Instrukcja A1, Python I - Podstawy

1) Print, input

Funkcja *input("optional text")* służy do pobierania danych od użytkownika i przekazania go do zmiennej typu *string*. Funkcja przed pobraniem danych może wyświetlić także dowolny tekst podany jako argument funkcji.
Najprostszą funkcją do wyświetlania danych w terminalu jest funkcja *print("tekst1", "tekst2", "tekst3", sep="/n")*. Tekst do wyświetlenia podajemy jako argument funkcji. Do funkcji można także podać kilka argumentów rozdzielając je przecinkiem i podając na końcu opcjonalny argument symbolu separatora.

Proszę stworzyć program, który pobierze od użytkownika imię i nazwisko oraz wiek, zapisze dane do odpowiednich zmiennych i wyświetli wszystko w jednym zdaniu.

```
### zadanie 1 ###
name = input("Podaj swoje imię: ") # pytanie o imię i zapisanie go do zmiennej tekstowej name
surname = input("Podaj swoje nazwisko: ") # pytanie o nazwisko i zapisanie go do zmiennej tekstowej surname
age = input("Podaj swój wiek: ") # pytanie o wiek i zapisanie go do zmiennej tekstowej age


print("Witaj", name, surname, "Twój wiek to", age, "lat")
```

Proszę przerobić powyższy program tak, aby pobierał od użytkownika dane: takie jak kolor, marka samochodu oraz rok urodzenia. Dane podczas wpisywania powinny pojawiać się w nowej linii (proszę wykorzystać znak nowej linii "\n"). Wszystkie proszę wyświetlić na ekranie w jednym zdaniu z użyciem operatora dodawania stringów ("+").

2) Konwersja i typy danych

Proszę stworzyć i uruchomić program, który pobierze od użytkownika dwie liczby i wyświetli ich sumę oraz iloczyn. Co się stało po uruchomieniu?

```
### zadanie 2 ###

number1 = input("Podaj pierwszą liczbę: ") # pytanie o pierwszą liczbę i zapisanie jej do zmiennej tekstowej number1
```

```
number2 = input("Podaj drugą liczbę: ") # pytanie o drugą liczbę i zapisanie jej do zmiennej tekstowej number2

sum = number1 + number2

product = number1 * number2


print("Suma podanych liczb to: " + sum + ", ich iloczyn to:" + product) # wypisanie sumy i iloczynu podanych liczb
```

Po uruchomieniu na ekranie terminala wyświetlił się błąd:

```
Podaj pierwszą liczbę: 23
Podaj drugą liczbę: 23
Traceback (most recent call last):
  File "/Users/grochot/instrukcja1.py", line 17, in <module>
    product = number1 * number2
              ~~~~~~~~^~~~~~~~
TypeError: can't multiply sequence by non-int of type 'str'
```

Jak myślisz dlaczego ?

Konwersja zmiennych (rzutowanie) polega na zamianie zmiennych z jednego typu na inny. Istnieje kilka funkcji konwertujących. Do najpopularniejszych należą *int(), float(), str(), bool()*.

Aby sprawdzić jakiego typu jest dana zmienna używamy funkcji *type()*.

```
data = input("podaj wiek")


print(type(data)) # <class 'str'>
```

Wykorzystując konwersję typów proszę poprawić kod programu, tak aby jego uruchomienie nie generowało błędu *TypeError*.

3) Listy

Niezwykle ważnym typem danych w językach programowania, gdyż pozwalają przechowywać wiele różnego typu danych w jednej zmiennej. Dostęp do danych przechowywanych w liście następuje poprzez odwołanie się do indeksu elementu listy (czyli miejsca, które zajmuje w liście).

- Proszę stworzyć pustą listę o nazwie “list1” oraz drugą listę o nazwie “list2”, w której przechowywane będą trzy dowolne imiona.
- Następnie proszę za pomocą metody *.append()* dodać do pustej listy trzy dowolne liczby.
- Proszę wyświetlić w terminalu ostatni element listy *list2*
- Odwołując się do określonego indeksu proszę zamienić imię “Anna” w pierwszej liście na imię “Ewa”.
- Proszę usunąć ostatnią liczbę z *list2*
- Proszę stworzyć listę *list3*, która będzie dokładną kopią listy *list2*.
- Proszę wyświetlić wyświetlić drugi i trzeci element listy *list3* w terminalu korzystając z zakresu indeksów.

4) Pętle

W momencie, kiedy chcielibyśmy np. dodać do listy bardzo dużo nowych elementów albo wyświetlić na ekran każdy element listy oddzielnie w tej sytuacji przydatne będą pętle.

Zasadniczo wyróżniamy dwa rodzaje pętli: pętle *for* i pętlę *while*. Pierwsza z nich iteruje po określonym zbiorze danych, druga iteruje dopóki, dopóty podany warunek jest prawdziwy.

Przykładowa pętla wypełniająca danymi pobranymi od użytkownika tablicę *tab*:

```
tab = []
for a in range(1, 11):
  print('Podaj liczbę: ')
  tab.append(input())  # następuje dynamiczne przypisanie # kolejnych wartości
wpisanych z klawiatury przez użytkownika
for a in range(1, 11, 1):
  print(tab[a]) # wypisanie kolejno tablicy od
# miejsca ostatniego do pierwszego
print(tab) # przykład szybkiego wypisania całej tablicy
```

- Proszę zmodyfikować program tak, aby wyświetlić tablicę od końca.

