STEP IT ACADEMY

PROGRAMMING WITH python

Lekcja # 4

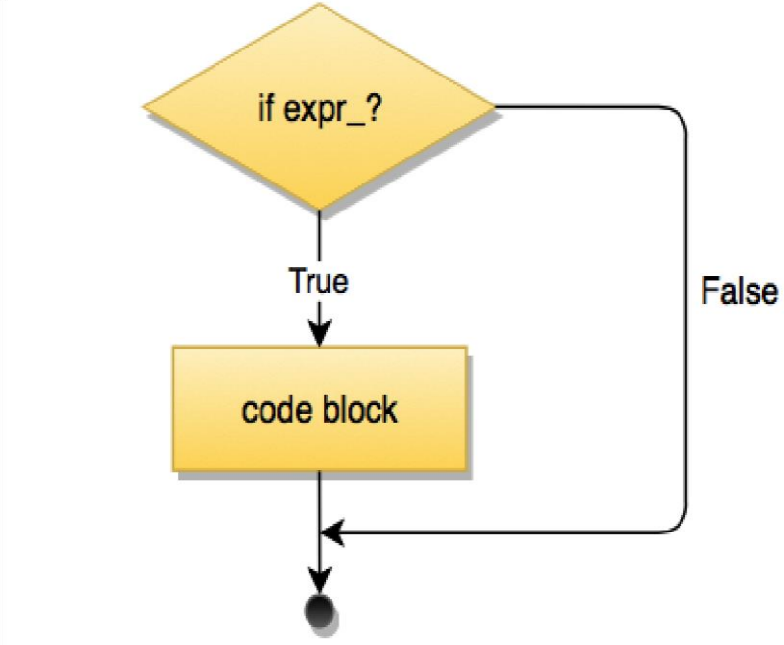
Instrukcja warunkowa if
Operatory porównania

Spis treści

Operator warunkowy

W swoim codziennym życiu spotykasz się z warunkami. Na przykład sytuacja z gry: jeśli gracz zebrał 500 mistycznych kamieni, może otworzyć dodatkową mapę.

Kolejny przykład: twoja postać w grze jest wojownikiem. Jego bronią jest miecz. Taki warunek jest bardzo łatwo do zweryfikowania, a odpowiedzią będzie tylko tak lub nie (Rysunek 1). ***Jesteś wojownikiem? – Tak. – Załóż miecz.***



Rysunek 1

Warunek jest sformułowany w ten sposób: piszemy operator `if` z wyrażeniem w nawiasach które sprawdzamy. Jeśli wyrażenie jest prawdziwe - `true`, wtedy blok kodu za dwukropkiem jest wykonywany, jeśli jest fałszywe - `false`, sekcja kodu jest pomijana.

Operatory porównania

Aby tworzyć warunki **wymagane jest użycie operatorów porównania: operatory dają odpowiedź tak lub nie**, na przykład: ***Czy masz dzisiaj kartkówkę w szkole?*** – ***Tak*** – idziesz do szkoły, – ***Nie*** – Robisz inne rzeczy.

Podstawowe operacje logiczne zostały podane niżej na prostych przykładach z liczbami. Odpowiednio `true` wynik porównania jest prawidłowy, a `false` fałszywy.

Operacja	Nazwa	Przykład	Wynik
>	Większy niż	`print(5>2)`	`true`
<	Mniejszy niż	`print(5<2)`	`false`
>=	Większy niż lub równy	`print(5>=2)`	`true`
<=	Mniejszy niż lub równy	`print(5<=2)`	`false`
==	Równy	`print(5==2)`	`false`
		`print(5==5)`	`true`
!=	Różny od	`print(5!=2)`	`true`

Spośród wszystkich wymienionych operatorów, mogłeś nigdy nie spotkać `!=`. To znaczy **“Różny od”** i zadaje pytanie: ***Czy to prawda że 5 jest różne od 2?*** Odpowiedź brzmi ***Tak, to nie jest równe***, więc rezultatem operacji jest prawda - `true`.

