Operacje wejścia i wyjścia w języku Python

Źródło: Christina Morillo, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Jak napisać, program, który będzie komunikował się z użytkownikiem? Służą do tego poznane już przez nas operacje wejścia i wyjścia. Pozwalają one na dwustronną komunikację użytkownika z programem. Obejmują wszystkie czynności, od odczytywania przycisków z klawiatury do wyświetlania filmów na ekranie.

W tym e-materiale zapoznamy się z operacjami wejścia i wyjścia w języku Python.

Charakterystykę operacji wejścia i wyjścia w innych językach programowania znajdziesz w e-materiałach:

- Operacje wejścia i wyjścia w języku C++,
- Operacje wejścia i wyjścia w języku Java.

> **Twoje cele**
>
> - Scharakteryzujesz operacje wejścia/wyjścia w języku Python.
> - Dokonasz zmiany typu danych.
> - Sformatujesz wyniki.
> - Użyjesz modułów `PySimpleGUI` oraz `Tkinter` do tworzenia prostych okien dialogowych.

Przeczytaj

Standardowe funkcje wejścia/wyjścia

Jak już wiesz, podczas pracy z programem użytkownik może wprowadzać dane przez standardowe wejście – klawiaturę. W języku Python wszystkie dane zwracane przez funkcję `input()` są typu `string`. W sytuacji, w której potrzebujemy zmienić typ danych, możemy wykorzystać kilka funkcji:

```
int() - zamiana na liczby całkowite

# przykładowo
dana_wejsciowa = "12"
print(type(dana_wejsciowa))
<class 'str'>
wyjsciowo = int(dana_wejsciowa)
print(type(wyjsciowo))
<class 'int'>
print(wyjsciowo)
12

float() - zamiana na liczbę zmiennoprzecinkową

# przykładowo
dana_wejsciowa = "12.45"
print(type(dana_wejsciowa))
<class 'str'>
wyjsciowo = float(dana_wejsciowa)
print(type(wyjsciowo))
<class 'float'>
print(wyjsciowo)
12.45

str() - zamiana na ciągi znaków

# przykładowo
dana_wejsciowa = 12.45
print(type(dana_wejsciowa))
<class 'float'>
wyjsciowo = str(dana_wejsciowa)
```

```
print(type(wyjsciowo))
<class 'str'>
print(wyjsciowo)
12.45
```

> **Polecenie 1**
>
> Napisz program obliczający wiek na podstawie danych podanych przez użytkownika. Wiek powinien być wyświetlony na standardowym wyjściu.
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - `rok_urodzenia` – rok urodzenia użytkownika, liczba naturalna dodatnia pobrana od użytkownika
>
> *Wynik:*
>
> Program na standardowym wyjściu wypisuje wiek użytkownika.

> **Ciekawostka**
>
> Polecenie `int('4A',base=16)` spowoduje przeliczenie liczby `4A` z postaci szesnastkowej na dziesiętną.

Sposoby formatowania informacji przekazywanych przez program

Funkcja `print()` służy do wyświetlania informacji na standardowym wyjściu. Gdy chcemy wypisać dużą liczbę różnych informacji w ujednolicony sposób, możemy ułatwić sobie takie zadanie, stosując metodę `.format()`.

> **Przykład 1**
>
> Przeanalizujmy wyświetlenie 5 różnych liczb zmiennoprzecinkowych – zostaną one wypisane na ekranie. Kod będzie wyglądał w następujący sposób:
>
> ```
> ```

```
liczby = [ 2.34, 11.24, 1.864, 33.1, 12.456 ]
for liczba in liczby:
    print(liczba)

2.34
11.24
1.864
33.1
12.456
```

Chcielibyśmy wypisać te liczby w kolejnych wierszach. Tak, aby kropka dziesiętna była zawsze w tym samym miejscu. Formatowanie specjalne to sposób formatowania powodujący np. równe rozmieszczenie znaku separatora tych liczb. Użyjemy opisu formatowania `{:>6.3f}`, który oznacza:

- znak `">"` – równamy do prawej strony,
- `6` – długość napisu ma mieć minimum 6 znaków (wliczając separator), jeśli napis będzie krótszy, zostanie uzupełniony znakami spacji na początku,
- `3` – ilość miejsc dziesiętnych, jeśli liczba będzie miała więcej, wynik będzie zaokrąglony, jeśli mniej, zostaną dodane `0`,
- znak `"f"` – oznacza, że liczba jest typu `float`,
- znaki `":"` oraz „`"."` są wymagane zgodnie z dokumentacją Pythona.

```
for liczba in liczby:
    print("{:>6.3f}".format(liczba))

 2.340
11.240
 1.864
33.100
12.456
```

