Instrukcja warunkowa if

Stosując if w Python pamiętaj o kilku najważniejszych zasadach:

- W linii dotyczącej warunku (tej z if) używaj znaku dwukropka ":" na zakończenie
- sam **if w Python** pisany jest zawsze z małej litery
- w kolejnych blokach po instrukcji if używaj wcięcia o długości czterech spacji (chyba, że konsekwentnie używasz innej liczby wcięć)
- operator równoważności w instrukcji if to dwa znaki równości "=="

Operatory, z których możesz skorzystać, to przede wszystkim **operatory logiczne** i **operatory porównania**. Poniżej kilka przykładów:

```
x = 100
if x < 100:
    print("Liczba mniejsza od 100")
elif x == 100:
    print("x jest równe 100")
elif x > 100:
    print("x jest większe od 100")
else:
    print("Nie wiadomo co to")

# przykład 2
x = 10.1
print(x, 'is', większy od 100' if x > 100 else 'nie jest większy od 100')
```

Pętla for i pętla while

Pętlę `for` możemy zrealizować z użyciem funkcji `range`.

Funkcja `range()` może być używana na 3 sposoby.:

Najbardziej podstawowy i najczęściej używany `range(0, end)` utworzy sekwencję od 0 do podanej liczby. Zapis ten możemy skrócić. Jeśli w range podamy tylko jedną wartość np. `range(5)` oznacza, że start i step mają wartości domyślne – $start = 0, step = 1$.

Podając start, możemy zdecydować od jakiego indeksu Python rozpocznie pętlę np. `range(3, 10)` – `range(start, end)` pozwala określić początek i koniec zakresu.

Natomiast `range(start, end, step)` dodaje do tego co jaki krok ma się wykonać pętla.

Utworzona sekwencja nigdy nie zawiera końca zakresu(!).

Oznacza to, że `range(0, 5)` przeczytamy: *jako wygeneruj zakres od 0 do 5, ale na 5 przestań się wykonywać!*.

Sprawdź to:

```
for i in range(0, 5):
    print("wartosc: ", i)
```

Powyższy kod wyświetli: 0, 1, 2, 3, 4 (wyświetli 5 cyfr, od 0, bez ostatniej będącej końcem zakresu tj. cyfry 5). Tak jak wspomniałam, wartość 0 jest domyślna, więc możemy ją pominąć. Jednak, co jeśli chcemy zacząć od innej wartości?

Porównaj ten kod z

```
for i in range(2, 5):
    print("krok: ", i)
```

oraz

```
for i in range(0, 13, 3):
    print("krok: ", i)
```

Załóżmy, że chcemy zapytać 3 użytkowników o imię, a następnie przywitać każdego po imieniu. Możemy zrobić to tak:

```
name = input("Jak masz na imię?")
print("Cześć", name)
name = input("Jak masz na imię?")
print("Cześć", name)
name = input("Jak masz na imię?")
print("Cześć", name)
```

wygodniej byłoby jednak tak:

```
for user in range(0, 3):
    name = input("Jak masz na imię?")
    print("Cześć", name)
```

Pętla `while` wykonuje się dopóki pewien warunek logiczny jest spełniony. Przykład:

```
# Wypisze 0 1 2 3 4

licznik = 0
while licznik < 5:
    print licznik,
    licznik += 1  # Ma to taki sam efekt jak licznik = licznik + 1
```

Instrukcje "break" i "continue"

`break` jest używany do zakończenia pętli `for` i `while`, podczas gdy `continue` pozwala opuścić blok instrukcji niżej i wrócić do nagłówka pętli. Kilka przykładów:

```
# Wypisze 0 1 2 3 4

licznik = 0
while True:
    print(licznik),
    licznik += 1
    if licznik >= 5:
        break
# spróbuj wykonać z poniższą linijką i bez
print(licznik)



# Wypisze tylko liczby nieparzyste - 1 3 5 7 9
for x in range(10):
    # Sprawdz, czy x jest parzyste
    if x % 2 == 0:
        continue
    print(x)
```

Zadania

1. Przetestuj wszystkie przykłady
2. Napisz program, który oblicza sumę 10 kolejnych parzystych liczb całkowitych począwszy od 2 i wyświetla ją na ekranie monitora. Zastanów się nad dwoma możliwymi rozwiązaniami
3. Rozwiąż problem za pomocą programu: Ojciec ma syna, któremu daje przez 10 dni pieniądze w następujący sposób. Pierwszego dnia syn otrzymuje 2 złote, każdego następnego dnia otrzymuje dwa razy więcej niż w dniu poprzednim. Ile pieniędzy zaoszczędzi syn