STEP IT ACADEMY

PROGRAMMING WITH python

Lekcja # 1

Instalacja Atom
Dane wejściowe
i wyjściowe

Spis treści

Wstęp

Programowanie jest to proces tworzenia programów, które zawierają specjalnie przygotowany kod, zrozumiały dla komputerów. **Kod komputerowy** jest rodzajem instrukcji podążających według konkretnego algorytmu.

Algorytm jest zbiorem zasad/instrukcji, których wykonanie rozwiązuje dany problem. Algorytmów używamy nie tylko w programowaniu ale również życiu codziennym. Na przykład rozwiązując zadanie: przeczytaj treść, zapisz wzory, wykonaj obliczenia, zapisz odpowiedź.

Python – język programowania

Python jest stosunkowo młodym i obiecującym językiem programowania. Celem tego kursu jest nauka tworzenia interesujących gier.

Dlaczego Python? Można by długo opowiadać o zaletach tego języka. Poniżej zebraliśmy najważniejsze cechy przekonujące nas do tego wyboru:

- **Prostota**. Python jest łatwy do nauki i zrozumienia.
- **Zwięzłość**. Przejrzystość kodu i proste konstrukcje. Napisany kod jest intuicyjny do odczytania przez inne osoby.
- **Szybki start**. Początki programowania w Python Będą szybkie i produktywne. Najprostsze programy mogą się składać dosłownie z kilku linijek kodu.

- **Różnorodność**. Możesz wykorzystać Python do różnych platform (pulpitowe, sieciowe, mobilne, i inne) (Rysunek 1).

Rysunek 1

Przygotowanie PC i instalowanie głównych komponentów

Zaczniemy od instalacji wszystkich niezbędnych komponentów do dalszego programowania (Rysunek 2).

Rysunek 2

Python 3.7.

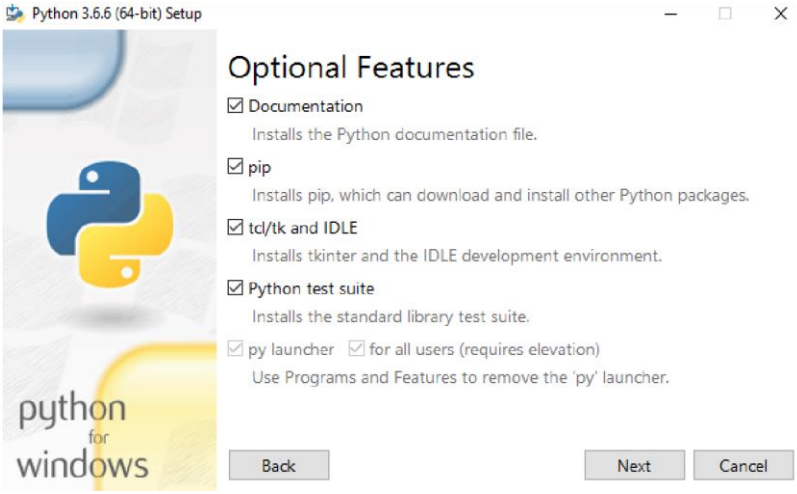
Oczywiście powinniśmy zacząć od instalacji najnowszej wersji Pythona!

Jak w tytule, użyjemy wersji 3.7, która może być pobrana z oficjalnej strony Python. Wybierz odpowiednią wersję, zgodną z twoim systemem operacyjnym i rozpocznij proces pobierania. Uruchom pobrany plik i podążaj według instrukcji.

Wybierz **Customize installation** (Rysunek 3).