Wyrażenie if – else

Pierwszym wyrażeniem logicznym jest `if` (*jeśli*) i jest stosunkowo prosty do użycia – wynikiem jest `true` lub `false`. Jeśli warunek jest prawdziwy, sekcja kodu z pętli jest wykonywana. Będzie powtarzana dopóki wyrażenie `if` jest prawdą.

Składnia operatora `if` wygląda jak poniżej:

```
if warunek_logiczny:
   komenda_1
   komenda_2
   ...
```

Pojedynczy operator `if` nie zawsze jest wystarczający. Dla takich przypadków powstał blok `else`, który pozwala na wykonanie bloku instrukcji, jeśli warunek operatora `if` nie został spełniony.

Składnia `if-else` wygląda jak poniżej:

```
if warunek_logiczny:
  komenda_1
  komenda_2
  ...
else:
  komenda_3
  ...
```

Napiszmy teraz kod w którym zastosujemy tę składnię i wyświetlimy wynik:

```
# operator if-else
a = 5
b = 6
if (a < b):
 print("a jest mniejsze niż b")
else:
 print("a jest większe niż b")
```

Ten program używa operatora warunkowego `if-else`. Wyrażenie logiczne (`a < b`) jest określane po operatorze `if`. Następnie umieszczony jest **blok (sekwencja)** instrukcji, który zostanie wykonana jeśli warunek jest spełniony.

Następnie jest `else` (*w przeciwnym wypadku*) i blok instrukcji, który zostanie wykonany jeśli warunek (`a < b`) nie jest spełniony.

Napiszemy krótki program, w którym użytkownik może wprowadzić liczbę od 0 do 24 – oznaczającą godzinę, a program wyświetli czy jest dzień, czy noc.

Mówiliśmy wcześniej o pracy z funkcjami wejść/wyjść `input()` i `print()`. Połączmy to z operatorem warunkowym `if-else` w naszym programie.

```
# time
time = input("Wprowadź godzinę: ")
if time < 12:
   print("Jest przedpołudnie!")
else:
   print("Jest popołudnie!")
```

Świetnie! Teraz już wiesz jak używać operatora warunkowego.

Biblioteka math i random

Aby napisać kolejne programy, będziesz musiał skorzystać z biblioteki `math` i `random`. Jak wiesz z poprzednich lekcji **biblioteki to kolekcje funkcji lub zmiennych zebranych w całość**.

W poprzednich zajęciach używaliśmy biblioteki `turtle`, aby rysować. Jak pamiętasz, importowaliśmy ją na samym początku programu.

Pewnie już zauważyłeś, Python posiada ogromną liczbę bibliotek, którą pomagają rozwiązać różnorodne problemy.

Biblioteka math

Zawsze odsyłamy do oficjalnej dokumentacji. **Biblioteka `math` pozwala implementować operacje matematyczne**, jak zaokrąglanie podnoszenie do potęgi, pierwiastkowanie:

```
import math
print(math.ceil(2.5)) # ceil(2.5) ≈ 3
print(math.floor(2.5)) # floor(2.5) ≈ 2
print(math.pow(2, 4)) # pow(2, 4) =
                                2*2*2*2 = 16.0
print(math.sqrt(16)) # sqrt(16) = 4*4 = 4.0
```

W module `math`, funkcja `math.ceil(x)` i `math.floor(x)` są funkcjami zaokrąglania. Pierwsza z nich zaokrągla w górę, druga zaokrągla w dół. Funkcja `math.pow(x, y)` podnosi liczbę x do potęgi liczby y: $x^y=$

$x_1 * x_2 * \ldots * x_y$, a funkcja `math.sqrt(x)` oblicza pierwiastek z x: $\sqrt{x}$.

Biblioteka random

Biblioteka `random` pozwala generować losowe liczby. To tak jakbyś poprosił przyjaciela o pomyślenie o dowolnej liczbie, na przykład od 0 do 10 lub od 100 do 200.

