Temat 5. Wprowadzenie do programowania w języku Python

1. **Usystematyzowanie podstawowych pojęć**
2. **Programowanie grafiki w języku Python – przypomnienie**
   2.1. **Środowisko programowania IDLE**
   2.2. **Wykorzystanie grafiki żółwia do tworzenia rysunków**
3. **Wyprowadzenie napisów na ekran**
4. **Stosowanie zmiennych i wykonywanie obliczeń**
5. **Stosowanie instrukcji warunkowej do realizacji algorytmów z warunkami**
6. **Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych**

Warto powtórzyć

1. Jak tworzymy program w języku Scratch?
2. W jaki sposób w języku Scratch wyprowadzamy napisy na ekran monitora?
3. Jak tworzymy zmienną w języku Scratch?
4. W jaki sposób w języku Scratch przypisujemy zmiennej konkretną wartość, a jak wprowadzamy wartość zmiennej z klawiatury?
5. Za pomocą jakiego polecenia realizujemy sytuację warunkową w języku Scratch?
6. W jaki sposób zapisujemy iterację w języku Scratch?

1. Usystematyzowanie podstawowych pojęć

Na lekcjach informatyki w klasie ósmej omówimy wybrane algorytmy, które będziemy zapisywać w języku programowania Python. Niektóre z nich przedstawimy również (dla porównania) w dydaktycznych środowiskach programowania – Scratch i Baltie.

Pojęcia związane z programowaniem:

- **Algorytm** – uporządkowany i uściślony sposób rozwiązania danego **problemu**, zawierający szczegółowy opis wykonywanych czynności w skończonej liczbie kroków.
- **Język programowania** – specjalny język służący do pisania **programów komputerowych**. Jest on zbiorem określonych instrukcji i zasad składni.
- **Program komputerowy** – ciąg instrukcji języka programowania realizujący **algorytm**.

- **Program (kod) źródłowy** – program napisany w języku programowania (np. języku Python), w postaci ciągu instrukcji tekstowych.
- **Program (kod) wynikowy (maszynowy)** – program w języku wewnętrznym komputera (ciąg instrukcji dla procesora wraz z danymi dla tych instrukcji).
- **Algorytm z warunkami** – algorytm zawierający sytuacje warunkowe. Aby w języku programowania napisać program realizujący algorytm z warunkami, stosujemy **instrukcję warunkową**.
- **Iteracja** – polega na powtarzaniu tej samej operacji (ciągu operacji). Iterację implementujemy (zapisujemy w kodzie źródłowym), stosując tzw. **pętlę**. Z pętlą mamy do czynienia, gdy w pewnym kroku algorytmu wracamy do jednego z wcześniejszych kroków, co powoduje, że kroki te mogą zostać wykonane wiele razy.
- **Algorytm iteracyjny** – algorytm zawierający powtarzanie poleceń. Aby w języku programowania napisać program realizujący algorytm iteracyjny, stosujemy **instrukcje iteracyjne**.
- **Zmienna** – w programie komputerowym to nazwana część pamięci operacyjnej komputera (RAM) o unikatowym adresie. W każdym języku programowania jest określony sposób korzystania ze zmiennych, w tym nadawania im wartości. W niektórych językach programowania musimy określić nie tylko nazwę zmiennej, ale i jej **typ**.
- **Typ zmiennej** – wyznacza zbiór wartości, które może ona przyjmować (np. liczby całkowite, liczby rzeczywiste).

Kompilacja K

Przetłumaczenie programu w całości na język zrozumiały dla procesora, tak by mógł go wykonać komputer. Jeśli kompilacja przebiegnie poprawnie, można uruchomić program. Raz skompilowany program nie wymaga już powtórnej operacji tłumaczenia.

Interpretacja I

Tłumaczenie programu tworzonego w języku programowania instrukcja po instrukcji, tak aby komputer mógł wykonać każdą z nich. Tłumaczenie następuje każdorazowo przy uruchomieniu programu.

Program komputerowy może występować w dwóch postaciach:
- jako program (kod) źródłowy,
- jako program (kod) wynikowy (maszynowy).

Program napisany w języku programowania (**kod źródłowy**) nie jest zrozumiały dla komputera. Komputer wykonuje tylko program napisany w języku wewnętrznym (**kod wynikowy**). Tłumaczeniem programu źródłowego na kod maszynowy zajmuje się specjalny program zwany **translatorem**, a proces tłumaczenia programu napisanego w języku programowania wysokiego poziomu na język wewnętrzny komputera nazywamy **translacją**. Może ona przebiegać w formie **kompilacji** lub **interpretacji**.

W językach kompilowanych (np. C++) musimy napisać cały program, zapisać go w pliku, skompilować i dopiero po poprawnej kompilacji uruchomić. Jeśli **kompilator** znajdzie w programie błędy, wskaże je. Wtedy program należy poprawić, zapisać, ponownie skompilować, a następnie uruchomić. W środowisku programowania Baltie pod elementami graficznymi jest ukryty również język kompilowany – język programowania C.

W językach interpretowanych (np. Scratch, Python) możemy napisać program składający się z wielu poleceń (tworząc tzw. skrypt), zapisać go w pliku, a następnie uruchomić. Mówimy wówczas, że pracujemy w **trybie skryptowym**.

W odróżnieniu od kompilatora, **interpreter** próbuje wykonać program instrukcja po instrukcji. Jeżeli w programie wystąpi błąd, interpreter przerwie wykonanie programu i wyświetli komunikat o błędzie. Należy poprawić wskazany błąd, ponownie zapisać program w pliku i go uruchomić. Jeśli interpreter nie znajdzie więcej błędów, zostanie wykonany cały program.

W dydaktycznych środowiskach programowania (np. Scratch, Baltie) zwykle upraszcza się sposób wprowadzania poleceń (instrukcji języka programowania). Można je zastąpić elementami graficznymi umieszczanymi w obszarze roboczym – z tego powodu takie środowiska nazywa się „wizualnymi".

