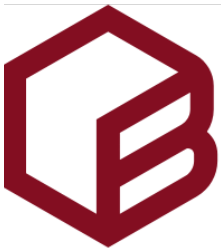	Politechnika Bydgoska im. J. J. Śniadeckich **Wydział Telekomunikacji, Informatyki i Elektrotechniki**		
Przedmiot	Skryptowe języki programowania		
Prowadzący	mgr inż. Martyna Tarczewska		
Temat	*Python - wprowadzenie*		
Student			
Nr ćw.	2	**Data wykonania**	
Ocena		**Data oddania spr.**	

1. Cel ćwiczenia

Celem ćwiczenia jest instalacja oprogramowania oraz poznanie podstaw składni skryptowego języka programowania *Python* przez wykonanie prostych zadań. **Funkcje i zmienne stworzone podczas zajęć powinny być odpowiednio otypowane z użyciem biblioteki typing. Kod powinien być napisany w języku angielskim.**

2. Informacje podstawowe

2.1. Oprogramowanie

Do pracy podczas ćwiczeń niezbędne jest oprogramowanie do uruchamiania skryptów napisanych w języku *Python*. Wersję adekwatną do swojego systemu operacyjnego można znaleźć na stronie: https://www.python.org/downloads/ Z uwagi na dłuższe wsparcie zaleca się korzystanie z wersji 3.

Visual Studio Code https://code.visualstudio.com/docs/languages/python**,**

Jupyter Notebook,

pyCharm, Tonny, Notepad

2.2. Dokumentacja

Dokumentacja języka *Python* jest udostępniona online. Można ją znaleźć pod adresem: https://www.python.org/doc/

2.3. Uruchamianie skryptów

Aby zapisać plik jako uruchamialny skrypt *Pythona*, należy zapisać go jako plik z rozszerzeniem *.py*, np. *testowy.py*. Aby go uruchomić, należy w konsoli wpisać (w zależności od wykorzystywanej wersji oprogramowania):

python testowy.py lub *python3 testowy.py*

2.4. Kody programów

Kody należy pisać wykorzystując język Python i wszystkie jego możliwości (chyba że w zadaniu ustalono inaczej). Należy tworzyć własne funkcje, które później można ponownie wykorzystać.

Uwaga: w sprawozdaniu należy zamieścić kod programu wraz z komentarzami!

3. Podstawowe elementy Pythona

3.1. Komentarze

Komentarz wierszowy tworzy się przed dodanie znaku #. W tym przypadku tekst od znaku # do końca wiersza nie wpływa na działanie programu.

```
# first comment
SPAM = 1
# second comment
# third comment
STRING = "# To nie jest komentarz!"
```

Komentarz blokowy obejmuje dowolny tekst ujęty w potrójne cudzysłowy lub potrójne apostrofy.

```
# To jest komentarz jednoliniowy
print("Hello, World!")  # Ten komentarz jest obok instrukcji

# Komentarze można umieszczać na początku lub końcu linii

"""
To jest wieloliniowy komentarz
Można w nim zawrzeć wiele linii tekstu
"""

def example_function() -> None:
    """
    Ta funkcja nic nie robi.
    Jej głównym celem jest posiadanie przykładowego wieloliniowego komentarza.
    """
    pass
```

3.2. Shebang line

Linia ta zaczyna się od znaków #! i określa ścieżkę do interpretara, który ma wykonać dany skrypt. To powinna być pierwsza linia każdego skryptu.

```
#!/usr/bin/python3
```

3.3. Importy

Python bardzo często korzysta z bibliotek. Aby użyć funkcji zaimplementowanych w zewnętrznych bibliotekach, należy je zaimportować, a potem wykorzystać z podaniem

nazwy biblioteki lub aliasu wprowadzonego słowem kluczowym *as*. Można importować tylko część funkcji z biblioteki, do czego służy słówko kluczowe *from*.

```
import datetime
import calendar as cal
from math import pi

cal.month(y, m)
datetime.datetime.now()
```