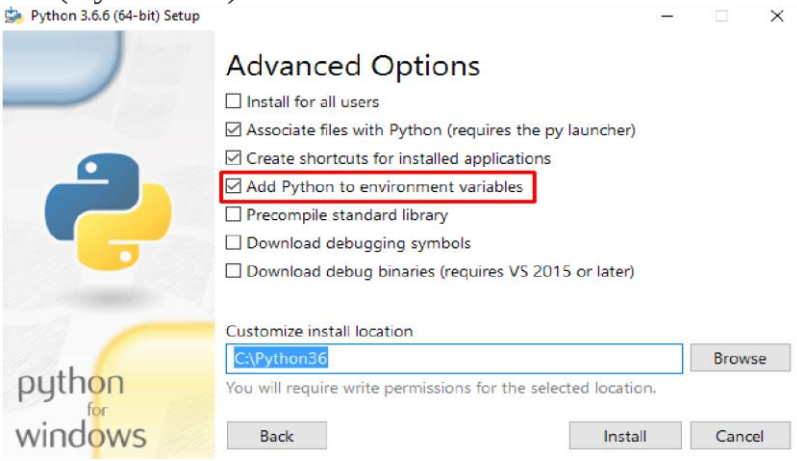
Rysunek 3

Następnie kliknij **Next** i przejdź do następnego kroku (Rysunek 4 na stronie 6).



Rysunek 4

Sprawdź czy na pewno zaznaczyłeś **Add Python to environment variables** a w polu **Location**, wybierz ścieżkę instalacji (zostaw domyślną lub ustaw preferowaną przez Ciebie). Kliknij **Install** i zaczekaj aż instalacja dobiegnie końca (Rysunek 5).



Rysunek 5

Po instalacji folder **Python 3.7** doda się do menu start. Otwórz folder i uruchom **IDLE (Python 3.7)**. Zauważ że nazwa będzie się różnić w zależności od wersji. Otwórz **Python Shell**, gdzie możesz testować kod (Rysunek 6).

```
64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>>
```

Rysunek 6

Przejdź do zakładki **File** > **New File** aby stworzyć nowy plik z rozszerzeniem `.py`. Wciśnij **F5** aby uruchomić.

W chwili wykonywania kodu otworzy się **Python Shell** gdzie możesz wprowadzić dane wejściowe i wyświetlić dane wyjściowe. Znak `>>>` oznacza, że program się zakończył. Aby uruchomić program ponownie, musisz wrócić do okna w którym jest napisany i wcisnąć **F5**.

Nie jest wygodnie używać domyślnego środowiska IDLE, ponieważ nie wykonuje ono automatycznego domykania nawiasów i ustawiania odstępów

Wcześniej wielu programistów pisało swoje programy np. w notatniku. Aczkolwiek nie jest to dobre rozwiązanie, ponieważ nie mamy żadnych podpowiedzi o popełnionych błędach i musimy ich sami pilnować. A co jeśli kod jest długi? W tym przypadku to wymaga dużej ilości czasu. **IDE** pomaga nam zaoszczędzić ten czas ponieważ:

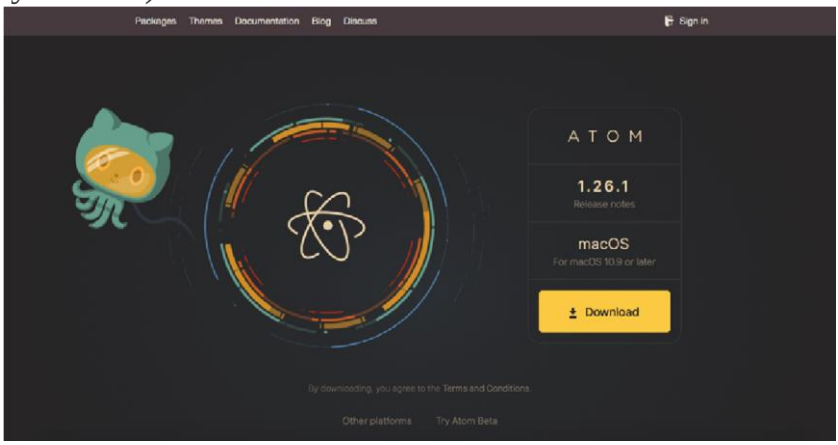
- wyświetla błędy;
- poprawia komendy;
- pokazuje wskazówki.

Przejdziemy teraz do instalacji bardziej przyjaznego środowiska.

Atom

Oczywiście potrzebujemy oprogramowania do efektywnego pisania kodu w języku Python - **Atom**. Ma wiele wbudowanych narzędzi do pracy z kodem i zapoznamy się z nimi w miarę pisania programów.