Niezależnie od zastosowanego środowiska:

- kolejność występowania instrukcji w programie (także przedstawionych jako elementy graficzne) powinna odpowiadać kolejności operacji realizujących dany algorytm,
- postać instrukcji musi być zgodna z zasadami składni danego języka – nie może zabraknąć nawet jednej spacji, jednego dwukropka, przecinka, średnika czy innego znaku w miejscach, w których są wymagane.

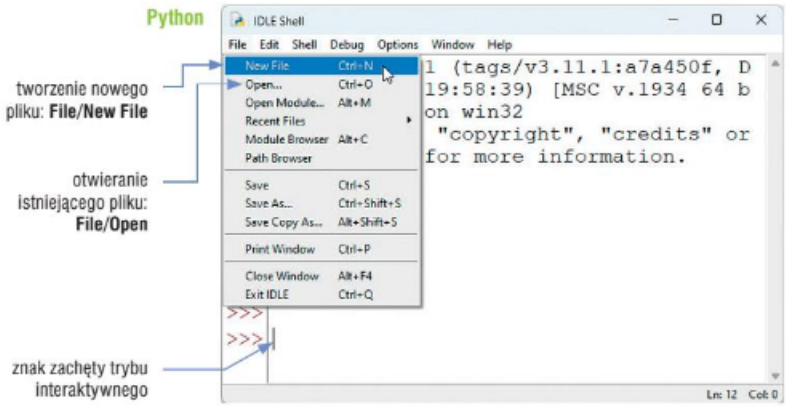
2. Programowanie grafiki w języku Python – przypomnienie

2.1. Środowisko programowania IDLE

Programy w języku Python piszemy w specjalnym edytorze, zawartym w zintegrowanym środowisku programistycznym o nazwie **IDLE**. Pakiet IDLE zawiera także interpreter i inne narzędzia wspomagające programowanie.

Środowisko programistyczne IDLE udostępnia dwa tryby pracy: **interaktywny** i **skryptowy**.
IDLE zawiera narzędzia ułatwiające pisanie programów, m.in.: edytor kodu źródłowego, interpreter i tzw. **powłokę** (narzędzie umożliwiające m.in. pracę w trybie interaktywnym).

Do tworzenia programów w języku Python służy tryb skryptowy, do którego możemy przejść z trybu interaktywnego, z okna powłoki Pythona (Python Shell) – rys. 1. Po wybraniu opcji **File/New File** otworzy się okno edytora kodu źródłowego (rys. 2a), w którym piszemy program. Okno to otwiera się również po wybraniu opcji **File/Open**, gdy otwieramy istniejący plik.



Rys. 1. Powłoka Pythona (Python Shell)

2.2. Wykorzystanie grafiki żółwia do tworzenia rysunków

Pierwsze programy w języku Python pisaliśmy już w klasie 7., tworząc grafikę z wykorzystaniem modułu `turtle`. Zanim zaczniemy poznawać inne możliwości języka Python, przypomnimy sobie, jak pisaliśmy program wykorzystujący grafikę żółwia.

Python

```
import turtle

turtle.color("black","green")
turtle.begin_fill()
turtle.circle(100)
turtle.end_fill()

turtle.color("black","red")
turtle.begin_fill()
turtle.circle(80)
turtle.end_fill()

turtle.color("black","yellow")
turtle.begin_fill()
turtle.circle(60)
turtle.end_fill()

turtle.color("black","blue")
turtle.begin_fill()
turtle.circle(40)
turtle.end_fill()
```

Rys. 2a. Okno edytora kodu źródłowego z napisanym programem

Rys. 2b. Wynik działania programu widoczny w oknie **Python Turtle Graphics**

Program zapisujemy w pliku na dysku, wybierając w oknie edytora opcję **File/Save As**. Następnie uruchamiamy program poprzez wywołanie interpretera Pythona, wybierając opcję **Run/Run Module** (lub naciskając klawisz **F5**).

W oknie edytora kodu źródłowego umieszczono również opcje otwierania plików w menu **File** (**New File** i **Open**).

W przypadku używania modułu `turtle` wynik działania programu pojawi się w oknie **Python Turtle Graphics** (rys. 2b).

Ćwiczenie 1. Programujemy grafikę w języku Python

1. Uruchom środowisko programistyczne IDLE/Python.
2. Napisz program rysujący pawie oczko, podobne do pokazanego na rysunku 2b. Na rysunku 2a. pokazano kod przykładowego programu. Przeanalizuj go, objaśniając poszczególne instrukcje. Użyj kolorów według własnego pomysłu.
3. Zapisz program w pliku pod nazwą *Pawie oczko*. Uruchom program.

Ćwiczenie 2. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 1., aby powstała figura złożona z kół współśrodkowych, podobna do pokazanej na rysunku 3. Na koniec schowaj żółwia.
2. Zapisz program w pliku pod nazwą *Koła*. Uruchom program.

Wskazówka: Poza poleceniami użytymi w programie pokazanym na rysunku 2a, przydatne mogą być polecenia: `left()`, `right()`, `forward()`, `penup()`, `pendown()`, `hideturtle()`.

Rys. 3. Wynik działania programu – ćwiczenie 2.

3. Wyprowadzenie napisów na ekran

Język Python, poza instrukcjami umożliwiającymi tworzenie grafiki, posiada instrukcje realizujące czynności, takie jak: wprowadzanie danych, wyprowadzanie wyników, wykonywanie obliczeń, określanie warunków czy powtarzanie operacji. Język Python, tak jak inne języki programowania, posiada odpowiednie zasady składni oraz właściwe słownictwo – podobnie jak języki naturalne, którymi posługujemy się na co dzień.

Wyprowadzanie na ekran komunikatów i wyników umożliwia **instrukcja wyjścia** – funkcja `print()`. Argumentami funkcji (podanymi w nawiasach) mogą być teksty, wyrażenia arytmetyczne lub zmienne oddzielone przecinkami.