Ogólna postać zapisu kodów sterujących formatowaniem dla liczb wygląda następująco:

```
[ wyrównanie ][ znak ][ilość cyfr][.precyzja][typ]

wyrównanie: "<" | ">" | "=" | "^"
do lewej, do prawej, wypełnianie znakami '0', centrowanie

znak: "+" | "-" | " "
```

```
znak dla liczb dodatnich i ujemnych, znak tylko dla liczb ujemnyc

liczba cyfr - łączna minimalna długość (liczba znaków) wypisanego
będzie mniejsza, wynikowy ciąg zostanie uzupełniony na początku z

.precyzja - minimalna ilość cyfr części dziesiętnej, która będzie
dziesiętnej będzie więcej


typ: "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" |
określa sposób, w jaki ma być prezentowana liczba, np:
"f" - zmiennoprzecinkowa, domyślnie 6 miejsc po przecinku
"e" - notacja wykładnicza, inżynierska
"%" - procentowa, wyświetla liczbę przeliczoną na procent
```

Przykładowe wykonanie różnych sposobów formatowania:

```
l1 = 2.36
l2 = -3.14
l3 = 4.467
print('Liczba: {}, {}, {}'.format(l1, l2, l3))
Liczba: 2.36, -3.14, 4.467
print('Liczba: {:<+6.3f}, {:>6.4f}, {}'.format(l1, l2, l3))
Liczba: +2.36 ,  -3.14, 4.467
print('Liczba: {:<+6.3f}, {:>6.4f}, {:6.2f}'.format(l1, l2, l3))
Liczba: +2.360, -3.1400,   4.47
print('Liczba: {:<+6.3f}, {:>6.4f}, {:6.3f}'.format(l1, l2, l3))
Liczba: +2.360, -3.1400,  4.467
print("Liczba: {:<10.5f}.".format(123.122))
Liczba: 123.12200 .
print("Liczba: {:<10.5f}.".format(123.122345678))
Liczba: 123.12235 .
print("Liczba: {:<10.5f}.".format(123123.122345678))
Liczba: 123123.12235.
print('Liczba: {:f}'.format(453))
Liczba: 453.000000
print('Liczba: {:e}'.format(453))
Liczba: 4.530000e+02
```

> **Ważne!**

> Sposób formatowania i postać kodów sterujących są opisane w dokumentacji, dostępnej po wydaniu polecenia `help('FORMATTING')`.
> **Polecenie 2**
>
> Napisz program, który pobierze od użytkownika dane 3 osób: imię, liczbę punktów uzyskanych na sprawdzianie oraz obliczy wynik procentowy każdej osoby. Przyjmij, że maksymalnie można uzyskać 120 punktów. Wyniki wyświetl w tabeli zawierającej nagłówek:
> `' Lp. |  Imię  | Punkty | Procenty '.`
>
> **Specyfikacja problemu:**
>
> *Dane:*
>
> - `maks` – liczba naturalna dodatnia; maksymalna liczba punktów możliwych do uzyskania na sprawdzianie
> - `imiona` – pobrane od użytkownika trzy imiona; trzy łańcuchy znaków
> - `punkty` – pobrane od użytkownika liczby zdobytych przez każdego ucznia punktów
>
> *Wynik:*
>
> Na standardowym wyjściu program wypisuje tabele składającą się z nagłówka oraz wierszy reprezentujących wyniki każdego z trzech uczniów.

> **Polecenie 3**
>
> Przygotuj kilka tabel podobnych do tej z poprzedniego polecenia. Porównaj i przedyskutuj z innymi osobami wyniki działania programów.

Interfejs graficzny i wprowadzanie danych

Niektóre programy mają graficzny interfejs użytkownika (GUI). Podczas pisania takich aplikacji wykorzystywane są środowiska RAD. Dla języka Python nie powstało rozbudowane środowisko tego typu.

Do pisania graficznych aplikacji desktopowych możemy wykorzystać środowiska IDE, które pomagają tworzyć kod źródłowy. W przypadku języka Python dostępny jest moduł

PySimpleGUI, pozwala on wyświetlać proste okna dialogowe.