- Za pomocą pętli for można także iterować bezpośrednio po liście (bez użycia metody *range()*). Proszę stworzyć pustą tablicę, którą proszę wypełnić za pomocą pętli *for* markami samochodów pobranymi od użytkownika a następnie za pomocą drugiej pętli *for* wyświetlić w terminalu elementy tablicy iterując bezpośrednio po niej.

Przykład pętli while, który dodaje do każdego elementu istniejącej tablicy (niezależnie od jej długości) liczbę 1:

```
tab = [1,23,43,23,434,53,23] # tablica z elementami
i = 0 # początkowa wartość licznika
while i < len(tab): # pętla wykonuje się dopóki licznik jest mniejszy od długości tablicy
	tab[i] = tab[i] + 1
	i += 1 # zwiększenie licznika o 1
print(tab) # wypisanie tablicy po zwiększeniu każdego elementu o 1
```

Proszę stworzyć pętlę while, która iteracyjne doda do siebie wszystkie elementy w liście o dowolnej długości.

5) Instrukcje warunkowe

W językach programowania istnieje możliwość podejmowania działania po spełnieniu określonego warunku. Elementem języka za to odpowiedzialnym są instrukcje warunkowe (sterujące).
Poniżej przedstawiono przykład działania prostej instrukcji warunkowej:

```
a = float(input('Proszę wpisać wartość : '))
if a == 0: # sprawdzenie czy liczba jest równa 0
	print('Liczba to 0')
elif a % 2 == 0: # sprawdzenie czy liczba jest parzysta
	print('Liczba jest parzysta')
else : # jeśli liczba nie jest równa 0 ani parzysta to jest nieparzysta
	print('Liczba jest nieparzysta')
```

Warunki podane w *if* oraz *elif* mogą być połączone operatorami logicznymi *and* i *or*. Proszę przepisać powyższy przykład z wykorzystaniem operatorów logicznych *and* i *or*.

Instrukcje warunkowe mogą być także zagnieżdżane (tzn. piszemy instrukcję warunkową wewnątrz innej instrukcji warunkowej);

```
a = float(input('Proszę wpisać wartość : '))

if a > 0:
	if a % 2 == 0:
	   print('Liczba jest dodatnia i parzysta')
	else :
	   print('Liczba jest dodatnia i nieparzysta')
elif a < 0:
```

```
    if a % 2 == 0:
       print('Liczba jest ujemna i parzysta')
    else :
       print('Liczba jest ujemna i nieparzysta')
```

- **Proszę samodzielnie stworzyć program pozwalający na obliczanie miejsc zerowych funkcji kwadratowej. W zależności od ilości miejsc zerowych wyświetlana powinna być ich ilość.**

Zadanie domowe:

Napisz program, który będzie działał jako kalkulator kredytowy. Program będzie umożliwiał sprawdzenie, czy określona wartość kredytu będzie dostępna dla danego użytkownika i jaka będzie wysokość miesięcznej raty na podstawie podanych parametrów.

Wprowadzenie danych (2 pkt):

- Poproś użytkownika o podanie kwoty kredytu (w złotych).
- Poproś użytkownika o podanie oprocentowania kredytu (w procentach).
- Poproś użytkownika o podanie liczby lat na jakie zostanie udzielony kredyt.
- Poproś użytkownika o podanie miesięcznych dochodów.

Obliczenie miesięcznej raty (3 pkt):

- Na podstawie wprowadzonych danych oblicz miesięczną ratę kredytu.
- Wykorzystaj wzór do obliczenia wielkości rat stałych kredytu:

```
R=A*((b/m)*(1+b/m)**n)/((1+b/m)**n -1)
```

gdzie: **R** – wysokość raty równej, **A** – kwota udzielonego kredytu, **b** – wysokość oprocentowania kredytu w skali roku, **m** – liczba miesięcy w roku, **n** – liczba rat

Instrukcja warunkowa (2 pkt):

- Dodaj instrukcję warunkową, która sprawdzi, czy miesięczna rata jest niższa niż 1/3 miesięcznych dochodów użytkownika.
- Jeśli rata jest niższa niż 1/3 dochodów, wyświetl komunikat "Kredyt jest dostępny".
- Jeśli rata jest równa lub wyższa niż 1/3 dochodów, wyświetl komunikat "Kredyt nie jest dostępny".

Wyświetlanie wyniku (2 pkt):

- Wyświetl wynik obliczeń w czytelny sposób, informując użytkownika o wysokości miesięcznej raty.

Pętla (1 pkt):

- Po zakończeniu obliczeń zapytaj użytkownika, czy chce dokonać kolejnych obliczeń (tak/nie).
- Jeśli użytkownik wybierze "tak", wróć do punktu 1 i pozwól mu wprowadzić nowe dane.
- Jeśli użytkownik wybierze "nie", zakończ program.

Plik w formacie .zip proszę umieścić w terminie na platformie UPEL.