Wydział: **Zarządzania i Modelowania Komputerowego**
Katedra Technologii Informatycznych
Przedmiot: **Technologie informacyjne**
Rok **I**

PYTHON - ĆWICZENIE 2

1. Formatowanie wyników.

Wyprowadzając wyniki obliczeń przy użyciu instrukcji **print** możemy wykorzystać jej dodatkowe możliwości.

print (lista-wyjścia,sep=' ',end='\n')

Parametr **sep** domyślnie jest pojedynczą spacją, parametr **end** to domyślnie zmiana wiersza (nowa linia). Przykładowo:

```
>>> x=2
>>> print(x,2*x,3*x,sep='_')
2_4_6
```

Wykorzystanie parametru **end** przydaje się w skrypcie, gdy np. jedna linia wydruku generowana jest dwiema instrukcjami **print**. Po uruchomieniu skryptu:

```
File  Edit  Format  Run  Options  Window  Help
from math import sqrt
print('wynik obliczeń:', end=' ')
x=sqrt(5)-2.0
print(x)
```

na ekranie pojawi się:

```
========== RESTART: F:\PYTHON\test_print.py
wynik obliczeń: 0.2360679774997898
>>>
```

Dodatkową możliwość formatowania wyników zapewnia zastosowanie łańcucha formatującego, czyli instrukcji w postaci:

print ("łańcuch formatujący" % (lista wyników))

Za pomocą parametru **łańcuch formatujący** należy przygotować projekt wydruku, wstawiając w miejsca, w których mają pojawić się wartości wyprowadzanych wyrażeń, opis pola przeznaczonego na te wartości. Opis pola zależy od typu wyprowadzanej wartości:

%s – gdy wyprowadzany jest ciąg znaków
%c – gdy wyprowadzany jest jeden znak
%i – gdy wyprowadzana jest liczba całkowita
%f – gdy wyprowadzana jest liczba rzeczywista
%e – gdy wyprowadzana jest liczba w postaci wykładniczej

Między znakiem % a symbolem literowym można określić szerokość pola, jakie ma zajmować wyprowadzana wartość. Jeśli szerokość pola zostanie pominięta, zostaną zastosowane szerokości domyślne.

```
>>> a=2
>>> x=3.5
>>> print('a=%2i x=%f 2*x=%6.2f' % (a,x,2*x))
a= 2 x=3.500000 2*x=  7.00
>>> print('a=%2i x=%e 2*x=%6.2e' % (a,x,2*x))
a= 2 x=3.500000e+00 2*x=7.00e+00
>>>
```

Kolejny przykład pokazuje, że łańcuch znaków (napis) wyrównywany jest do prawej krawędzi zarezerwowanego pola. Widać z niego także, że w przypadku, gdy lista wyników jest jednoelementowa, można pominąć obejmujące ją nawiasy

```
>>> print('%s' % 'zima')
zima
>>> print('%12s' % 'zima')
        zima
>>>
```

2. Operatory porównań i wyrażenia logiczne.

W trybie interaktywnym przetestuj działania operatorów relacyjnych i logicznych:

operatory relacyjne

- **<** mniejsze
- **<=** mniejsze lub równe
- **>** większe
- **>=** większe lub równe
- **==** równe
- **!=** różne

operatory logiczne

- **not** negacja
- **or** alternatywa
- **and** koniunkcja

```
>>> 10>8
True
>>> 5<=5
True
>>> 2==1
False
>>> 1>2>3
False
>>> 2>=1>4
False
>>> 3<5>2
True
>>> 'zima'>'lato'
True
>>> not 2==1
True
>>> 2>=2 or 3==1
True
>>> 3==5 and 2>1
False
>>> not 3>4 or 2>3 and 4!=2
True
>>>
```

Uwaga:

- nie wolno mylić operatora przypisania (=) z relacją równości (==)
- w pierwszej kolejności wykonywane są operatory porównania, później operator negacji, następnie iloczynu logicznego, a na końcu sumy logicznej (takie same operatory wykonywane są w kolejności od lewej do prawej)