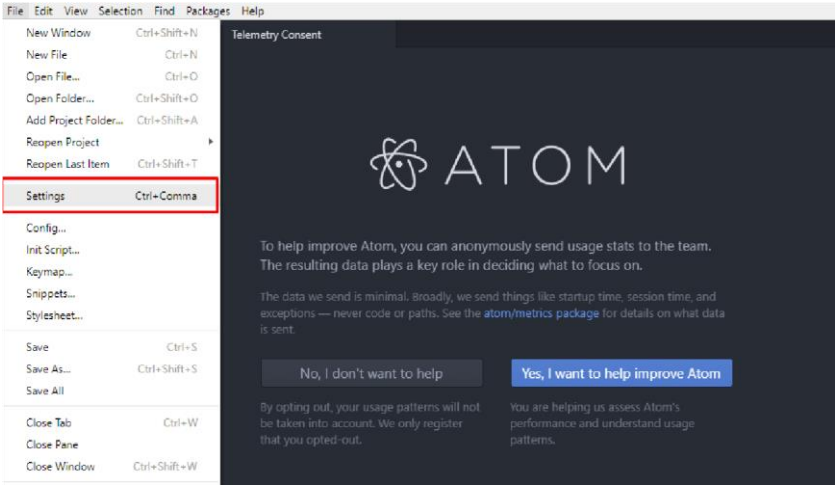
Pobierz **Atom** z oficjalnej strony i uruchom instalator (Rysunek 7).



Rysunek 7

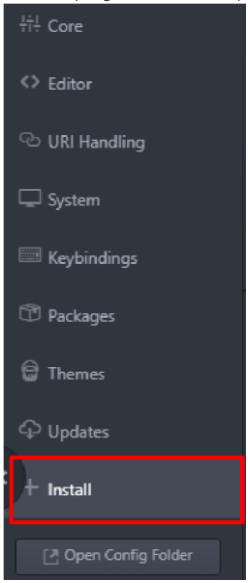
Otworzy się instalator środowiska **Atom**. Teraz musisz ustawić pewne funkcje aby móc pracować i pisać kod.

Wybierz **File** i otwórz **Settings** jak na Rysunek 8 (strona 9). W oknie **Settings** możesz skonfigurować środowisko, zobaczyć zainstalowane dodatki, zmienić wygląd, czcionkę, jej rozmiar i więcej).



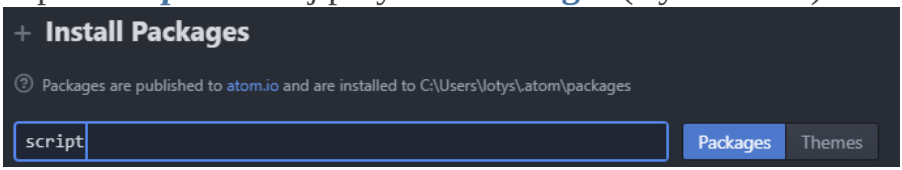
Rysunek 8

Wybierz **Install** aby zobaczyć wszystkie dostępne do zainstalowania dodatki(Rysunek 9).



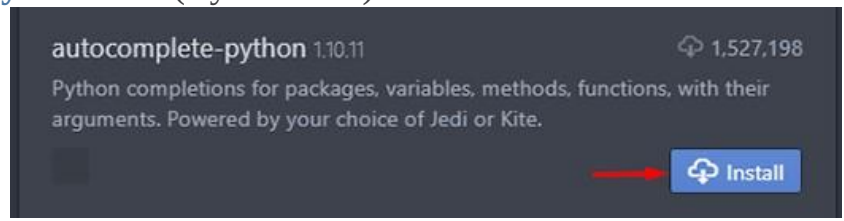
Rysunek 9

Wpisz ***script*** i kliknij przycisk **Packages** (Rysunek 10).



Rysunek 10

Opcjonalnie przejdź do listy dodatkowych plug-in i wybierz **autocomplete-python** (Rysunek 11) i **atom-python-run** (Rysunek 12).