Opisywany moduł trzeba zainstalować za pomocą menedżera `pip`. Należy w tym celu wydać komendę w linii poleceń systemu operacyjnego:

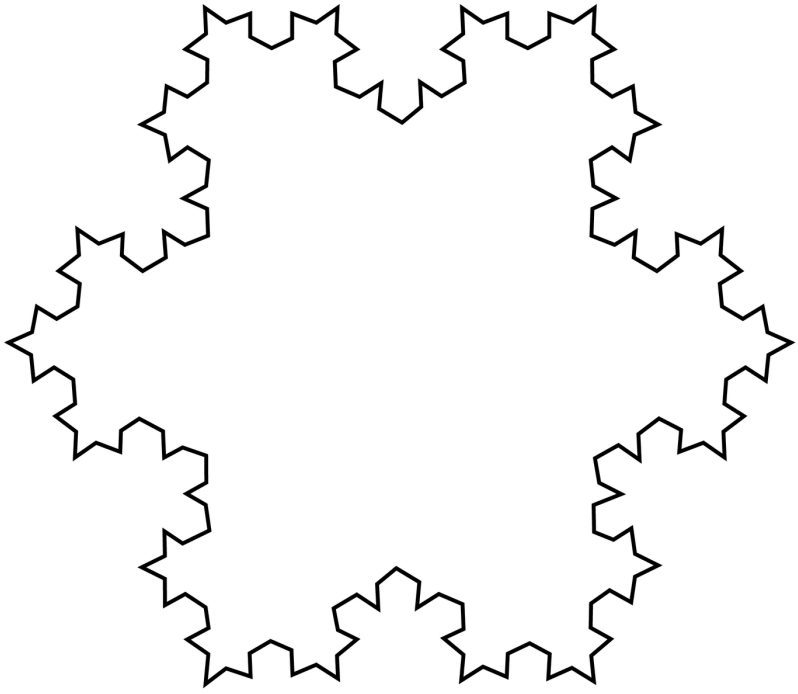
```
pip install PySimpleGUI
```

A oto wywołanie funkcji, która wyświetli okno komunikatu:

```
import PySimpleGUI as sg
sg.Popup("Nasz pierwszy program graficzny",
    "Witam serdecznie - nazywam się Python,",
    "jestem językiem programowania, potrafię działać w wielu środ
    "warto się ze mną zaprzyjaźnić - a zobaczysz, ile jestem w st
    "Zatem - do nauki i pracy  ;-)")
'OK'
>>>
```

> **Ciekawostka**
>
> Składnię `import PySimpleGUI as sg` stosujemy w celu skrócenia zapisu w programie.

Innym przykładem biblioteki realizującej operacje graficzne jest `turtle`. Będziemy korzystać z niej przy tworzeniu rysunków fraktali.