Po wyświetleniu komunikatu lub wyniku następuje przejście do nowego wiersza. Jeśli argumentem jest wyrażenie, najpierw zostanie obliczone, a następnie program wyprowadzi jego wartość.

Na przykład:

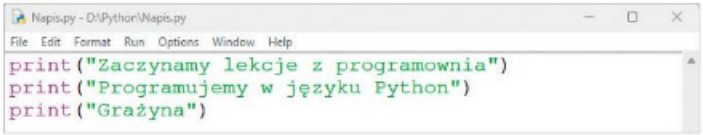
`print("Informatyka")`
– wyprowadza na ekran napis „Informatyka",

`print(23 + 17)`
– wyprowadza na ekran wartość wyrażenia, czyli 40,

`print(wzrost)`
– wyprowadza na ekran wartość zmiennej wzrost (o ile zmiennej tej została wcześniej przypisana wartość),

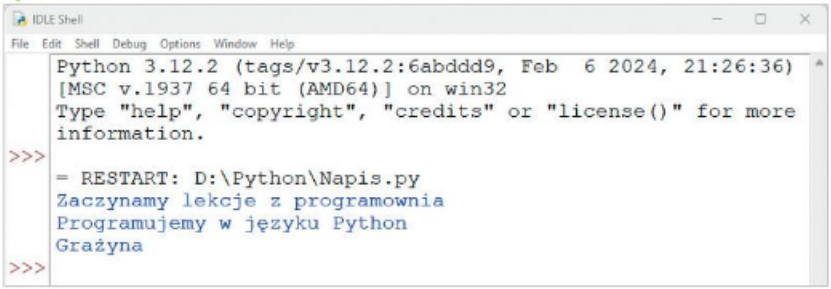
`print("suma liczb 345 i 36 wynosi:", 345 + 36)`
– wyprowadza na ekran napis „suma liczb 345 i 36 wynosi: " i w tym samym wierszu obliczoną wartość wyrażenia (czyli 381).

Python



Rys. 4a. Okno edytora kodu źródłowego (Python) – ćwiczenie 4.

Python



Rys. 4b. Okno powłoki Pythona z wynikiem działania programu (wyświetlonymi napisami) – ćwiczenie 4.

Ćwiczenie 3. Wyświetlamy napisy na ekranie

1. Napisz program wyświetlający na ekranie napis: „Jestem na lekcji informatyki.".
2. Zapisz program w pliku pod nazwą *Napisy*. Uruchom program.

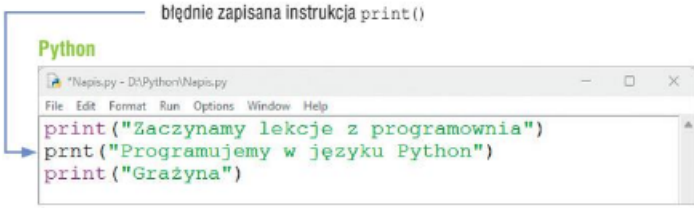
Aby uniknąć popełniania błędów, możesz stosować w edytorze kodu źródłowego operacje kopiowania, wycinania i wklejania fragmentu tekstu z wykorzystaniem **Schowka**, tak jak w każdym innym edytorze tekstu.

Poza poprawnym zapisaniem instrukcji i słów kluczowych musimy również pamiętać o odpowiednim wstawieniu m.in. nawiasów, przecinków, spacji. Jeśli niepoprawnie zapiszemy instrukcję, komputer jej nie wykona, a interpreter wyświetli komunikat o błędzie.

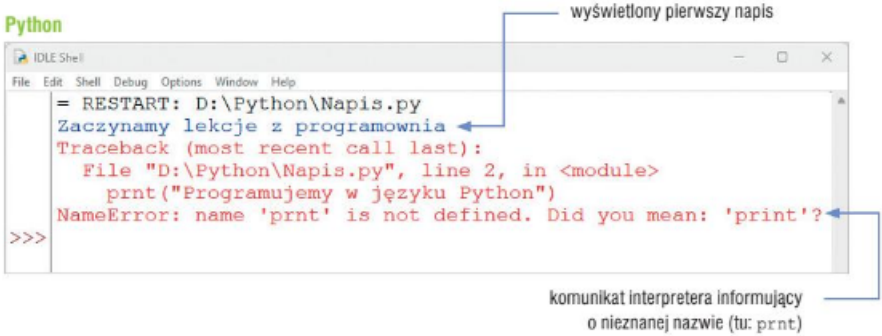
Interpreter tłumaczy program instrukcja po instrukcji. Co to oznacza?

Na rysunku 5a pierwsza instrukcja jest poprawna, ale druga zawiera błąd. W takiej sytuacji (po zapisaniu i uruchomieniu programu) interpreter wyświetli odpowiedni komunikat (rys. 5b). Zostanie wykonana tylko pierwsza instrukcja programu, a druga i trzecia – już nie. Komunikaty interpretera mogą pojawiać się w oddzielnym oknie.

Jeśli podczas pisania programu zwrócimy uwagę na kolory czcionki, możemy uniknąć niepotrzebnych błędów. Warto zauważyć, że czcionka ma różne kolory, np. w nazwie `print()` jest fioletowa, ale w błędnym zapisie tej instrukcji – czarna. Napis umieszczony wewnątrz nawiasów w instrukcji `print()` jest zielony, ale po wyświetleniu na ekranie tekst jest niebieski.



Rys. 5a. Okno edytora kodu źródłowego z błędnie zapisaną instrukcją programu



Rys. 5b. Okno powłoki Pythona z komunikatem interpretera

Ćwiczenie 4. Wyświetlamy napisy na ekranie

1. Zmodyfikuj program zapisany w ćwiczeniu 3., tak aby w drugim wierszu wyświetlało się zdanie „Programujemy w języku Python.", a w trzecim – twoje imię. Zapisz i uruchom program. Jeśli interpreter wykrył błędy, popraw je. Zapisz program i ponownie go uruchom.
2. Dodaj jeszcze dwa inne napisy, w dwóch kolejnych wierszach. Zapisz program i ponownie go uruchom.
3. Zwróć uwagę na różne kolory czcionki kodu pisanego w edytorze kodu źródłowego i napisów wyświetlanych po uruchomieniu programu. Czy jest w tej różnej kolorystyce jakaś prawidłowość? Wyjaśnij na przykładzie wykonywanego ćwiczenia.