3. Instrukcja warunkowa.

Operatory relacyjne i logiczne wykorzystywane są przybudowie wyrażeń logicznych, niezbędnych do tworzenia, przy pomocy instrukcji warunkowej (**if**),.rozgałęzień w programach

Najprostsza postać instrukcji warunkowej jest następująca:

```
if warunek:
    instrukcja1
    instrukcja2
    …
```

Działanie instrukcji jest następujące: jeśli spełniony jest ***warunek*** to wykonywany jest blok instrukcji poniżej warunku (bloki instrukcji w Pythonie wyróżniane są za pomocą wcięcia - tabulatora), w przeciwnym przypadku ten blok instrukcji jest pomijany – następuje przejście do instrukcji poniżej, zapisanej z takim samym wcięciem jak pierwsza linia instrukcji **if.**
W przypadku, gdy blok ogranicza się do pojedynczej instrukcji wolno zapisać ją bezpośrednio po dwukropku za warunkiem, w tej samej linii.

Instrukcja warunkowa może mieć postać:

```
if warunek:
    instrukcja11
    instrukcja12
    …
else:
    instrukcja21
    instrukcja22
    …
```

Wtedy w przypadku gdy ***warunek*** jest prawdziwy, wykonywany jest blok instrukcji poniżej warunku, a gdy jest fałszywy, blok instrukcji po **else.**

Czasem zachodzi konieczność rozpatrzenia większej liczby przypadków i wtedy można zastosować instrukcję warunkową w postaci:

```
if warunek1:
    instrukcja11
    instrukcja12
    …
elif warunek2:
    instrukcja21
    instrukcja22
    …
else:
    instrukcja31
    instrukcja32
    …
```

Działanie instrukcji w tej postaci jest następujące: jeśli spełniony jest ***warunek1*** wykonywany jest blok instrukcji poniżej warunku; jeśli ***warunek1*** nie jest spełniony instrukcje tego bloku są pomijane i badany jest ***warunek2***, ten po **elif**. Jego spełnienie powoduje wykonanie bloku po **elif**, w przeciwnym przypadku jego pominięcie. (człon **elif** może wystąpić wielokrotnie). Jeśli żaden z warunków nie jest prawdziwy, wykonywany jest blok instrukcji po **else**.

Przykład 1

Po wczytaniu liczby x ma się pojawić informacja czy liczba jest dodatnia, ujemna czy równa zero. Na zakończenie dołączany ma być tekst ‘koniec’.

Utwórz nowe okienko ***File/NewWindow*** a następnie zapisz pustą aplikację do pliku **badanie.py** (funkcja ***File/Save As***) w swoim katalogu stanowiskowym.
Wprowadź tekst programu i przetestuj jego działanie.

```
File  Edit  Format  Run  Options  Window  Help
x=float(input('podaj liczbę x='))
print('liczba x jest',end=' ')
if x<0:
    print('ujemna')
elif x==0:
    print('równa zero')
else:
    print('dodatnia')
    print('koniec')
```

Dla przypomnienia: parametr **end=' '** w pierwszej instrukcji **print** powoduje, że rezultaty następnej instrukcji **print** będą wyprowadzane w tym samym wierszu.

W tej wersji programu napis 'koniec' wyprowadzany będzie jedynie w przypadku liczb dodatnich (dlaczego?). Jak zmienić program aby komunikat 'koniec' pojawiał się w każdym wariancie?

Przykład 2

Dana jest kwota netto ***x*** (w PLN) oraz stawka podatku ***vat*** (w %). Opracować program, który wczytuje dane ***x*** i ***vat*** a następnie wyznacza i drukuje wartość podatku vat oraz wartość kwoty brutto. W przypadku, gdy stawka ***vat*** wynosi 0 drukuje napis 'BEZ VAT!'.