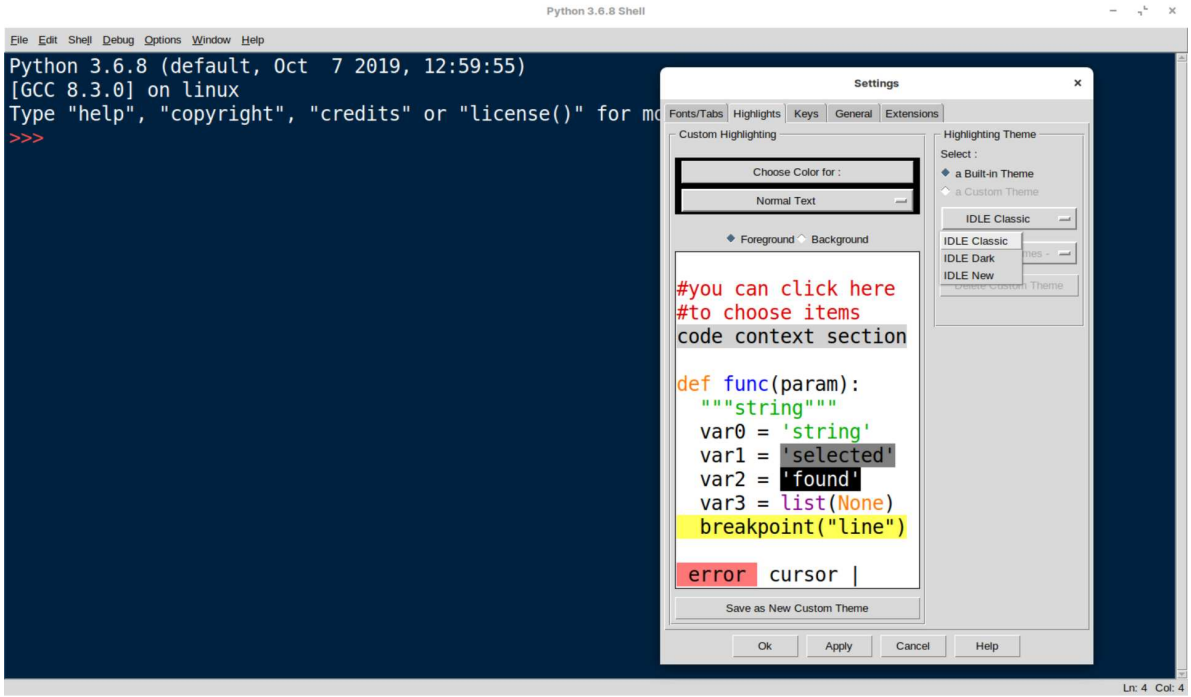


Python Turtle Graphics

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

`Turtle` bazuje na innej bibliotece – `Tkinter`. Jest to standardowa biblioteka języka Python, dostępna bez konieczności doinstalowania.

Środowisko IDLE, które wykorzystujemy do tworzenia programów, zostało opracowane z zastosowaniem właśnie tej biblioteki.



Środowisko programistyczne IDLE bazuje na bibliotece TKinter .

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Oto przykładowy kod, który pozwala wyświetlić minimalne okno programu i zamknąć okno po naciśnięciu przycisku.

```
1  import tkinter
2
3  def ok():
4      window.destroy()
5
6  window = tkinter.Tk()
7  window.title("Nasz pierwszy program graficzny")
8  label = tkinter.Label(window,
9      text = "Witam serdecznie - nazywam się Python,").pack()
10 label = tkinter.Label(window,
11     text = "jestem językiem programowania, potrafię działać w wie
12 label = tkinter.Label(window,
13     text = "warto się ze mną zaprzyjaźnić - a zobaczysz, ile jest
```

```
14 label = tkinter.Label(window,
15     text = "Zatem - do nauki i pracy  ;-)").pack()
16
17 b = tkinter.Button(window, text="OK - koniec", command=ok).pack()
```

Słownik

argument

element składni w określonym języku programowania, który w wyniku wywołania podprogramu zostaje utożsamiony (skojarzony) z określonym parametrem podprogramu

graficzny interfejs użytkownika (GUI)

(ang. *Graphical User Interface*) interfejs pozwalający komunikować się z programem za pomocą widżetów; pozwala również na rysowanie

IDE

(ang. *Integrated Developement Environment*) zintegrowane środowisko programistyczne, które wspomaga pracę z kodem źródłowym; przykładowymi IDE są Atom, PyCharm, Visual Code i Sublime Text

inspektor obiektów

w środowisku PyCharm narzędzie umożliwiające podgląd obiektów w czasie rzeczywistym, sprawdzanie ich wartości oraz typu

parametr

element składni w określonym języku programowania umożliwiający komunikację pomiędzy podprogramem wywołanym a programem wywołującym; parametry określa się wraz z deklaracją określonego podprogramu w jego nagłówku

RAD

(ang. *Rapid Application Developement*) metodyka udostępniająca zbiór narzędzi do szybkiego tworzenia aplikacji; programista rysuje w nim elementy interfejsu, zaś kod dodaje się tylko do poszczególnych obiektów; przykładowymi środowiskami RAD są QT Designer, Visual Studio, Lazarus i Delphi

standardowe wejście

standardowe urządzenie służące do wprowadzania danych; zazwyczaj jest to klawiatura; istnieją również niestandardowe wejścia (np. pliki)

standardowe wyjście

standardowe urządzenie służące do przedstawiania wyników działania programu; zazwyczaj jest to ekran (w przypadku Pythona: aktywne okno terminala tekstowego); istnieją też niestandardowe wyjścia, takie jak plik

Schemat interaktywny

Polecenie 1

Napisz program, który dla danych podanych przez użytkownika obliczy jego BMI. Wynik powinien zostać wypisany na standardowe urządzenie wyjścia.

Specyfikacja problemu:

Wzór:

- $\mathrm{BMI} = \frac{\mathrm{waga}[\mathrm{kg}]}{\mathrm{wzrost}[\mathrm{m}^2]}$

Dane:

- `waga` – liczba naturalna dodatnia; waga wprowadzona przez użytkownika wyrażona w kilogramach

- `wzrost` – liczba zmiennoprzecinkowa dodatnia; wzrost wprowadzony przez użytkownika wyrażony w metrach

Wynik:

- Na podstawie wprowadzonych danych program wypisuje na standardowym wyjściu BMI użytkownika.

```
# Tutaj dodaj własny kod. Użyj funkcji
# print()
```

```
1
```

Jeśli chcesz powtórzyć wiadomości ze szkoły podstawowej, stwórz program, używając schematu blokowego.