Spójrzmy na przykład dwóch funkcji, które pozwolą wygenerować dwa typy liczb `int` i `float`:

```
import random

print(random.random()) # 0.4956579385740163
print(random.randint(0, 1)) # 0
print(random.randint(1, 100)) # 66
print(random.randint(10, 20)) # 16
```

W bibliotece random nazwana tak samo metoda `random()` generuje liczby zmiennoprzecinkowe (`float`) w przedziale <0.0, 1.0). Aby tego użyć należy podać nazwę biblioteki i jej funkcję `random.random()`. W tym przypadku, `randint()` generuje liczby całkowite (`int`) w przedziale `a <= N <= b`, gdzie `N` jest liczbą losową.

Jak widzisz jest to bardzo przydatna biblioteka. Będziemy jej używać za każdym razem, kiedy chcemy symulować rzut kostką lub monetą i każdej innej sytuacji, gdzie nasz wynik musi być kompletnie losowy.

Operacje matematyczne

Nie zapominajmy o licznych operacjach matematycznych, których także potrzebujemy do rozwiązywania problemów. Operacje te obejmują sumę, różnicę, mnożenie, dzielenie i podnoszenie do potęgi.

Zapamiętaj – w przypadku sumy i różnicy możesz zapisać operacje w różny sposób, a wynik będzie taki sam.

Suma:

```
a = 2
a += 1 # a = a+1
print(a)
```

Różnica:

```
a = 2
a -= 1 # a = a-1
print(a)
```

W rozważanym przykładzie, kombinacja znaków `a += 1` jest równoznaczna równaniu sumy `a = a + 1`. To samo dotyczy różnicy. Spróbuj zamiast += użyć kombinacji:

- mnożenia `a *= 2`;
- dzielenia `a /= 2`, `a //= 2`, `a %= 2`;
- podnoszenia do potęgi `a **= 2`.

Podnoszenie do potęgi, jak już wiesz, również może być zapisane w różny sposób:

```
import math
a = 2
a = math.pow(a, 2) # a = a**2
print(a)
```

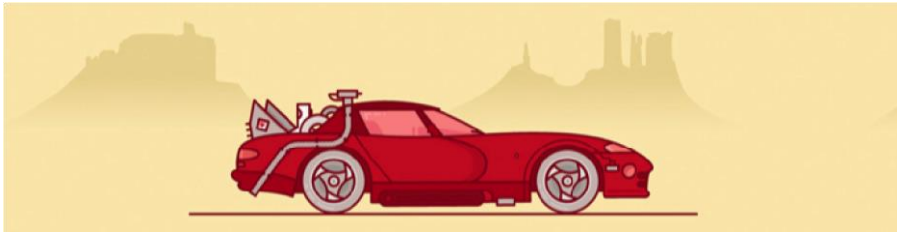
Zobacz że operacja `**` może być zastąpiona funkcją `math.pow(x, y)` z biblioteki `math`.

Rozwiązywanie problemów matematycznych w Python

Zobaczysz jak satysfakcjonujące jest własnoręczne napisanie programu rozwiązującego dany problem.

Konwersja jednostek US na metryczne

W amerykańskich książkach i filmach używane jest określenie ***"mile"*** jeśli chodzi o odległość. Mila jest jedną z jednostek długości. W dodatku jeśli chodzi o prędkość (Rysunek 2), możesz spotkać się z określeniem ***"mile na godzinę"***. Jedna mila jest równa 1.609 km.



Rysunek 2

Napiszmy program, który konwertuje mile na kilometry:

```
# mile na km
n = input("Wprowadź mile: ")
mile = 1.609
km = float(n) * mile
print(f"{n} mile = {km} km")
if km > 80:
  print("Jedziesz za szybko!")
```

Rozwiążmy problem.

Licznik samochodu pokaże prędkość w milach na godzinę. Ile kilometrów na godzinę pokaże licznik, jeśli samochód jedzie z prędkością 43 mil na godzinę?

Aby to zrobić musimy wprowadzić w konsoli liczbę 43 i spojrzeć na wynik.