Rysunek 11

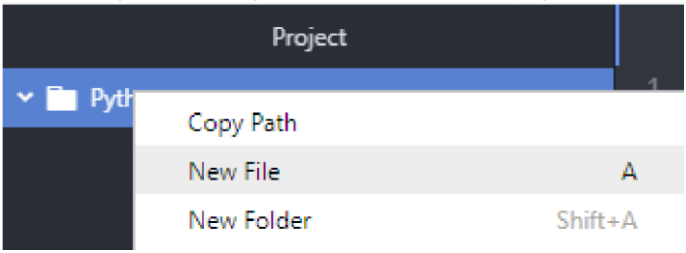
Rysunek 12

Po udanej instalacji, środowisko jest gotowe do poprawiania naszych błędów i uruchamiania skryptów przez wciśnięcie przycisku **F5**.

Twój pierwszy program

Napiszmy teraz najprostszy program i zobaczmy jak działa w środowisku **Atom**.

Kliknij **File**, wybierz **Add project folder** i określ ścieżkę gdzie powinien zostać zapisany. Kliknij prawym przyciskiem myszki i wybierz **New File** (Rysunek 13).



Rysunek 13

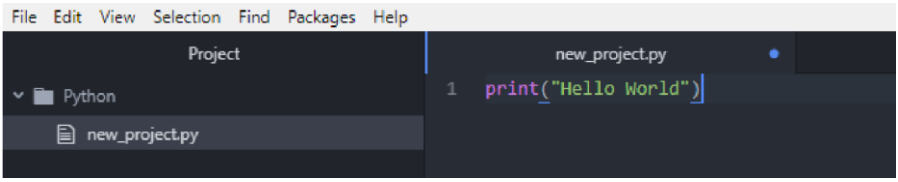
Nadaj mu nazwę i nie zapomnij dodać rozszerzenia `.py` na końcu nazwy (Rysunek 14).

Rysunek 14

Wspaniale! Teraz napiszmy nasz pierwszy kod. Będzie zawierał tylko jedną linię. Jest to standardowa komenda dla wszystkich komend i zawiera dwa słowa: `Hello World`.

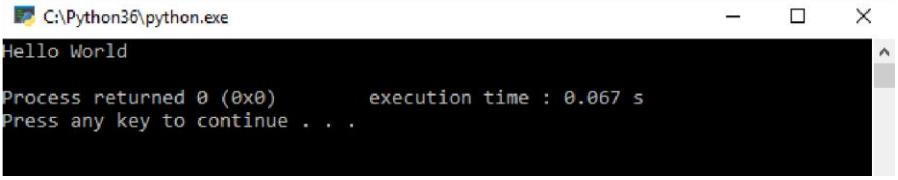
```
print("Hello World")
```

Tak powinno to wyglądać w Twoim programie (Rysunek 15).



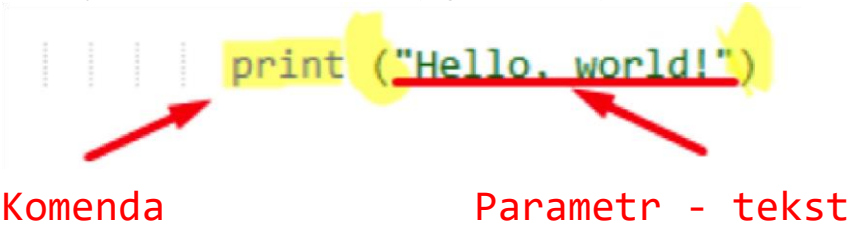
Rysunek 15

Uruchom program wciskając **F5**. Pojawi się nowe okno w którym zobaczymy wynik wykonania naszego kodu (Rysunek 16).



Rysunek 16

Po wykonaniu tego zobaczmy co tak naprawdę napisaliśmy. Komenda **print** wyświetla zawartość w konsoli (parametry są niezbędne dla prawie każdej komendy). Tekst może być w różnych językach ale musi się zawierać pomiędzy dwoma znakami „” (Rysunek 17).



Rysunek 17

Czasami musimy użyć dodatkowych **modułów (bibliotek)**. W tej lekcji nauczymy się czym są, a w następnych lekcjach omówimy je dokładniej i użyjemy w kodzie.

Prawdopodobnie wiesz czym są biblioteki z prawdziwego życia. W języku programowania jest to **ogólnodostępny kod dla innych użytkowników**. Często jest to wygodne rozwiązanie, ponieważ nie musimy pisać formuły na coś, co zostało już stworzone.