4. Stosowanie zmiennych i wykonywanie obliczeń

Aby utworzyć zmienną w środowiskach programowania Scratch i Baltie, określaliśmy jej nazwę (w Baltiem również typ). W języku C++ każdą zmienną trzeba zadeklarować, czyli określić jej nazwę i typ.

W języku Python nie trzeba deklarować zmiennych. Nie musimy wcześniej określać typu zmiennej. Interpreter określa typ zmiennej na podstawie rodzaju danych, jaki jej przypiszemy. Nie ustanawia jednak typu zmiennej na zawsze, ponieważ przy kolejnym przypisaniu może się on zmienić. Na przykład jeśli na początku programu przypiszemy zmiennej *liczba* wartość 34 (która jest liczbą całkowitą), to jej typ jest całkowity. Jeśli jednak w dalszej części programu przypiszemy jej wartość 45,8 (która jest liczbą rzeczywistą), to zmienna *liczba* będzie już typu rzeczywistego.

Zmiennej stosowanej w programie możemy nadać konkretną wartość za pomocą **instrukcji przypisania**.

Znak (operator) = oznacza przypisanie zmiennej zapisanej po lewej stronie znaku wartości umieszczonej po jego prawej stronie.

Na przykład:

```
rok = 2018
```

– zmiennej *rok* przypisujemy wartość 2018, która jest liczbą całkowitą, czyli zmienna *rok* będzie typu całkowitego (`int`),

```
wzrost = 159.5
```

– zmiennej *wzrost* przypisujemy wartość 159,5, która jest liczbą rzeczywistą, czyli zmienna *wzrost* będzie typu rzeczywistego (`float`),

```
nazwisko = "Kowalski"
```

nazwisko będzie typu tekstowego (`str`).

Uwagi:

- W języku Python, tak jak w większości języków programowania, część dziesiętną liczby zapisujemy po kropce, a nie po przecinku jak w matematyce.
- W instrukcji przypisania po prawej stronie znaku równości możemy umieścić wyrażenie, np.: `rok = rok + 1` (co oznacza: zmiennej *rok* przypisz wartość zmiennej *rok* zwiększoną o 1).

W języku Python małe i wielkie litery w nazwach zmiennych (także instrukcji) mają różne znaczenie, np. *Suma* i *suma* to dwie różne zmienne. W nazwach zmiennych powinno się używać liter, znaku podkreślenia i cyfr. Nazwa nie może zaczynać się od cyfry. Przyjęte jest stosowanie małych liter i niestosowanie polskich liter. W przypadku nazw kilkuczłonowych zamiast spacji stosujemy znak podkreślenia. Należy nadawać nazwy, które określają znaczenie danej zmiennej, np. *suma*, *liczba_elementow*.

Ćwiczenie 5. Nadajemy wartości zmiennym

1. W trybie interaktywnym sprawdź, w jaki sposób można przypisać wartość zmiennej. Umieść kolejno następujące polecenia (każde po znaku zachęty):

```
liczba = 23
liczba = liczba + 1
print(liczba)
reszta = liczba % 3
print(reszta)
```

2. Napisz jeszcze pięć instrukcji przypisania wartości (liczb lub wyrażeń) zmiennym według własnego pomysłu. Wykorzystaj operatory arytmetyczne z tabeli 1.

Operator	Działanie	Przykład	Wynik
+	dodawanie	23 + 56	79
-	odejmowanie	987 - 233	754
*	mnożenie	432 * 6	2592
//	dzielenie całkowite (z obcięciem części ułamkowej)	55 // 3	18
/	dzielenie zmiennoprzecinkowe (z zachowaniem części ułamkowej)	55 / 3	18.333333333333332
%	obliczenie reszty z dzielenia dwóch liczb całkowitych	37 % 4	1

Tabela 1. *Podstawowe operatory arytmetyczne w języku Python*

Zmiennej stosowanej w programie możemy nadać również wartość za pomocą instrukcji przypisania, wprowadzając wartość z klawiatury po uruchomieniu programu. W tym celu stosujemy **instrukcję wejścia** – funkcję `input()`.

Argumentem funkcji (podanym w nawiasach) jest tekst (ciąg znaków, łańcuch). Tekst pojawi się na ekranie jako swego rodzaju „zaproszenie" do wpisania ciągu znaków z klawiatury. Jeśli użyjemy funkcji `input()` w instrukcji przypisania, wpisany przez użytkownika ciąg znaków zostanie zapamiętany w zmiennej podanej po lewej stronie:

`nazwa_zmiennej = input("zaproszenie do wprowadzenia danej z klawiatury")`.

Na przykład:

`nazwisko = input("Wprowadź nazwisko: ")`

– jeśli wpiszemy z klawiatury nazwisko „Kowalski", w zmiennej *nazwisko* zostanie zapamiętany ciąg znaków „Kowalski",

`a = input("Wprowadź liczbę: ")`

– jeśli wpiszemy z klawiatury liczbę 346, w zmiennej *a* zostanie zapamiętany ciąg znaków „346", a nie liczba 346.

Aby wprowadzona za pomocą instrukcji `input()` dana została zapamiętana jako liczba, musimy dodać instrukcję, która zamieni ciąg znaków na liczbę:
`int()` – w przypadku liczb całkowitych, `float()` – w przypadku liczb rzeczywistych.