Wprowadź poniższy program realizujący podane zadanie (**vat.py**) i przetestuj jego działanie.

```
File  Edit  Format  Run  Options  Window  Help
x=float(input('podaj kwotę netto (PLN)  x='))
vat=int(input('podaj stawkę VAT (%)  vat='))
if vat==0:
    print('bez VAT \n kwota brutto = kwota netto=',x)
else:
    print('podatek VAT =', x*vat/100)
    print('kwota brutto  =', x*(1+vat/100))
```

Zadanie 1

Pracownik otrzymuje wynagrodzenie zasadnicze ***w*** oraz dodatek stażowy zależny od lat pracy (***staz***). Dodatek stażowy jest obliczany jako 1% wynagrodzenia za każdy rok pracy, ale w przypadku gdy staż przekracza 20 lat wynosi 20 %. Opracować aplikację (**staz.py**), która dla danych ***w*** oraz ***staz*** wyznacza i drukuje dodatek stażowy oraz wynagrodzenie z dodatkiem stażowym. W przypadku gdy staż=0 drukowany jest tylko tekst 'Brak stażu!'.

Przykład 3

Po wprowadzeniu dwóch liczb rzeczywistych **x** oraz **y** ma się pojawić informacja, w której ćwiartce układu współrzędnych znajduje się punkt o współrzędnych (x,y). Nazwa skryptu: **punkt.py**.

Przykładowe rozwiązanie zostało zamieszczono poniżej. Czy wszystkie możliwości zostały przewidziane? Rozbuduj ten skrypt tak, by w przypadku punktów położonych na osiach pojawiał się odpowiedni komunikat.

```
File  Edit  Format  Run  Options  Window  Help
x = float(input("x="))
y = float(input("y="))
if x > 0:
    if y > 0: print("ćwiartka I")
    else: print("ćwiartka IV")
else:
    if y > 0: print("ćwiartka II")
    else: print("ćwiartka III")
```

Przykład 4

Dana jest funkcja:

$$y=\begin{cases}\sin^2(x) & gdy \quad 0\le x\le \pi \\ 0 & w\ innych\ przypadkach\end{cases}$$

Wprowadź poniższy program (**p4.py**) realizujący podane zadanie i przetestuj jego działanie.

```
File  Edit  Format  Run  Options  Window  Help
from math import sin,pow,pi
x=float(input('podaj x='))
if 0<=x<=pi: y=pow(sin(x),2)
else: y=0
print('dla x=%5.2f y=%8.4f' % (x,y))
```

Przykład 5

Dana jest funkcja:

$$y=\begin{cases}x+1 & gdy \quad -1\le x\le 0 \\ -x+1 & gdy \quad 0< x\le 1 \\ 0 & w\ innych\ przypadkach\end{cases}$$

Wprowadź poniższy program realizujący podane zadanie (**p5.py**) i przetestuj jego działanie.

```
File  Edit  Format  Run  Options  Window  Help
x=float(input('podaj x='))
if -1<=x<=0: y=x+1
elif 0<x<=1: y=-x+1
else: y=0
print('dla x =%5.2f y =%5.2f' % (x,y))
```

Zadanie 2

Opracować skrypty (**z2a.py** oraz **z2b.py**) do wyznaczania wartości funkcji podanych wzorami:

a)

$$y=\begin{cases}\sqrt{x} & gdy \quad x\ge 0 \\ |x| & w\ innych\ przypadkach\end{cases}$$

b)

$$y=\begin{cases}2 & gdy \quad x\le 0 \\ 2-x & gdy \quad 0< x\le 2 \\ 0 & gdy\ x \quad x>2\end{cases}$$

Zadania dodatkowe

Napisz, uruchom i przetestuj skrypty:

- **parzyste.py**, który wczyta liczbę całkowitą i wyświetli informację, czy jest to liczba parzysta, czy nieparzysta.
- **calkowite.py**, który po wczytaniu liczby wyświetla informację, czy jest to liczba całkowita, czy niecałkowita. (wykorzystać funkcję konwersji na typ całkowity **int**(x))
- **rownanie.py**, który wyznacza wszystkie pierwiastki rzeczywiste równania kwadratowego w postaci $ax^2+bx+c=0$, gdzie a, b i c podaje użytkownik; obliczenia powinny być poprzedzone sprawdzeniem, czy wprowadzone równanie jest rzeczywiście kwadratowe.
- **prosta.py,** sprawdzający położenie punktu o współrzędnych (px,py) w stosunku do prostej o równaniu: y=mx+b (na prostej, powyżej prostej, poniżej prostej.