Aby zaimportować bibliotekę w Python, trzeba napisać słowo `import` na samym początku kodu i sprecyzować nazwę biblioteki. Możesz dodać ich kilka w zależności od Twoich potrzeb. Jakie są moduły i do czego służą? Możesz się dowiedzieć z oficjalnej strony Python Documentation contents.

Spróbujmy zaimportować bibliotekę, która pomoże nam wyświetlić aktualną datę i czas:

```
import datetime
print(datetime.datetime.now())
```

Zauważ że najpierw zaimportowaliśmy bibliotekę, a potem użyliśmy jej funkcji. Wszystko jest opisane w dokumentacji Python, zaleca się aby z niej skorzystać w razie pytań.

Dane Wejściowe i Wyjściowe

Nauczmy się teraz jak pisać nasze programy. Użyjemy do tego poznanej już komendy `print()` i nowej dla nas komendy `input()`.

Funkcja print()

Popracujmy z komendą `print()` i zauważmy jej właściwości, których jeszcze nie poznaliśmy.

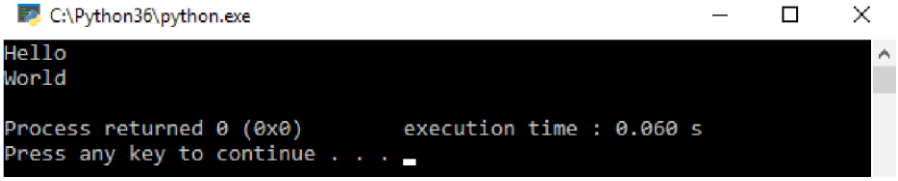
Rozważmy przypadek, w którym tekst powinien być umieszczony w dwóch różnych linijkach :

```
print("Hello")
print("World")
```

Zauważ, że powyższe rozwiązanie jest niewygodne i co jeśli chcemy stworzyć więcej niż dwie linie? Możemy łatwo zapisać `Hello World` w dwóch liniach, rozdzielając je wyrażeniem `\n`. **Znaku** (`\n`) używamy do przerwania linii. Teraz tekst będzie wyświetlany w dwóch liniach jak przy pomocy dwóch funkcji `print()`:

```
print("Hello\nWorld")
```

Oba sposoby dają ten sam wynik (Rysunek 18).



Rysunek 18

Teraz rozważmy sposoby tworzenia tekstu w jednej linii. Aktualnie jest na to kilka sposobów. Użyj do tego przecinka, a wyrazy będą oddzielone spacją.

Jak widać nie ma tu nic trudnego:

```
print("Hello", "World")
```

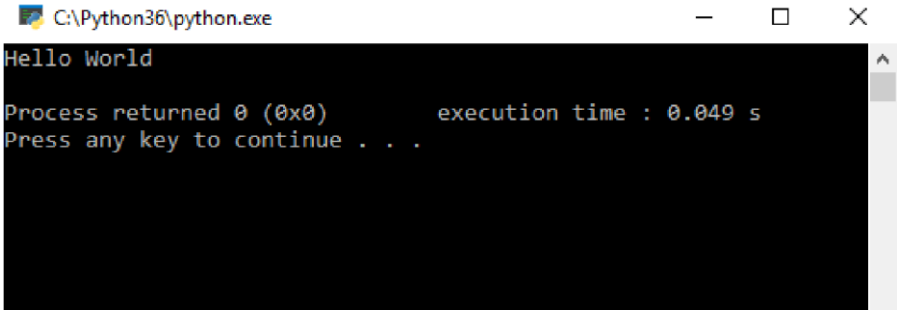
Teraz użyjemy znaku + aby dodać dwa napisy:

```
print("Hello" + "World")
```

Parametr `end=' '` pozwala zastąpić koniec linii spacją:

```
print("Hello", end=' ') print("World")
```

Powyższe sposoby dają ten sam rezultat (Rysunek 19).



Rysunek 19

Pomyślmy co zrobić jeśli chcemy zmienić wartości. W tym wypadku łatwiej jest używać opcji formatowania kodu.