3.4. Standardowe wejście/wyjście

Aby skorzystać z danych podanych przez użytkownika, należy wykorzystać polecenie *input*. Do wypisywania danych w konsoli służy polecenie *print*.

```
name = input("Input your First Name : ")
print("Hello, : " + name)
```

3.5. Przykładowy plik wykonywalny

Poniżej przedstawiono przykładowy szablon pliku Python. Pierwszą linią jest poznana już shabang line. Następnie mogą pojawić się importy. Po nich powinny zostać zadeklarowane zmienne i stałe globalne (takie z których będą korzystały wszystkie lub kilka funkcji). Następnie pojawiają się nasze własne funkcje. Dobrze jest tworzyć własne funkcje (np. jedna funkcja na zadanie), gdyż potem kod można ponownie wykorzystać. Prostsza jest też nawigacja po takim pliku. Po własnych funkcjach należy umieścić definicję funkcji main, od której rozpocznie się wykonywanie takiego pliku.

W Pythonie nie ma znanych z C nawiasów klamrowych do oznaczania początku i końca funkcji. **Tutaj kluczowe są wcięcia.**

```
#!/usr/bin/python3

# stałe i zmienne globalne
CONSTANT_1: float = 0.3
CONSTANT_2: str = "value"
variable_1: int = 5

# definicja funkcji, w nawiasie możemy podać parametry
def display_message() -> None:
     """ Opis funkcji, np. wyświetlany przez punkcję help() """
     # ciało funkcji
     print('Calling from display_message')
     # zwracana wartość nie jest potrzebna, ale dodajemy typowanie

# funkcja main - jeśli moduł ma być wykonywalny - funkcja od której się
zaczyna wywoływanie
def main() -> None:
   # jeśli funkcja wykorzystuje zmienne globalne, muszą być one oznaczone
w następujący sposób
```

```
    global variable_1
    variable_1 = 7
    # zmienna lokalna - po zakończeniu działania funkcji ta zmienna jest
usuwana
    local_variable: bool = True
    # wywołanie innych funkcji
    display_message()

# na końcu modułów (plików) które mają być wykonywalne
if __name__ == '__main__':
    main()
```

3.6. Słowa kluczowe

Jak każdy język programowania, Python ma listę słów kluczowych, których nie można używać m.in. jako nazwy naszych stałych czy zmiennych. Przykłady słów kluczowych:

False	return	try	while
None	global	with	def
True	import	from	finally
if	elif	else	continue

3.7. Wbudowane typy danych

Podstawowymi typami danych są: zmienne liczbowe, tekstowe, binarne oraz logiczne. W Pythonie nie mamy jednak konieczności deklarowania typów. W poniższym zapisie zmienna *a* automatycznie otrzyma typ *integer*. Typ zmiennej można sprawdzić poleceniem *type*.

```
a = 10
print(type(a))
```

3.8. Konwersja typów

Czasem potrzebna będzie zmiana typu danej na innych (np. z liczbowego na tekst). Do tego służy mechanizm jawnej konwersji typów. Szczególnie przydatna jest konwersja na napis.

```
b = 10.10
print(type(b)) #float
c = int(b)
print(type(c)) #int
print("to jest liczba a: " + str(a)) #konwersja do string
```

3.9. Podstawowe operatory

Do wykonywania działań matematycznych wykorzystuje się operatory. W tablicy przedstawiono podstawowe operatory wraz z ich opisem i sposobem działania. Można też skorzystać ze skróconego zapisu *a += 2*, który oznacza dodanie 2 do zmiennej *a*.

Operator	Opis	Przykład
+	Dodawanie	1 + 2 = 3
-	Odejmowanie	4 – 1 = 3
*	Mnożenie	2 * 2 = 4
/	Dzielenie	1 / 4 = 0.25
//	Dzielenie całkowite	13 // 5 = 2
%	Reszta z dzielenia	13 % 2 = 1
**	Potęgowanie	2 ** 8 = 256

4. Przebieg ćwiczenia

4.1. Zadanie 1.

Stworzyć szablon pliku (komentarze, shebang line, miejsce na importy, zmienne i stałe globalne itp.). Napisać funkcję, która wypisze napis „Hello World!”. Wywołać ją z funkcji *main*.

