第 3 回　Python の基礎

Python のプログラムを実行する方法と、エラーの修正方法について学ぶ。

【Python のバージョンについて】
Python のバージョン番号は A.B.C や A.B のように付けられている。A はメジャーバージョン番号で、言語の仕様に関わる重要な変更のときに上げられる。B はマイナーバージョン番号で、それほど大きくはない変更のときに上げられる。C はマイクロレベルで、バグを修正するときに上げられる。
Python は 1991 年にバージョン 0.9 のソースコードが公表されたときにはすでにオブジェクト指向言語としての特徴を備え、1994 年にバージョン 1.0 が公開されたときには、関数型言語の基本であるラムダ計算などが組み込まれた。2000 年にバージョン 2.0 が公開され、リスト内包表記やメモリを効率的に管理するガーベージコレクションなどの機能が実装され、一躍メジャーな言語になった。2008 年には Python 3.0 がリリースされた。現在広く使われている Python のメジャーバージョンは 2 と 3 であり、それぞれ Python 2、Python 3 と書かれる。2020 年 1 月 1 日に Python 2 の開発は停止したため、この授業では Python 3 を使って学習する。

【Python のセットアップ】
Python でプログラミングをするためには、Python をインストールする必要がある。その方法は、Windows, macOS, Linux などの UNIX プラットフォームなどの OS によって異なり、詳しくは Python ドキュメントページで解説されている。

https://docs.python.org/ja/3/

基本的には、Python のページ https://www.python.org/ から環境に合わせたインストーラーをダウンロードしてインストールする。プログラムは文字コード UTF-8 のテキストファイルで作成する。
この授業では、Python のインストールはせずにオンライン実行環境を使って実習をする。

【Python の実行方法】
Python の実行方法には、何通りかの方法がある。1 つ目は、ターミナルから起動する方法である。Windows ではコマンドプロンプトまたは PowerShell を起動して py と入力する。Mac ではターミナルを起動して python と入力する。この方法では、Python が対話モード（インタラクティブモード）で起動する。最初に Python のバージョンなどが表示されて、>>> のような「プロンプト」が表示されて、コマンドの入力待ちをする状態になる。直接 Python のコマンドを入力して実行することができる。Ctrl+D を押すか exit() と入力する事で、対話モードを終了することができる。細かい説明は省略するが、複数の Python のバージョン（たとえばバージョン 2 とバージョン 3 など）がインストールされている場合には、バージョンを指定して起動する場合もある。対話モードの Python の中で現在使っているバージョンを確認するには、import sys; print(sys.version) と入力する。
統合開発環境 IDLE を起動することもできる。Windows の場合はスタートメニューから IDLE を選ぶ。Mac の場合には、ターミナルから idle と入力する。
Jupyter Notebook によって、プログラム、メモ、グラフなどの実行結果をまとめて管理する手法は、特に機械学習やデータサイエンスの分野では主流となっている。

【オンライン実行環境 Paiza】
通常は PC にプログラム開発環境をインストールしてプログラムを開発するものであるが、ウェブブラウザ上でプログラムを開発できる。たとえば、次のようなサイトがある。

(1) paiza.IO （ https://paiza.io/ja ）
日本語対応。
(2) Ideone （ https://ideone.com/ ）
対応している言語が多い。
(3) Replit （ https://replit.com/ ）
対応している言語が多く機能が豊富。Matplotlib のグラフ出力も見ることができる。
(4) Google Colaboratory （ https://colab.research.google.com/ ）
Google のクラウド上で Jupyter Notebook の Python 実行環境を利用できる。

この授業では、OS などの環境の違いを吸収するために、Paiza を使ってプログラムを編集し、実行する。

【Hello プログラム】
それでは、早速 Paiza を使って Hello プログラムを実行してみよう。

★マークの箇所は、実際にやってみること。

★ https://paiza.io/ja にアクセスする。
★ 言語が日本語になっていなければ日本語を選ぶ。

以下はログインしなくても続けることができるが、ログインすることでそれまでに自分が作成したプログラムを一覧する機能があるので、今後の効率的な学習のためには、サインアップしてログインすることを推奨する。

★ 緑色の「コード作成を試してみる（無料）」ボタンを選ぶ。
★ 左上から言語を選択する。「Python3」を選ぶ。
★ 次のプログラムを打ち込む。日本語入力はオフにすること。