Spójrzmy na najprostszą metodę:

```
a = "World"
print("Hello {}".format(a))
```

Działa to tak: najpierw dodajemy tekst `"Hello {}"`, znak `{}` oznacza, że w tym miejscu pojawi się wartość metody `format()`. W kodzie literka „a" jest zmienną do której przypisujemy wartość.

Użyj tej metody dla kilku argumentów i umieść je w różnych miejscach:

```
a = 1
b = 0
print(f"Pupils = {a} Students={b}")
```

Rezultatem będzie:

```
Pupils = 1 Students = 0.
```

Rozważmy parametr `sep`, który jest separatorem. Kiedy używamy `sep=",  "`, każdy z separatorów będzie oddzielony przecinkiem, z wyjątkiem ostatniego:

```
print("small", "medium", "large")
print("small", "medium", "large", sep="")
print("small", "medium", "large", sep=", ")
```

Funkcja input()

Funkcja `input()` jest odpowiedzialna za wprowadzanie danych. Wymusza w programie oczekiwanie na wprowadzenie znaku przez użytkownika.

Spójrzmy na najprostszy przykład, gdzie prosimy o podanie swojego imienia i mówimy tej osobie „Hello”:

```
name = input("Your name: ")
print("Hello, " + name)
```

Tekst wpisany w funkcji `input()` zostanie wyświetlony w konsoli, po tym użytkownik musi wprowadzić tekst i kliknąć **Enter**. W naszym kodzie `name` jest zmienną, która przechowuje tekst.

Zmienne i operacje na zmiennych

Język programowania Python posiada wiele arytmetycznych operacji, których będziemy potrzebować. W naszym programie może mnożyć, dzielić, dodawać, odejmować czy też podnosić do potęgi. Używamy do tego dobrze znanych nam operatorów matematycznych.

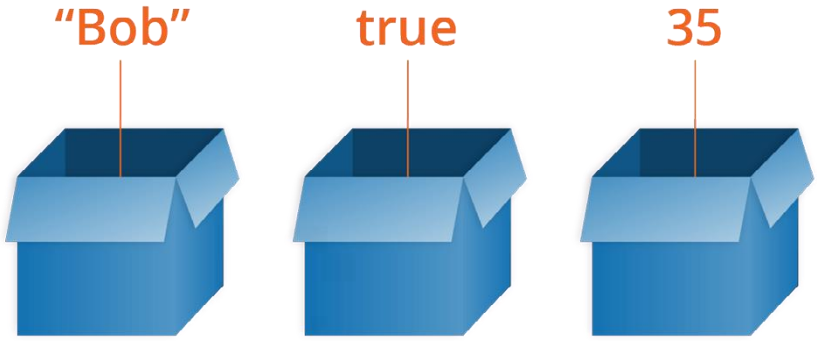
Zapoznaliśmy się już z użyciem funkcji `print()` teraz jest czas aby wyświetlić wyniki najprostszych operacji.

Operator	Nazwa	Przykład	Wynik
+	Dodawanie	`print(8+2)`	10
-	Odejmowanie	`print(8-2)`	6
*	Mnożenie	`print(8*2)`	16
**	Podnoszenie do potęgi	`print(8**2)`	64

Operator	Nazwa	Przykład	Wynik
/	Dzielenie	`print(8/2)`	4.0
//	Pierwiastkowanie	`print(8//2)`	4
%	Reszta z dzielenia	`print(8%2)`	0

Zauważ że używaliśmy ciągle tych samych liczb (8 i 2). Wyobraź sobie że masz dużo dłuższy program i nagle musisz zmienić 8 na 10. Używając tego sposobu, będziesz musiał odnaleźć każdą wartość i ręcznie ją zmienić. Zobacz ile to by było pracy!

Aby ułatwić sobie to zadanie, możemy wykorzystać zmienne. Będziemy ich używać mniej lub więcej ale są potrzebne praktycznie w każdym programie. **Zmienne są nazwanymi komórkami, które przechowują wartość**. W stylu pudełek, do których wrzucamy wartości (liczby, słowa), a potem możemy je z nich wyjmować lub zmieniać. Aby nie pomylić się co mamy w którym pudełku, nadajemy im nazwy (Rysunek 20).



