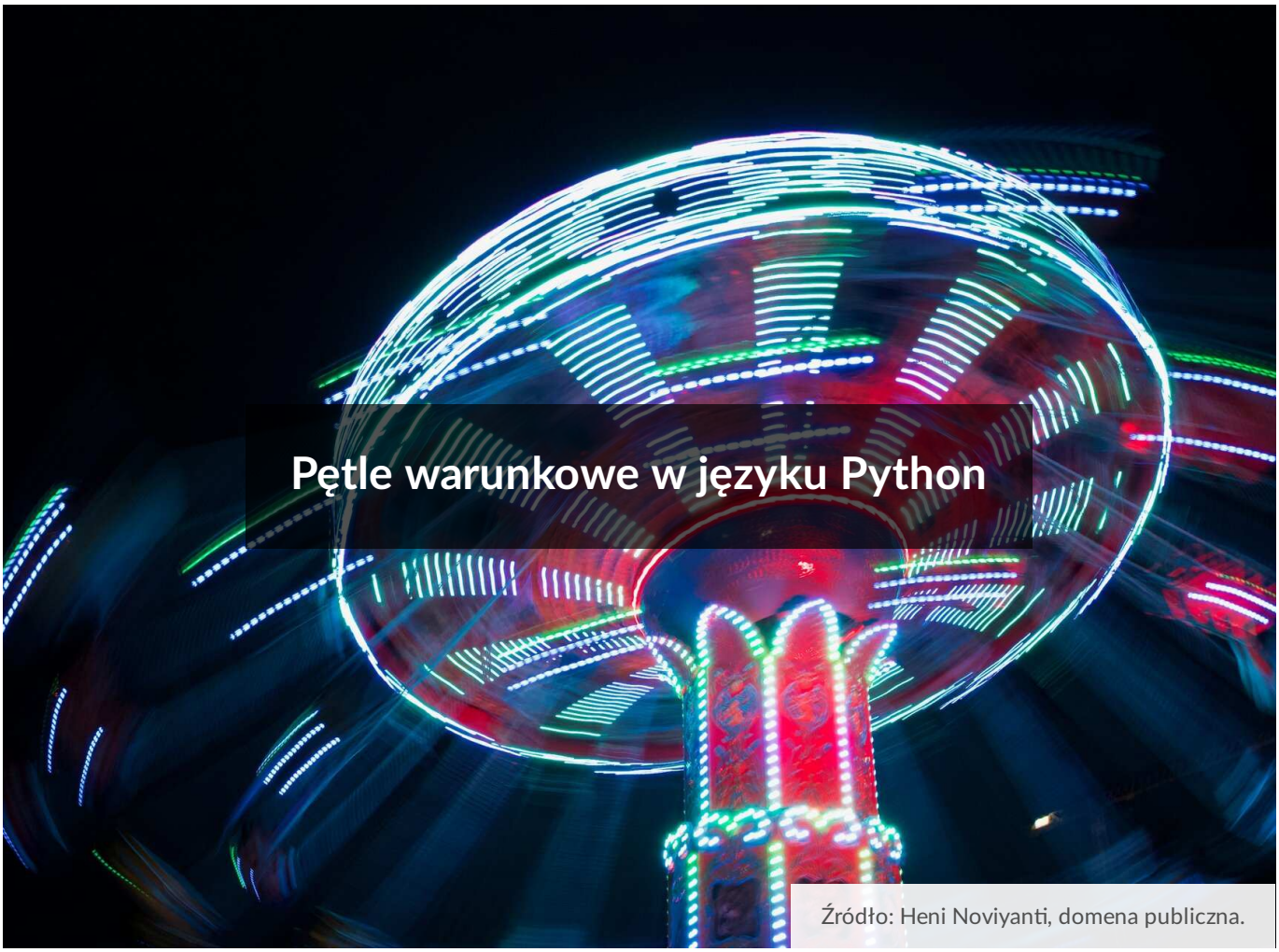
Pętle warunkowe w języku Python

- Wprowadzenie
- Przeczytaj
- Schemat interaktywny
- Sprawdź się
- Dla nauczyciela



Źródło: Heni Noviyanti, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Pętle należą do grupy instrukcji sterujących działaniem programu. Pozwalają one na cykliczne wykonanie instrukcji. Poznaliśmy już pętlę `for` oraz działanie pętli warunkowej `while`.

W tym e-materiale dowiesz się, jak dokonać implementacji pętli `while` w języku Python.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Pętle warunkowe w języku C++,
- Pętle warunkowe w języku Java.