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

このように四角で囲まれた箇所はプログラムをあらわす。1 行目と 2 行目には # （ハッシュ）で始まる行があらかじめ入っているが、 # で始まる行は「コメント」でありプログラムの実行に影響を与えないので、そのまま残しておいても、消しても構わない。なお、最初の行に書かれている「# coding: utf-8」は、ファイルの文字コードが UTF-8 であることを示していて、Python 2 ではこの記述が必要であったが、Python 3 では UTF-8 が標準で使われるので、この記述を省略することが可能である。

★ 「実行（Ctrl-Enter）」ボタンを押す。「出力」に「Hello!」と表示されることを確認する。「Hello!」以外の文字が表示された時、あるいは「実行時エラー」に英語のメッセージが表示されたときには、プログラムを修正する。その方法については、【エラーのデバッグ】で詳しく解説する。
★ デバッグが終わって、実行したら「Hello!」と表示されるようになったら、【★課題】で指示されている通りに課題の提出をする。

【プログラムの解説】
print() は、Python に標準で用意されている「組み込み関数」の一つである。「関数」は、数学の関数に似ている。たとえば、y = 2x という関数は、x という引数に対して y と

いう値を返す関数であり、そのような関数が定義されていれば、x=2 という引数を与えることで、y=4 という結果が得られる。このように、なんらかの値を「引数」として与えてプログラムに処理をさせて、なんらかの結果（戻り値）を返すものを「関数」と言う。ただし、「戻り値」がない場合もあり、print は戻り値がない関数である。

```
print(引数)
```

のように指定すると、引数の内容を文字列として出力する（さらに詳しい使い方については、ここでは触れない）。ここで「出力する」というのは、結果を表示するということで、この場合は Paiza の「出力」に表示されるが、どこに出力するか（たとえばファイルに出力するか）といったことを制御することも可能である。

print(引数)の「引数」のところには、"Hello!"と書かれている。ここで、”は二重引用符であり、いわゆる「半角文字」で入力する必要がある。日本語入力をオンにしていると、「“」という全角文字が入力されてしまいエラーになる。

Python では、二重引用符 “ あるいは引用符 ‘ で囲まれた文字の並びは「文字列」というデータとして解釈される。ここで Hello! を二重引用符で囲んで “Hello!” と書くことで、Hello!という文字列として解釈され、それが引数として print 関数に渡されて、Hello! という文字列が出力される。Hello! を二重引用符で囲まずに print(Hello!) として実行すると、「!」の意味を解釈できないとして、次に説明する文法エラー（SyntaxError）となる。

【エラーのデバッグ】

「デバッグ」とは、プログラムのおかしいところ（バグ）を見つけて修正する作業である。バグとは「虫」を意味し、虫取りというような意味である。プログラミングは、何度も「思い通りに動かない」ことに遭遇し、ミスを修正するという作業を繰り返す。今回のように、たった 1 行のプログラムであっても、最初はなんらかのエラーがでてうまく動かないことがある。そのときに「どこがエラーであるかを見つける」ことが、プログラミングの重要な技術である。なお、【Python の実行方法】で紹介した統合開発環境 IDLE にはデバッグ機能があり、効率良くデバッグをすることができる[1]。

ここでは、典型的なエラーを実際に発生させてみて、どのようなミスをするとどのようなエラーが表示されるかを確認する。それがわかれば、エラーの表示を見て修正する作業が容易となる。

(1) ケース 1：最後の)を全角文字にしてしまった。

Python を含む多くのプログラミング言語では、ASCII の文字[2]が使われるため（文字列を指定したりコメントを書いたりする場合を除く）、非 ASCII 文字、つまり ASCII ではない文字は「無効な文字」であるとされる。日本語入力によって変換された文字はそのような無効な文字になる。この授業では ASCII の文字を「半角文字」、日本語入力された非 ASCII 文字を「全角文字」と呼ぶ[3]。たとえば、数字には半角文字の「1234567890」と全角文字の「１２３４５６７８９０」がある。アルファベットや()”,などの記号にも対応する全角文字があ

[1] Python の IDLE でデバッグを行う方法 https://gammasoft.jp/python/debugging-python-idle/

[2] ASCII（American Standard Code for Information Interchange; アスキー）は、コンピュータで最もよく使われている文字コード体系で、7 桁の 2 進数（10 進数では 0 から 127 まで）に文字コードが割り当てられている。キーボードで日本語入力モードを ON にしないで入力される文字（123…のような数字、abc…のようなアルファベット、()のような記号）は、ASCII の文字コードであらわされる。