Na przykład:
`a = int(input("Wprowadź liczbę: "))`
– jeśli wpiszemy z klawiatury liczbę 346, w zmiennej a zostanie zapamiętana liczba całkowita 346,

`srednia = float(input("Podaj średnią ocen"))`
– jeśli wpiszemy z klawiatury liczbę 4.3, w zmiennej *srednia* zostanie zapamiętana liczba rzeczywista 4,3.

Ćwiczenie 6. Piszemy program z użyciem zmiennych wprowadzanych z klawiatury

1. Program pokazany na rysunku 6. pozwala na wprowadzenie z klawiatury dwóch liczb całkowitych *a* i *b*, obliczenie ich sumy i wyprowadzenie wyniku (wartości zmiennej *suma*) na ekran. Przepisz kod tego programu.
2. Zapisz program w pliku pod nazwą *Suma*. Uruchom program.

Wskazówka: Jeśli zamkniemy okno programu i chcemy ponownie uruchomić program, możemy wybrać w oknie powłoki opcję **File/Open**. Program otworzy się w oknie edytora kodu źródłowego (rys. 6.).

Python

Suma.py - C:\Python\Suma.py

File Edit Format Run Options Window Help

```
a = int(input("Wprowadź liczbę: "))
b = int(input("Wprowadź liczbę: "))
suma = a + b
print("Suma wynosi:", suma)
```

Rys. 6. Program obliczający sumę dwóch liczb całkowitych wprowadzanych z klawiatury i wyprowadzający wynik na ekran – ćwiczenie 6.

Po uruchomieniu programu (opcja **Run/Run Module** lub klawisz **F5** w oknie edytora kodu źródłowego) wynik działania programu pokaże się w oknie powłoki Pythona (rys. 7a).

Python

"Python Shell"

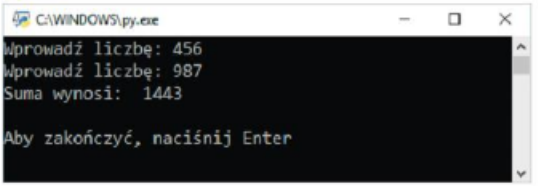
File Edit Shell Debug Options Window Help

```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Python\Suma.py ==========
Wprowadź liczbę: 456
Wprowadź liczbę: 987
Suma wynosi: 1443
```

Rys. 7a. Wynik działania programu w oknie powłoki Pythona (Python Shell) – ćwiczenie 6.

Program możemy również uruchomić, klikając dwukrotnie w Eksploratorze plików nazwę pliku z zapisanym programem – otworzy się okno, w którym zobaczymy wynik działania programu (rys. 7b). Aby program został wykonany, niezbędny jest interpreter Pythona.

Python



Rys. 7b. Wynik działania programu – ćwiczenie 6.

Aby okno pokazane na rysunku 7b nie zamknęło się automatycznie po wyświetleniu napisów, możemy umieścić na końcu programu instrukcję oczekiwania na naciśnięcie klawisza **Enter**: `input("\n\nAby zakończyć, naciśnij Enter")`.

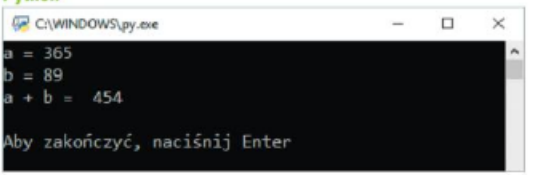
W tym przypadku wartość zwracana przez funkcję `input()` jest pomijana (nie jest przypisywana żadnej zmiennej). Zapis `\n` odpowiada kodowi znaku przejścia do nowego wiersza. Umieszczenie zapisu `\n\n` oznacza wstawienie dwóch nowych wierszy.

Ćwiczenie 7. Modyfikujemy program

1. Zmodyfikuj program *Suma* zapisany w ćwiczeniu 6., aby wynik działania programu wyglądał jak na rysunku 8.
2. Zapisz program w pliku pod tą samą nazwą. Uruchom program tak.

Wskazówka: Jeśli przy uruchomieniu programu wprowadzisz liczbę rzeczywistą, to interpreter wyświetli błąd.

Python



Rys. 8. Wynik działania programu – ćwiczenie 7.

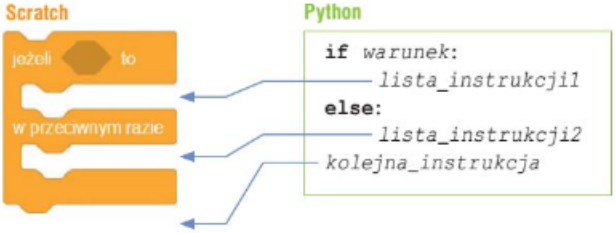
Ćwiczenie 8. Uzupełniamy program

1. Do programu zapisanego w ćwiczeniu 7. dodaj obliczanie iloczynu liczb *a* i *b* oraz wyprowadzanie wyniku mnożenia na ekran (podobnie jak wyniku sumowania w ćwiczeniu 7.).
2. Zapisz program w pliku pod nazwą *Działania*. Uruchom program.

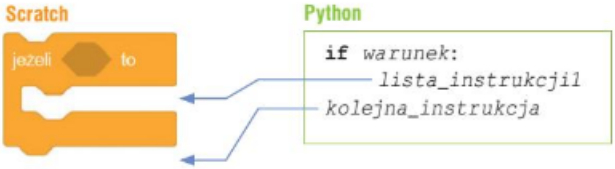
5. Stosowanie instrukcji warunkowej do realizacji algorytmów z warunkami

Zamierzamy napisać w języku Python program obliczający iloraz dwóch liczb pod warunkiem, że druga liczba jest różna od zera. W jaki sposób w języku Python zrealizować sytuację warunkową?

Aby w języku programowania napisać program realizujący algorytm z warunkami, stosujemy **instrukcję warunkową**. Język Python zawiera taką instrukcję – działa ona podobnie jak w środowiskach programowania Baltie i Scratch oraz w języku C++. Instrukcja ta występuje w wersjach pełnej i uproszczonej (rys. 9a, 9b). Jako *lista_instrukcji1* i *lista_instrukcji2* może wystąpić jedna instrukcja (w tym instrukcja warunkowa) lub ciąg instrukcji. Po *warunku* i słowie kluczowym **`else`** umieszczamy dwukropek.



