroku.py'+argstr
    pid = subprocess.Popen(cmd,shell=True)
```

subprocess には call と Popen とがあり
call は終了時のコードを返す、Popen は Popen オブジェクトを返すの違いがあります。
bat ファイルの実行は以下のような記述になります。

```
cmd = 'bat\\pythonseisei.bat'
pid = subprocess.Popen(['start',cmd],shell=True)
```

プログラムリンクで考慮すべきことは、プログラム間での変数渡しがあります。
上記 logincheck では、ログイン時に id、password 等の値を次プログラム以降に引き渡しますが、更に条件検索画面で条件値を入力し、次のプログラムへその値を渡さなくてはならない場合が出てきます。こうした場合、args という特別な変数に値を渡します。
各プログラムには

```
#syokisyori
args = sys.argv
syokisyory(args)
```

の記述がありますが、この args と言う変数にプログラム間引き渡し値を設定します。
例えば、hanisitei.py では jikkou()では以下のように記述されています。

```
# 次の一覧閲覧プログラムに以下のように閲覧範囲条件を args 変数にセット
para='start='+varstartno+'/end='+varendno
cmd = ' python itiran.py '+para  #itiran.py の後１文字半角空白を入れて para を結合
pid = subprocess.Popen(cmd,shell=True)

# 次の itiran.py の def syokisyoripid では以下のような記述になっています。
def syokisyori(args):
    global id
    global password
    for i in range(len(args)):
        if args[i].find('id=')>-1:   #以下で id,password の値を受け取る
            id=args[i].replace('id=','')
        if args[i].find('pass=')>-1:
            password=args[i].replace('pass=','')
    global endno
    global offset
    global startno
    l=len(args)                          #以下で startno と endno の値を受け取る
    if args[l-1].find('start=')>-1 and args[l-1].find('/end=')>-1:
```

```
        para=args[l-1].split('/')
        para[0]=para[0].replace('start=','')
        para[1]=para[1].replace('end=','')
        startno=int(para[0])
        if startno==0:
          startno=1
        endno=int(para[1])
        if endno==0:
          endno=9999
      else:
        startno=1
        endno=9999
      offset=startno
      dataread()
```

③選択値の可変化

入力項目の選択値を上記自己紹介サンプルでは、固定値で定義していましたが、ＣＳＶデータもしくはデータベースより読んで選択値を用意したいと思うことが出てくるかと思います。ここでは、ＣＳＶより選択肢を作成するパイソンソースの関数を記述し、その記述を取り込んで自動生成します。

参考にｇ:¥Python38¥kaihatu フォルダーに listread.py を用意しました。

学歴のプルダウンを CSV ファイルを読んで作る場合、選択半角 or 初期値に listread..gakureki()と定義します。関数にパラメータを渡す場合は、listread.gakureki(abc)

のように記述します。listread には、データーベーステーブルを 読む場合の関数も記述しています。

④bat ファイルでのフォルダー名の可変化

bat ファイルの login.bat をご覧ください。以下のような記述になっています。

```
@cd ..
@cd ..
path %cd%;%cd%/Scripts
@cd kaihatu
python %cd%/__pycache__/login.cpython-38.pycwo
```

python のあるフォルダーがｃであったり、ｄであったり、またバージョンが変わります。こうした実行フォルダー名を可変化するために、%cd%を使っております。

また、例えば login.py プログラムでは、カレントデレクトリー名を使用して可変化を図っています。

```
initial_dir = os.getcwd()
def jikkou():
global initial_dir
try:
    with open(initial_dir+'/csvdata/login.csv') as f:
```

１６．最後に

最後までお付き合いいただきありがとうございました。

ところで、この入門編をまとめておるさなかにコロナウイルスが世界に蔓延しています。この問題をどのようにとらえるか、今後の世界を大きく変える歴史的出来事です。近年の頻発する自然災害は温暖化によるものと言われており、グローバル化経済によるエネルギー浪費へのコロナウイルスからの警鐘とも言えます。

一方日本では、１０年も立たない東日本大震災と原発事故、特に原発事故では汚染水

問題も解決せず、更には廃炉への技術的解決策も立たない中で、オリンピック開催による更なる東京一極集中を進めてきました。遠からずの首都直下地震リスクが迫っておりながらです。コロナウイルスは日本では、東京一極集中リスクへの警鐘と言えます。コロナの自粛対策としてテレワークが声高に叫ばれるようになり、また日本のＩＴ遅れが白日の下になり、今後ＩＴ業界は大きな変化を強いられることが予想されます。それは、これまでの東京一極集中での多重派遣による派遣先での人手に頼る開発体制です。また、経済産業省が提起している、2025 年の崖問題でのレガシーシステムに縛られ、ＩＴ予算の８割をシステム保守に割かざるを得ない状況からの早急な脱却があります。レガシーシステムから新システム移行で問われることは、東京一極集中での派遣体制での人海戦術的開発から地方分散によるテレワーク開発と自動生成技術による高生産性と高保守性による開発手法が問われます。

今回のコロナウイルスが提示した問題は、益々ネットワーク化する社会インフラとしてのコンピュータシステムが、コンピュータウイルスによるネットワーク遮断が起こった場合の社会的損害はコロナウイルスを越えることが想定されます。そうした観点からも、下請け派遣による人手任せの開発体制には空恐ろしさを感じます。５Ｇ等の益々高速になるネットワーク環境を考える時、地方分散テレワーク開発により、密におびえながらの生活から、地方での自然に囲まれた伸びやかな環境での開発体制に移行すべきです。

今回のシステムは、パイソン言語による単体システムの自動生成に限定しています。しかし、レガシーシステムの新システムへの移行、ＡＩ、ＩＯＴ等の開発を考える時、Ｗｅｂシステム、クライアントとＷｅｂの連携システムでの開発システムが要求されます。そうしたシステム要求に対して、すでに開発をすすめており有償での提供を想定しています。しからば、どのようなプログラムが開発できるのか、以下に資料を掲載していますのでお読みいただければと思います。

https://www.webrobo.jp/siryou/webrenkei.pdf

また、弊社では今回のパイソン入門編とＷｅｂ連携版を利用した地方でのＩＴ塾による人材開発の試みを進めています。

弊社の進めるパイソン自動生成システムは、開発ツールであるとともに教育ツールとして位置付けております。間違いなく文法書によるＩＴ教育よりは、サンプルの自動生成による教育はより効果的です。今後、特に新人教育にＩＴ教育は必須であるべきで、弊社自動生成システムをＩＴ教育ツールとしてもご利用いただければ幸いです。

皆様のこれからの効率的社会システムへのチャレンジを期待しています。