[3] 厳密には不正確だが、話をわかりやすくする。

る。「日本語入力はオフにすること」という指示を実行していれば全角文字が入力されることはないはずであるが、ここでは試しに、最後の)を全角文字）にして実行をしてみると、「実行時エラー」に次のように表示される。

```
  File "Main.py", line 1
    print("Hello!")
                 ^
SyntaxError: invalid character in identifier
```

このメッセージには、2 つの情報が含まれている。それは「どこにエラーがあるか」そして「どのような種類のエラーがあるか」ということである。まずは、最初の行に

```
  File "Main.py", line 1
```

とあり、これは Main.py というファイルの 1 行目にエラーがあることを示している。今回はたった 1 行のプログラムであるためエラー箇所を見つけるのは簡単であるが、プログラムの行数が増えたときには、まずどこの行でエラーが発生したのかを確認することが大切である。さらに、次の行には

```
    print("Hello! “)
                  ^
```

と表示されている。 ^ ここで「 ^ 」マークの位置が、エラーが発生している場所を示している。つまり、print 文が終わったところでエラーが発生している、ということがわかる。この位置の前後をよく見ることで、エラーを発見できる。

今回の授業では、テキストに書かれているプログラムをそのまま打ち込めば、正常に実行ができる。そうなっていないということは、必ずどこかに打ち間違いがある。単語の綴りを間違えていたり、アルファベットの大文字と小文字を間違えていたりしている。したがって、もしエラーが発生したときには、エラーの原因がわからなくても、エラーが発生した箇所をよくみて、その前後を確認する、あるいは打ち直すことで、エラーを修正することができる。さらに、次に表示されている

```
SyntaxError: invalid character in identifier
```

という表示の意味がわかれば、さらにエラーの原因がつかみやすくなる。より複雑なプログラムをデバッグするときには、エラーの箇所とともに、このエラーメッセージを確認することで、エラーの原因を突き止める手がかりとする。

エラーメッセージは英語で表示されるので、英語の意味を考えればある程度想像することができる。エラーメッセージをそのまま検索にかけることで、そのエラーメッセージの意味を解説するサイトを見つけることができる場合もある。ここでは、上記のエラーメッセージの意味について解説する。

SyntaxError は「文法エラー」つまり Python の文法に則ってないということを意味している頻出のエラーである。次に、どのような文法エラーであるかが解説されている。Identifier の正確な意味[1]についてはわからなくても、無効な文字 (invalid character) と表示されていることで、「全角文字」が使われていることが原因であると推測できる。

[1] https://docs.python.org/ja/3/reference/lexical_analysis.html#identifiers

全角と半角の間違いを目で確認するのは大変だが、エラーメッセージで「エラーが発生している場所」と「無効な文字がある」という情報が示されているので、ミスを発見できるはずである。特に、全角スペースと半角スペースは見た目で区別できないので発見が大変であり、注意が必要である。

今回は日本語の文字を使っていないため、全角文字によるエラーは発生しにくいが、次回以降の授業の課題で日本語の文字を扱うようになると、日本語モードを元に戻さないことにより、半角文字で入力するべき文字を全角文字で入力してしまうことによるエラーは頻繁に出る。実際に、毎年多くの学生がこのエラーに遭遇しているので、ここでよく理解をしておくこと。

(2) ケース 2：二重引用符を全角文字にしてしまった。

print(″Hello!″)の最後の”を、全角文字の「“」として実行すると、このようなエラーが表示される。

```
  File ″Main.py″, line 1
    print(″Hello! “)
                  ^
SyntaxError: EOL while scanning string literal
```

今度もまた文法エラーである。EOL とは、End of line、つまり行が終わっている、ということであり、文字列の値 (string literal) を読み取っている途中で、行が終わってしまった、ということが書かれている。print(″ の ″で文字列の読み取りが開始して、次の″までが文字列であると解釈するところ、次の″が見つからない、というエラーが表示されていることがわかる。最後の二重引用符が「全角文字」であるために、文字列を閉じる二重引用符であると解釈されずに、エラーとなっているわけである。このように、全角文字としたことによるエラーが必ず invalid character というエラーとして表示されるとは限らないので注意すること。