Rys. 9a. Instrukcja warunkowa w języku Python – po lewej odpowiadający jej element w języku Scratch



Rys. 9b. Uproszczona postać instrukcji warunkowej w języku Python – po lewej odpowiadający jej element w języku Scratch

Sprawdzany jest warunek logiczny – jeśli warunek jest spełniony (jest prawdziwy), wykonywana jest *lista_instrukcji1*. Jeśli warunek nie jest spełniony (jest fałszywy) – wykonywana jest *lista_instrukcji2* po słowie **`else`** (rys. 9a), a jeśli brak tej części instrukcji – od razu wykonywana jest *kolejna_instrukcja* po instrukcji warunkowej (rys. 9b).

W języku Python ważne są wcięcia w programie – *listę_instrukcji1* i *listę_instrukcji2* należy przesunąć w prawo przynajmniej o jedną spację (przyjęte jest wcięcie składające się z czterech spacji). Jeśli *lista_instrukcji1* i *lista_instrukcji2* obejmują więcej instrukcji, wszystkie należy tak przesunąć. Jest to sposób wyróżnienia **bloku kodu**.

Jeśli nie zastosujemy wcięcia, komputer potraktuje *listę_instrukcji1* i *listę_instrukcji2* jako *kolejne_instrukcje* i wykona niezależnie od spełnienia warunku.

Operatory		Określenie	Przykład wyrażenia	Interpretacja wyrażenia
Porównania	==	równy	`a == b`	*a* równe *b*
	!=	różny	`a != b`	*a* różne od *b*
	<	mniejszy	`a < b`	*a* mniejsze od *b*
	>	większy	`a > b`	*a* większe od *b*
	>=	większy lub równy	`a >= b`	*a* większe lub równe *b*
	<=	mniejszy lub równy	`a <= b`	*a* mniejsze lub równe *b*
Logiczne	or	alternatywa logiczna (lub)	`a < -5 or a > 5`	*a* mniejsze od –5 lub *a* większe od 5
	and	koniunkcja logiczna (i)	`a > 0 and a < 10`	*a* większe od 0 i *a* mniejsze od 10
	not	negacja logiczna	`not a == 5`	nieprawda, że *a* jest równe 5

Tabela 2. *Podstawowe operatory porównania i logiczne w języku Python*

Python

```
if b != 0:
    iloraz = a / b
    print("a / b = ", iloraz)
else:
    print("Nie dzielimy przez zero")
```

Rys. 10. Przykład stosowania instrukcji warunkowej w języku Python – ćwiczenie 9.

Ćwiczenie 9. Stosujemy instrukcję warunkową

1. Otwórz program *Działania* zapisany w ćwiczeniu 8.
2. Uzupełnij program o obliczenie ilorazu liczb *a* i *b*. Iloraz możemy obliczyć tylko dla *b* różnego od zera – jeśli ten warunek nie będzie spełniony, program powinien wyświetlić odpowiedni napis (rys. 10.).
3. Zapisz program w pliku pod tą samą nazwą. Uruchom program i sprawdź jego działanie dla różnych danych.

Uwaga: Wynik dzielenia liczb całkowitych przypisany zmiennej *iloraz* może nie być liczbą całkowitą. W takiej sytuacji zmienna *iloraz* będzie miała typ rzeczywisty i również otrzymamy poprawny wynik dzielenia.

Python

```
punkty = 100
if sr_ocen > 4 and sr_ocen <= 6:
    punkty_o = 20
print("Liczba punktów:", punkty + punkty_o)
```

Rys. 11. Przykład stosowania instrukcji warunkowej w wersji uproszczonej (Python) – ćwiczenie 10.

Ćwiczenie 10. Stosujemy instrukcję warunkową w wersji uproszczonej

1. Pewien uczeń zdobył 100 punktów w konkursie szkolnym. Jeśli jego średnia ocen będzie wyższa niż 4, uczeń otrzyma dodatkowe 20 punktów. Napisz program obliczający i wyświetlający na ekranie liczbę punktów ucznia. Na rysunku 11. pokazano fragment programu.
2. Dodaj do programu wprowadzanie wartości zmiennej *sr_ocen* typu rzeczywistego (`float`).
3. Zapisz program w pliku pod nazwą *Punkty*. Uruchom program. Sprawdź działanie programu dla kilku różnych danych.

6. Stosowanie instrukcji iteracyjnej do realizacji algorytmów iteracyjnych

Chcemy obliczać, ile metrów bieżących siatki potrzeba na ogrodzenie każdego z pięciu placów zabaw znajdujących się w różnych miejscach. W algorytmie mamy do czynienia z powtarzaniem poleceń obliczania obwodu placu zabaw, czyli z iteracją. W jaki sposób napisać program realizujący algorytm iteracyjny w wybranym języku programowania?

Scratch

Rys. 12a. Element reprezentujący polecenie **powtórz** w języku Scratch

Python

```
import turtle

for i in range(4):
    turtle.left(90)
    turtle.forward(100)
```

Rys. 12b. Rysowanie kwadratu w języku Python

Python

```
for zmienna in lista_wartości:
    lista_instrukcji
```

Rys. 12c. Ogólna postać instrukcji iteracyjnej `for` w języku Python

Aby w języku programowania napisać program realizujący algorytm iteracyjny, korzystamy z **instrukcji iteracyjnych** (instrukcji pętli). W języku Python jedną z takich instrukcji jest instrukcja **`for`**, którą poznaliśmy podczas rysowania kompozycji graficznych z wykorzystaniem modułu `turtle`.