Odpowiedź: 69.187 km.

Kalkulator zakupów

Wyobraź sobie że jesteś w sklepie, który sprzedaje magiczne lizaki w cenie 270 monet za 1 kg. Sklep ma specjalną ofertę: jeśli kupisz więcej niż 500 g lizaków, ich cena spada do 200 monet za 1 kg (Rysunek 3).

Napisz program, który obliczy cenę zakupów i przekonwertuje gramy na kilogramy; 1 kilogram to 1000 gramów.

Rysunek 3

Ten program rozwiąże nasz problem:

```
# calculate
import math
price_1 = 270
price_2 = 200
gramy = int(input("Wprowadź wagę (gramy): "))
kg = gramy / 1000
print("Twoje zakupy to: {} kg".format(kg))
if (kg<0.5):
 total = kg * price_1
else:
 total = kg * price_2
total = math.ceil(total)
print("Kwota całkowita: {} monet".format(total))
```

W tym programie użyliśmy biblioteki `math`, która pozwoliła zaokrąglić końcową kwotę.

Wyobraź sobie, że wprowadziłeś wartość:

`Wprowadź wagę (gramy): 10001`,

Wynikiem wykonania kodu będzie:

`Kwota całkowita: 2000.1999999999998 monet`.

Możesz to sprawdzić samodzielnie, skomentuj linię `# total = math.ceil(total)` i uruchom program.

Wróćmy do naszego programu. Aby temu zapobiec użyliśmy biblioteki `math` i funkcji `math.ceil(total)`.

Generowanie losowych liczb

Czasami programy wymagają generowania losowych liczb, na przykład jeśli symulujemy kostkę do gry lub rzut monetą. W tych przypadkach, powinniśmy losować liczbę od 1 do 6 (*kostka*) lub 0 i 1 (*rzut monetą*) (Rysunek 4).

Do tego użyjemy poznanej biblioteki `random`.

Rysunek 4

Napiszmy program w którym komputer będzie generował losową liczbę i wyświetlał ją na ekranie. Niech te liczby będą w przedziale od 1 do 6 (*jak kostka do gry*).

Rozważmy sytuację: numer 6 na kostce wygrywa. Ile razy wypadnie 6 jeśli podejmiemy 20 prób?

Ten problem rozwiąże poniższy program:

```
# Losowe liczby
import random
liczba = 0
for i in range(20):
 # random number from 1 to 6
 magiczna_liczba = random.randint(1, 6)
 #print("{}. The magic number is:
             {}" .format(i, magic_number))
if magiczna_liczba == 6:
 liczba = liczba + 1
print("Trafiłeś 6: {} razy".format(number))
```

Zaimportowaliśmy bibliotekę `random`. Stworzyliśmy zmienną liczbową i przypisaliśmy jej wartość 0 w drugiej linii. Następnie użyliśmy pętli `for`, na której pracowaliśmy w poprzedniej lekcji. Pętla pozwala na wykonanie sekcji kodu określoną ilość powtórzeń. funkcja `range(20)` tworzy sekwencję powtórzeń od 0 do 19, oznacza to, że kod po pętli `for`, powtórzy się 20 razy.

Zapamiętaj, Jeśli nie sprecyzujemy sekwencji liczb, program nie zacznie liczyć od 1 jakby się wydawało, lecz od 0.

Następnie stworzyliśmy zmienną magiczna_iczba, która przechowuje losową liczbę. Losowanie odbywa się dzięki funkcji `random.randint(1, 6)` i może to być 1,2,..., 5,6. Możesz usunąć komentarz z linii `#print("{}.The magic number is: {}" .format(count, magic_number))` usuwając znak `#`, aby wyświetlić wygenerowaną liczbę w konsoli.

Dodajemy 1 do zmiennej liczba `liczba = liczba + 1`, gdy wypadnie 6. Zapewni nam to zliczanie wylosowanych szóstek.

Lekcja # 4
Instrukcja warunkowa if
Operatory porównania