Rysunek 20

Wysuwamy wniosek, że podczas pracy ze zmiennym musimy nadać im nazwę i przypisać do nich wartość. Ważne jest aby zapamiętać, że nazwy zmiennych nie mogą się zaczynać cyfrą. Akceptowalne jest używanie dużych i małych liter, znaku podkreślenia, numerów za literami i słów.

```
a = 8

b = 2
print(a+b)
```

Zmienne mogą być różnego **typu: liczba**, **tekst, lista**, itp.

Typ zmiennej	Nazwa	Przykład
`int`	Integer – liczba całkowita	-150 0 13
`float`	Floating-point numer – liczba zmiennoprzecinkowa	-12.0 1.1 150.5
`str`	String – tekst	"Hello" "My name is.."
`bool`	Logical data type – zmienne logiczne	True False

Popularne błędy

Python ma prostą składnie, lecz mimo to posiada swoje reguły, których musimy przestrzegać

1. **Komentarze są częścią kodu, która nie wpływa na jego wykonanie** i są używane jako notatki do programu. Być może Ty i Twoi przyjaciele będziecie tworzyć duży kod i pewnego dnia zajdzie potrzeba edycji jednej z jego sekcji.

   Komentarze są dodawane po to, żeby nie musieć pamiętać za co jest odpowiedzialna każda linijka kodu. Komentarze mogą być jednoliniowe lub wieloliniowe. Jednoliniowe zaczynają się znakiem `#`:

```
# comments
```

Wieloliniowe wyglądają w ten sposób `'''...'''`:

```
''' comments
comments
comments
'''
```

Pisanie komentarzy wieloliniowych ze znakiem `#` jest błędem. W tym przypadku program nie zadziała prawidłowo, ponieważ uzna komentarz za komendę.

2. **Nazwy zmiennych muszą być tworzone zgodnie z zasadami.** Nie używaj dużych liter, rozdzielaj słowa podkreślnikiem, nie używaj spacji i innych symbolów pomiędzy słowami.

Poprawnie	Błędnie

`variable`
`variable1`
`my_variable`
	`1variable my-`
	`variable my`
	`variable`

3. **Średnik na końcu linijki nie jest wymagany.**

```
a = 8
b = 0.8
c = "string"
d = True
```

4. **Instrukcje są grupowane w bloki w zależności od ilości wcięć**. Używaj **tabulatora** aby tworzyć bloki kodu (**cztery spacje**). W IDLE, są one tworzone automatycznie po klawiszu **Enter** lub wciśnięciu **Tab**. Jest to bardzo ważne. Nie próbuj ustawiać tabulacji za pomocą spacji, możesz spotkać się z błędami.

Poprawnie	Błędnie
`print(a)` `print(b)`	`  print(a)` `print(b)`

5. **Częstym błędem jest niedomykanie cudzysłowu w komendzie** `print()`. Należy pamiętać, że na koniec tekstu musimy go „domknąć”. To samo dotyczy nawiasów.

Poprawnie	Błędnie
`print("Hello World")`	`print("Hello World)`

6. **Kolejnym błędem jest niedopasowanie typu zmiennych.** Zrozumieliśmy już że są różne typy zmiennych. Jeśli dodamy l. całkowitą (`int`) i tekst (`str`), pojawi się błąd w konsoli.

Poprawnie	Błędnie

`a = "Happy"`
`b = " New Year"`
`print(a + b)`
	`a = "Happy New Year"`
	`b = 2020`
	`print(a + b)`

7. **Pamiętaj – zanim użyjesz zmiennej musisz ją stworzyć (zadeklarować).** Inaczej pojawi się błąd.

Poprawnie	Błędnie

`a = 8`
`b = 22`
`print(a+b)`
	`a = 8`
	`print(a+b)`

8. **Ważne jest aby rozróżniać `=` i `==` są to dwa różne znaki**. Pierwszy z nich oznacza, że wartość po prawej stronie zostanie przypisana do zmiennej po lewej stronie (`a = 5`). Drugi z nich porównuje wartości z lewej strony do wartości po prawej stronie..

Poprawnie	Błędnie
`a = 8 b = 22` `print(a+b)`	`a == 8 b == 22` `print(a+b)`

Lekcja # 1
Instalacja Atom
Dane wejściowe i wyjściowe