Jako *lista_instrukcji* może wystąpić pojedyncza instrukcja (w tym instrukcja pętli) lub więcej instrukcji (blok instrukcji). *Lista_instrukcji* musi być przesunięta w prawo przynajmniej o jedną spację (przyjęte jest wcięcie składające się z czterech spacji).

Liczbę iteracji określa długość *listy_wartości* po słowie **`in`** (rys. 12b). *Lista_instrukcji* zostanie wykonana dla wszystkich wartości z *listy_wartości*. *Listę_wartości* zapisujemy w różny sposób. Możemy użyć funkcji `range()`, która tworzy sekwencję wartości całkowitych.

Na przykład:

```
for i in range(5):
    print(i)
```

– instrukcja `print(i)` zostanie wykonana pięć razy, a funkcja `range()` wygeneruje kolejne liczby całkowite z przedziału ⟨0, 5), czyli zmienna *i* będzie przyjmować kolejno wartości: 0, 1, 2, 3, 4.

Argumentem funkcji `range()` może być konkretna wartość lub zmienna, np. `range(n)`. Wartość zmiennej *n* możemy wprowadzać z klawiatury. Funkcja `range()` może mieć też dwa lub trzy argumenty.

Ćwiczenie 11. Stosujemy instrukcję **`for`** do rysowania kompozycji graficznych w języku Python z wykorzystaniem modułu `turtle`

1. Przepisz program pokazany na rysunku 13a. Zapisz program w pliku pod nazwą *Kwiat1*. Uruchom program. Ile razy będzie wykonana instrukcja `forward(50)`?
2. Zmodyfikuj program tak, aby najpierw żółw narysował wszystkie okręgi zielone, potem żółte, a na końcu czerwone. Dodatkowo dystans (liczbę) w poleceniu `forward()` wprowadzaj z klawiatury po uruchomieniu programu.
3. Zapisz program w pliku pod nazwą *Kwiat2*. Uruchom program.

Wskazówka: Aby wyświetlić numery wierszy, wybierz z menu **Options** opcję **Show Line Numbers**.

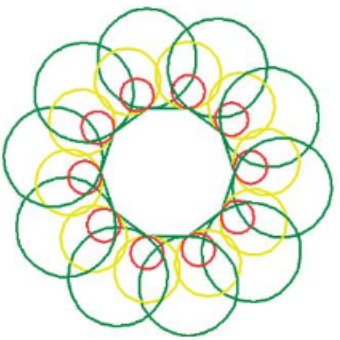
Python

kwiat.py - D:/Python/kwiat.py

File Edit Format Run Options Window Help

```
1  from turtle import *
2
3  pensize(3)
4  for i in range(10):
5      color("green")
6      circle(60)
7      forward(50)
8      color("yellow")
9      circle(40)
10     color("red")
11     circle(20)
12     right(36)
13 hideturtle()
14
15 mainloop()
```

Rys. 13a. Program w języku Python rysujący kompozycję z okręgów – ćwiczenie 11.



Rys. 13b. Wynik działania programu – ćwiczenie 11.

Ćwiczenie 12. Stosujemy instrukcję `for` w języku Python

1. Program pokazany na rysunku 14. realizuje algorytm obliczania obwodu pięciu placów zabaw, których wymiary mają być wprowadzane z klawiatury. Przepisz kod programu pokazany na rysunku 14. Wyjaśnij znaczenie poszczególnych wierszy programu.
2. Zapisz program w pliku pod nazwą *Ogrodzenia_5*. Uruchom program.
3. Odpowiedz na pytania:
   a. Ile razy komputer wykona instrukcję
      `obwod = 2 * szerokosc + 2 * dlugosc`?
   b. Jaka jest wartość zmiennej *i* na koniec programu?

Wskazówka: Aby sprawdzić, jaka jest wartość zmiennej *i* na koniec programu, wyświetl jej wartość na ekranie, umieszczając polecenie: `print(i)` na końcu programu.

Python

*Ogrodzenia_5.py - C:\Python\Ogrodzenia_5.py

File Edit Format Run Options Window Help

```
for i in range(5):
    szerokosc = int(input("Podaj szerokość placu zabaw: "))
    dlugosc = int(input("Podaj długość ogrodu: "))
    obwod = 2 * szerokosc + 2 * dlugosc
    print("Obwód placu zabaw o podanych wymiarach wynosi:", obwod)

input("\n\nNaciśnij Enter, aby zakończyć")
```

Rys. 14. Program realizujący algorytm iteracyjny w języku Python – ćwiczenie 12.

Ćwiczenie 13. Obliczamy obwody *n* placów zabaw

1. Zmodyfikuj program *Ogrodzenia_5* zapisany w ćwiczeniu 12., aby obliczał obwody *n* placów zabaw (wartość *n* wprowadź z klawiatury).
2. Zapisz program w pliku pod nazwą *Ogrodzenia_n*. Uruchom program.
3. Sprawdź działanie programu dla różnych wartości danych *n*, *dlugosc* i *szerokosc*.

Algorytm obliczania obwodów *n* placów zabaw możemy zrealizować w języku Scratch w podobny sposób jak w języku Python. Stosujemy w tym celu polecenie **powtórz**. Musimy również utworzyć zmienne (polecenia dotyczące stosowania zmiennych w języku Scratch umieszczono w grupie **Zmienne**).

Ćwiczenie 14. Stosujemy iterację w języku Scratch

1. Utwórz w programie Scratch program realizujący algorytm obliczania obwodu *n* placów zabaw. Wartość zmiennej *n* i wymiary placów wprowadzaj z klawiatury. Porównaj program z realizacją tego samego algorytmu w języku Python. Jakie widzisz podobieństwa, a jakie różnice?
2. Zapisz program w pliku pod nazwą *Ogrodzenia_n*. Uruchom program i sprawdź jego działanie dla różnych wartości danych *n*, *dlugosc* i *szerokosc*.