Polecenie 2

Porównaj swoje rozwiązanie z filmem.

Film dostępny pod adresem /preview/resource/Rw0ReNLErVoFh

Film nawiązujący do treści materiału: Komunikacja programu z użytkownikiem.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Uzupełnij zdania.

Wszystkie dane przekazywane ze standardowego [ ] Python zwraca jako [ ].

Dane te przekazuje funkcja [ ].

wyjścia | object | list | string | wejścia | słownik | int | print() | input()

Ćwiczenie 2

Wskaż poprawne wywołania funkcji input().

- [ ] input('Podaj treść:')
- [ ] input(print("Podaj treść:'))
- [ ] input("Podaj treść: ")
- [ ] input()

Ćwiczenie 3

Wypełnij puste pola odpowiednimi symbolami.

Formatowanie obiektu string metodą format:

[ ] – dopełnienie spacjami do lewej

[ ] – dopełnienie spacjami do prawej

[ ] – centrowanie

Ćwiczenie 4

Zaznacz poprawnie zapisane polecenie obliczenia wieku, na podstawie danych wprowadzonych z klawiatury.

- ◯ `wiek = 2021 + input("Podaj rok urodzenia")`
- ◯ `wiek = 2021 - int(input("Podaj rok urodzenia"))`
- ◯ `wiek = 2021 - input("Podaj rok urodzenia")`
- ◯ `wiek = str(2021) - input("Podaj rok urodzenia")`

Ćwiczenie 5

Pogrupuj odpowiednio aplikacje.

RAD – Rapid Application Developement

IDE – Integrated Developement Editor

Zwykłe edytory tekstowe

CodeBlocks | QT Creator | Sublime Text Editor | PyCharm | Vi | Eclipse | Atom | FocusWriter | Anjuta Dev | Geany | Notepad | Embracadero Studio | WaveMaker | Libre Writer | Lazarus Studio | MS-Word | Emacs | Visual Studio

Ćwiczenie 6

Zaznacz poprawną odpowiedź. Wskaż jaki wynik da polecenie `'Liczba -> {:>16,.3f}'.format(425935712.29389)`.

- ◯ `'Liczba -> 425,935,712.293'`
- ◯ `'Liczba -> 425,935,712.294'`
- ◯ `ValueError: Invalid format specifier`

Ćwiczenie 7

Przeanalizuj kod i wykonaj ćwiczenie.

```
1  print("Z jakiego filmu to cytaty?")
2  print("Niech Moc będzie z tobą.")
3  film = input( )
4  print("Wstrząśnięte, nie zmieszane.")
5  film = input( )
6  print("Houston, mamy problem!")
7  film = input( )
8  print("Cytat pochodzi z filmu:", film_1)
9  print("Cytat pochodzi z filmu:", film_2)
10 print("Cytat pochodzi z filmu:", film_3)
```

Wskaż błąd w kodzie.

Ćwiczenie 8

Uzupełnij kod, tak aby program wyświetlił pozostałe dwie odpowiedzi według wzoru. Popraw ewentualne błędy.

```
film1 = Star Wars
film2 = "'Goldfinger'
film3 = "'Apollo 13'"
print ("Z jakiego filmu to cytaty?")

print ("'Niech Moc będzie z tobą.'")
print (Cytat pochodzi z filmu, film1)
```

```
1
```

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Operacje wejścia i wyjścia w języku Python

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> Zakres podstawowy. Uczeń:
>
> 1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Scharakteryzujesz operacje wejścia/wyjścia w języku Python.
- Dokonasz zmiany typu danych.
- Sformatujesz wyniki.

- Użyjesz modułów `PySimpleGUI` oraz `Tkinter` do tworzenia prostych okien dialogowych.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Operacje wejścia i wyjścia w języku Python". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla i odczytuje temat lekcji oraz cele zajęć. Prosi uczniów o sformułowanie kryteriów sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z multimedium.** Uczniowie zapoznają się z treścią polecenia 1 z sekcji „Schemat interaktywny". Rozwiązują problem, pisząc program w języku Python lub za pomocą schematu blokowego. W kolejnym kroku porównują swoje rozwiązanie

z zaprezentowanym w filmie. Następnie na forum klasy omówione zostają ewentualne wątpliwości i spostrzeżenia.
2. Uczniowie w parach wykonują ćwiczenia nr 1-8. Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku Python.

Praca domowa:

1. Uczniowie wykorzystując moduł *PySimpleGUI*, piszą program, który będzie umożliwiał wpisanie dwóch liczb oraz poda prawidłowy wynik mnożenia.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.