4.2. Zadanie 2.

Napisać funkcję, która wyświetli wersję Pythona, z której korzystamy. Wywołać ją z funkcji *main*.

Wskazówka: czy nie trzeba dodać jakiegoś importu?

4.3. Zadanie 3.

Napisać funkcję, która wyświetli aktualną datę i godzinę. Wykorzystać sposób formatowania wyjścia *strftime* i zaproponować 3 różne formaty daty i godziny.

Wskazówka: tutaj też potrzeba importu! Należy nadać alias (słówko as).

4.4. Zadanie 4.

Napisać funkcję, która oblicza obwód i pole powierzchni koła na podstawie podanego promienia.

Wskazówka: wartość liczby PI jest dostępna w bibliotece *math*. Należy użyć słowa kluczowego *from*.

4.5. Zadanie 5.

Zainicjalizować różnymi wartościami zmienną globalną *a* oraz stałą globalną *A*. W nowej funkcji zainicjalizować zmienną lokalną *a* i nadać jej jeszcze inną wartość. Jaką wartość ma *a* w tej funkcji? Jaką wartość ma *A* w tej funkcji? Czy interpreter rozróżnia *A* i *a* ? Udzielić odpowiedzi na pytania w sprawozdaniu. Sprawdzić typ *A* i *a* korzystając z polecenia *type*.

4.6. Zadanie 6.
Napisać funkcję, która pobiera od użytkownika imię i nazwisko, a następnie zwraca je w odwrotnej kolejności.

4.7. Zadanie 7.
Napisać funkcję, która na podstawie podanych przez użytkownika wartości obliczy pierwiastki równania kwadratowego. Dla uproszczenia można założyć, że podane wartości są poprawne i równanie zawsze będzie miało 2 rozwiązania.

Wskazówka: która z wcześniej importowanych bibliotek wydaje się przydatna?

4.8. Zadanie 8.
Napisać funkcję, która na podstawie zadeklarowanych w kodzie programu dat obliczy różnicę dni między nimi.

Wskazówka: która z wcześniej importowanych bibliotek wydaje się przydatna?

4.9. Zadanie 9.
Napisać funkcję, która na podstawie roku i miesiąca zwróci dla niego kalendarz.

Wskazówka: która z wcześniej importowanych bibliotek może się przydać?

4.10. Zadanie 10.
Napisać funkcję, która pobierze od użytkownika dwie liczby. Wykonać 7 podstawowych działań z tabelki (punkt 3.9). Jaki wynik otrzymujemy przy dzieleniu przez 0?

4.11. Zadanie 11.
Napisać funkcję sprawdzającą kolejność wykonywania działań w Pythonie:

a) czy mnożenie i dzielenie wykonuje się od prawej czy lewej strony?

b) co wykona się wcześniej: mnożenie czy dodawanie?

c) jak nawiasy wpływają na kolejność wykonywania działań?

d) co wykona się wcześniej: mnożenie czy potęgowanie?

4.12. Zadanie 12.
Napisać funkcję, która sprawdzi typy zmiennej będącej wynikiem dzielenia, dzielenia całkowitego i reszty z dzielenia:

a) dwóch liczb całkowitych

b) liczby całkowitej i zmiennoprzecinkowej

c) dwóch liczb zmiennoprzecinkowych.

5. Sprawozdanie

Sprawozdanie z laboratorium powinno zawierać:

- wypełnioną tabelę z początku instrukcji,
- kody programów będących rozwiązaniami wszystkich zadań wraz z komentarzami,
- demonstrację działania programu,
- odpowiedzi na pytania zawarte w zadaniach,
- wnioski.