Warto zapamiętać

- Kompilator tłumaczy program w całości na język zrozumiały dla procesora, tak aby mógł go wykonać komputer. Natomiast interpreter tłumaczy program instrukcja po instrukcji, tak aby komputer mógł wykonać każdą z nich.
- Środowisko programistyczne Pythona (IDLE) udostępnia dwa tryby pracy: interaktywny i skryptowy. W trybie skryptowym możemy napisać program w edytorze kodu źródłowego, zapisać go w pliku, a następnie uruchomić; jeśli program będzie poprawny, zostanie wykonany.
- Każdy język programowania ma swój zbiór instrukcji, zasady składni i właściwe słownictwo.
- Aby wyświetlić komunikaty i wyniki na ekranie, w języku Python stosujemy instrukcję wyjścia – funkcję `print()`.
- W języku Python wartość przypisana zmiennej określa typ tej zmiennej.
- Zmiennej możemy nadać wartość za pomocą instrukcji przypisania:
  - podając w programie konkretną wartość

  lub
  - wprowadzając tę wartość z klawiatury po uruchomieniu programu (używając funkcji `input()`).
- W języku Python do zapisywania algorytmów, w których występują sytuacje warunkowe, stosujemy instrukcję warunkową **`if`**. Działanie tej instrukcji jest podobne jak w środowiskach programowania Baltie i Scratch oraz w języku C++.
- Jeśli w algorytmie występują powtórzenia, możemy zastosować instrukcję iteracyjną **`for`**. Liczba powtórzeń (iteracji) jest w niej zwykle z góry określona.

Pytania i polecenia

1. Czym różni się kompilacja od interpretacji?
2. W jaki sposób wyświetlamy na ekranie komunikaty i wyniki w języku Python?
3. Jak korzystamy ze zmiennych w języku Python?
4. Jaka będzie wartość zmiennej *b* po wykonaniu programu w języku Python:

```
a = 5
b = 30
a = a + 10
b = a
```

5. Czy po wykonaniu instrukcji przypisania `waga = input("Wprowadź wagę")` i wprowadzeniu z klawiatury liczby 60 komputer zapamięta w zmiennej *waga* liczbę 60? Uzasadnij odpowiedź.
6. W jaki sposób w języku Scratch wprowadzaliśmy dane z klawiatury po uruchomieniu programu (jakich używaliśmy poleceń)? Wskaż na konkretnym przykładzie, na czym polega podobieństwo do wprowadzania danych z klawiatury w języku Python.
7. Czym się różni operator == od = ? Podaj przykłady użycia każdego z tych operatorów.
8. Jak działa instrukcja warunkowa w języku Python? Porównaj jej działanie do działania instrukcji warunkowej w języku Scratch.
9. Kiedy zostaną wykonane instrukcje umieszczone po słowie **`else`** w instrukcji warunkowej?
10. Na czym polega działanie instrukcji **`for`** w języku Python? Podaj przykłady.

Zadania

1. Napisz program obliczający sumę i średnią arytmetyczną trzech liczb rzeczywistych wprowadzanych z klawiatury. Wyniki obliczeń i odpowiednie komunikaty wyświetl na ekranie. Zapisz program w pliku pod nazwą *Średnia*.
2. Napisz program obliczający pole trapezu (*pole*) o podstawach *a* i *b* oraz wysokości *h*. Wartości zmiennych *a*, *b* i *h* wprowadzaj z klawiatury. Wyniki obliczeń i odpowiednie komunikaty wyświetl na ekranie. Zapisz program w pliku pod nazwą *Trapez*.
3. Otwórz program *Trapez* zapisany w zadaniu 2. i dodaj sprawdzanie, czy wysokość *h* jest większa od zera. Dla *h* większego od zera należy wykonać obliczenia pola i wyświetlić wynik, a w przeciwnym wypadku wyświetlić komunikat „błędna dana". Zapisz program w pliku pod tą samą nazwą.
4. Korzystając z programu *Działania* zapisanego w ćwiczeniu 9., sprawdź, czym się różni wynik dzielenia w zależności od typu danych. Nie zapisuj zmian.
5. Otwórz program *Działania* zapisany w ćwiczeniu 9. Dodaj do programu możliwość obliczenia różnicy liczb *a* i *b*. Zapisz program w pliku pod tą samą nazwą.
6. Zmodyfikuj program z zadania 5., aby wykonał obliczenia dla dziesięciu wartości zmiennych *a* i *b* wprowadzonych z klawiatury. Zapisz program w pliku pod nazwą *Działania_10*.
7. Napisz program obliczający objętość (*v*) i pole powierzchni (*p*) sześcianu o boku *a*. Wartość zmiennej *a* wprowadzaj z klawiatury, a wyniki *v* i *p* wyprowadzaj na ekran. Zapisz program w pliku pod nazwą *Sześcian*.
8. Utwórz własną kompozycję graficzną w języku Python wykorzystującą grafikę żółwia. Zastosuj instrukcję `for`.

Dla zainteresowanych

9. Otwórz program zapisany w zadaniu 7. Zmodyfikuj program, aby dla wartości zmiennej *wybor* równej 1 obliczał objętość sześcianu, a dla wartości zmiennej *wybor* równej 0 – pole powierzchni sześcianu (wartość zmiennej *wybor* wprowadzaj z klawiatury). Zapisz program w pliku pod nazwą *Szescian_w*.
10. Zmodyfikuj program z zadania 9., aby wykonał obliczenia dla dziesięciu wartości zmiennej *a* wprowadzonych z klawiatury. Zapisz program w pliku pod nazwą *Sześcian_w_10*.

Przeczytaj, jeśli chcesz wiedzieć więcej...

Język Python zaczął powstawać w roku 1989 jako następca mniej znanego języka ABC. Python odziedziczył po ABC cechę wyróżniającą go wśród innych języków programowania – stosowanie wcięć do wydzielania bloków kodu. W innych językach używa się w tym celu na przykład nawiasów klamrowych `{}` (w C++) lub słów kluczowych `begin ... end` (w Pascalu). Wcięcia stosuje się, aby zachować przejrzystość programu.