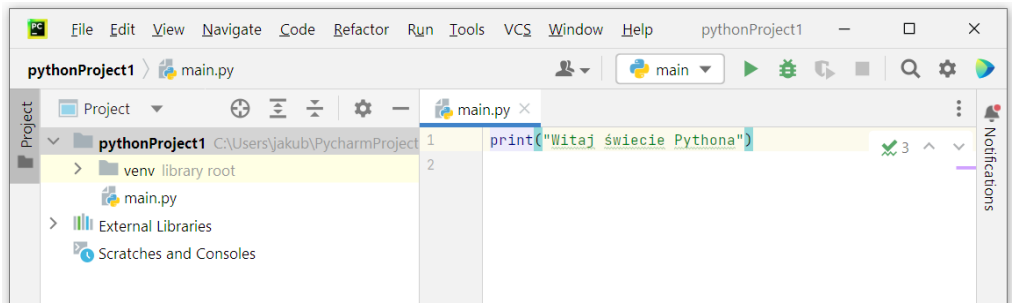
Wstęp do programowania



Lista 1 gr. 3

opracował dr inż. Jakub Długosz

Celem tej listy jest zaznajomienie z dystrybucjami Pythona Anaconda i/lub PyCharm, łańcuchami znakowymi, działaniem funkcji `print()` oraz wykorzystaniem `zmiennych`.

Lista nie na ocenę.

Uwaga 1

Dystrybucję Pythona Anaconda ze zintegrowanym środowiskiem programistycznym (ang. IDE) Spyder można pobrać ze strony https://www.anaconda.com/products/individual.

Uwaga 2

Dystrybucję PyCharm ze zintegrowanym środowiskiem programistycznym (ang. IDE) można pobrać ze strony jetbrains.com/pycharm/.

Uwaga 3

Opis wbudowanej w Pythona funkcji `print()` ze standardowej biblioteki Pythona znajduje się na stronie https://docs.python.org/3/library/functions.html#print .

Zadanie 1 (Dodanie myślników i kropki)

Przypomnij sobie czym są `pozycyjny` oraz `za pomocą słów kluczowych` sposoby przekazania argumentów funkcji `print()`. Za pomocą jednego wywołania funkcji `print()` (jednej inwokacji funkcji `print()`) oraz czterech argumentów pozycyjnych (`"Nazywam"`,`"sie"`,`"Monty"`,`"Python"`) wyświetl w konsoli napis `Nazywam-sie-Monty-Python,` tzn.

Uzupełnij kod `print("Nazywam","sie","Monty","Python", ...)`, tak aby w konsoli wyświetlił się napis `Nazywam-sie-Monty-Python.`

Uwaga: Na końcu wyświetlonego w konsoli napisu `Nazywam-sie-Monty-Python.` powinna znajdować się kropka (.).

Zadanie 2 (Świecie w cudzysłowie)

Za pomocą funkcji `print()` wyświetl w konsoli dwoma różnymi sposobami napis `Witaj "świecie" Pythona.`

Uwaga: Słowo `świecie` ma być wyświetlone na ekranie w cudzysłowie.

Zadanie 3 (Wiele wierszy w cudzysłowie)

Zmodyfikuj w poniższym kodzie tylko znaki cudzysłowu (").

```
print("pierwszy wiersz
 drugi wiersz
  trzeci wiersz")
```

tak aby po uruchomieniu program wyświetlał

```
pierwszy wiersz
 drugi wiersz
  trzeci wiersz
```

Uwaga: Program ma wyświetlać dokładnie takie odstępy i wiersze jak w prezentowanym wyniku.

Zadanie 4 (Jan wielokrotnie)

Nie używając żadnych liter zmodyfikuj kod w miejscu …

```
print("Jan"…)
```

tak aby wyświetlić wyraz `Jan` 12 razy, tzn. aby otrzymać po uruchomieniu

```
JanJanJanJanJanJanJanJanJanJanJanJan
```

Zadanie 5 (Dodawanie)

Pobierz od użytkownika dwie wartości liczbowe i korzystając z f-string wyświetl wynik dodawania.

Program powinien działać jak na poniższym przykładzie:

```
Podaj pierwszy składnik dodawania: 4
Podaj drugi składnik dodawania: 7
Wynik dodawania 4 + 7 to 11.
```

Zadanie 6 (Formatowanie wyświetlanych wartości liczbowych)

Wykorzystaj kod z dodatku D1. Sprawdź jakie są typy danych dla zmiennych liczba1, liczba2, liczba3, liczba4. Dodaj opcje formatujące w funkcji print(), by wyświetlić wartości liczbowe z dokładnością do 3 miejsc po przecinku.

Wynik działania programu:

```
3.142
3.000
7.120
5.000
```

Zadanie 7 (Formatowanie łańcuchów znakowych)

Wykorzystaj kod z dodatku D2. Dodaj opcje formatujące w funkcji print(), by wyświetlić czytelnie dane.

Wynik działania programu:

```
Osoba                Stanowisko                Pensja
Anna Cis             kierowniczka działu HR    7500
Konstanty Mączyński  kierowca                  12000
```

Poza wyrównaniem do lewej przedstaw inne opcje (wyrównanie do prawej, do środka, uzupełnienie symbolami wiodącymi np. znakami *)

Zadanie 8 (Wyświetlanie w kolorze)

Wykorzystaj z sekwencji sterujących ASCII (https://en.wikipedia.org/wiki/ANSI_escape_code) wyświetl łańcuch znakowy jak poniżej (słowo „fajne” zawiera podkreślenie, czcionka jest pogrubiona i koloru czerwonego).

Wynik działania programu:

Programowanie jest fajne.

D1

Kod:

```
liczba1 = 3.14159
liczba2 = 3
liczba3 = 7.12
liczba4 = "5"
print(liczba1)
print(liczba2)
print(liczba3)
print(liczba4)
```

D2

Kod:

```
osoba1 = "Anna Cis"
osoba2 = "Konstanty Mączyński"
stanowisko1 = "kierowniczka działu HR"
stanowisko2 = "kierowca"
pensja1 = 7500
pensja2 = 12000

print("Osoba", "Stanowisko", "Pensja")
print(osoba1, stanowisko1, pensja1)
print(osoba2, stanowisko2, pensja2)
```

D3

Specyfikacja pól zamiany {}:

```
{[<nazwa>][!<konwersja>][:<spec_formatu>]}
```

D4

Specyfikacja `<spec_formatu>`

`:[[<wypełnienie>]<wyrównanie>][<znak>][#][0][<szer_pola>][<grupa>][.<precyzja>][<typ>]`

D5

Sygnatura funkcji `print:`

```
print(p1, p2, p3, …, sep=..., end=...)
p1, p2, p3, … – parametry pozycyjne
sep=…, end=… – parametry z użyciem słów kluczowych
```

D6

Pełna sygnatura funkcji **`print()`** wraz z dokumentacją.

```
def print(self, *args, sep=' ', end='\n', file=None): # known special case of print
    """
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
    """
```