(3) ケース 3：print の綴りを間違えた。

prant(″Hello!″) として実行すると、次のようにエラーが表示される。

```
NameError: name ’prant’ is not defined
```

NameError は、名前のエラーである。次に ’prant’ という名前は定義されていない、と表示される。ここで、prant の箇所が間違えているとわかる。そして、print の綴りを prant と間違えていることがわかる。関数の名前は大文字と小文字を区別するので、たとえばｐを大文字として Print(″Hello!″)としても、

```
NameError: name ’Print’ is not defined
```

のようにエラーが表示される。

【★課題】

作成した Hello! と出力するプログラムを、ToyoNet-ACE のレポートから、次の手順で提出してください。

★ Paiza のプログラムの画面とは別のブラウザのタブでToyoNet-ACEを開き、「レポート」

から、「第 3 回 Python の基礎」を選ぶ。
★ Paiza タブの Hello プログラムの画面で、プログラムのエリアにカーソルをあわせ、ブラウザのメニューから「すべてを選択」を選んで（Windows では Ctrl + A、Mac では command + A)、さらに「コピー」を選ぶことで（Windows では Ctrl + C、Mac では command + C)、プログラムをすべてクリップコードにコピーする。
★ ToyoNet-ACE のタブに戻って、テキストエリアを右クリックして「貼り付け」をする（Windows では Ctrl + V、Mac では command + V）ことで、クリップボードにコピーされているプログラムを貼り付ける。それ以外の余計な文字（スペースなど）を入れないこと。プログラムとは関係のない文字(メッセージなど)が入力されていた場合には、その文字もプログラムに実行されてエラーとなるため、不合格点となる。#で始まるコメント行や何も入力されていない空行（スペースを入力しないこと）は実行に影響を与えないので入っていても大丈夫である。
★ 教員へのコメントを書く場合には、行を##で始めること。2 行以上のコメントを書く場合には、すべての行頭に##を入れること。
★ 「プレビュー（次へ)」ボタンを押す。
★ 「状態」に赤字で「まだ提出していません」と表示されるので、さらに「提出」ボタンを押す。
★ 「状態」に「提出済み」と表示されることを確認する。
★ 提出を確認したら帰って良い。授業時間内に提出できなかった場合には、提出期限までに提出をすること。

【課題の再提出機能】
提出期限前であれば、何度でも再提出できる。「提出取消（再提出する）」ボタンを押して、さらに「戻る」ボタンを押すことで、プログラムを編集できる状態になる。

【課題の採点】
提出されたファイルを教員のパソコンで実行して、指示されている通りの結果が得られるプログラムができているかどうかで採点する。採点結果は ToyoNet-ACE で確認できる。具体的には、以下のような基準で採点する。今後の課題でも同様である。

(1) 実行時にエラーが生じるプログラムは、不合格点となる。【エラーのデバッグ】をよく確認して、エラーが出ないプログラムを提出すること。
(2) 実行時にエラーは出ないけれど想定通りの結果とならないプログラムは、プログラムの完成度によって、不合格点（60 点未満）となるか、満点ではない合格点（60 点以上）となる。今回は Hello! という文字を出力するプログラムを作る、という課題なので、他の文字が表示されるプログラム（Hallo! のような綴りの間違いを含む）は「指示通りの結果」ではないので、満点にはならない。また、課題で例示されている通りの出力が出ない場合も減点となる。次回以降の課題で、入力時にプロンプトを出力するような処理を加えた場合も、指示とは異なるの出力が出るため減点となる。
(3) 課題で指示された通りの動きをするプログラムが提出されていれば、満点（100 点）である（ただし、円周率の計算アルゴリズムを確認するプログラムで円周率そのものを表示するといったような「課題の趣旨を理解していないプログラム」は、不合格点となる)。「指示通りの動きをするプログラム」を作ってから提出すること。今回の課題は、書かれている通りに打ち込むだけで、指示通りの動きとなる。
(4) 未提出は 0 点であり、授業欠席となる。
(5) 提出期限（ToyoNet-ACE のレポート提出画面を確認すること）を過ぎたら、いかなる理由があっても提出・再提出は不可である。PC やネットの不調などで提出できなかった場合も同様とするため、提出期限ぎりぎりで提出しようとしないこと。
(6) 成績は 15 回の課題の平均点で決まる。