Więcej zadań? Sięgnij do Pętle warunkowe – zadania maturalne.

> **Twoje cele**
>
> - Przeanalizujesz sytuacje, które wymagają zastosowania iteracji/pętli.
> - Zastosujesz pętlę `while` w programie.
> - Zaimplementujesz własny algorytm działania bankomatu, korzystając z pętli `while`.

Przeczytaj

Budowa pętli `while`

Przyjrzyjmy się pętli `while` i jej składni w języku Python.

```
while TEST:
    # instrukcja 1 bloku kodu
    # instrukcja 2 bloku kodu
    # [...]
```

Wyrażenie `TEST` jest wyrażeniem logicznym (jego wartość to prawda lub fałsz) i może być zapisane z wykorzystaniem operatorów logicznych, relacji lub porównania:

```
while zmienna > 12:
    # instrukcja 1
    # instrukcja 2
```

Instrukcje wewnętrzne pętli są wykonywane, dopóki wartość wyrażenia `TEST` to `PRAWDA` (wartość logiczna `True`).

> **Ważne!**
>
> Pamiętaj, aby tak skonstruować wyrażenie `TEST` oraz instrukcje wewnętrzne pętli, by pętla mogła się zakończyć. Pętlę, która nigdy nie zostanie zakończona, nazywamy pętlą nieskończoną:
>
> ```
> while True:
>     print('Witaj!')
> ```
>
> Powyższa pętla będzie w nieskończoność wypisywać na standardowym wyjściu słowo `Witaj!`
> **Przykład 1**
>
> Przygotujmy kod, który pozwoli stworzyć ciąg znaków. Do początkowo pustego ciągu będziemy dodawać znaki podane przez użytkownika tak długo, aż użytkownik wprowadzi znak „!” (wykrzyknik). Na końcu napis zostanie wypisany na ekranie.

Program zakończy swoje działanie, gdy użytkownik wprowadzi wyłącznie znak „!”, poniższe wprowadzone przykładowe łańcuchy tekstowe nie zatrzymają jego działania:

- `!!!`
- `a!`

Zapiszmy program, który wyświetli łańcuch znaków `Python!`

Uwaga, wykrzyknik jest ostatnim znakiem wynikowego tekstu.

```
tekst = ""
litera = ""
while litera != "!":
    litera = input("Podaj literę (znak '!' zakończy): ")
    tekst += litera
print(tekst)

Podaj literę (znak '!' zakończy): P
Podaj literę (znak '!' zakończy): y
Podaj literę (znak '!' zakończy): t
Podaj literę (znak '!' zakończy): h
Podaj literę (znak '!' zakończy): o
Podaj literę (znak '!' zakończy): n
Podaj literę (znak '!' zakończy): !
Python!
```

Żeby program zakończył swoje działanie, użytkownik musi wprowadzić znak „!”.

W niektórych przypadkach język Python pozwala na pominięcie zapisu testu logicznego.

W tym wypadku pętla będzie się wykonywać do czasu, dopóki zmienna, której wartość badamy w warunku pętli, jest różna od `None`, `False`, bądź `0`. Musimy pamiętać o modyfikacji zmiennej lub przerwaniu pętli poleceniem `break`.

Przykład 2

Pętla `while`, która będzie wykonywać się, dopóki zmienna całkowitoliczbowa `zmienna` jest większa od 0.

```
zmienna = 9
while zmienna:
    print("Zmienna=", zmienna)
```

```
    zmienna -= 1

# efekt
# Zmienna= 9
# Zmienna= 8
# Zmienna= 7
# Zmienna= 6
# Zmienna= 5
# Zmienna= 4
# Zmienna= 3
# Zmienna= 2
# Zmienna= 1
```

Są przypadki, w których zaprezentowana pętla będzie nieskończona. Stanie się tak, kiedy początkowa wartość zmiennej `zmienna` będzie liczbą ujemną.

Przykład 3

Pętla `while`, która będzie wykonywać się, dopóki zmienna logiczna jest różna od `False`.

```
warunek = True
liczba = 10
while warunek:
    print(liczba)
    if liczba < 5:
        warunek = False
    liczba = liczba - 1
# efekt
# 10
# 9
# 8
# 7
# 6
# 5
# 4
```

Przykład 4

Pętla `while`, która będzie wykonywać się, dopóki zmienna ma ustawioną wartość (jest różna od `None`).

```
napis = "Ala ma kota"
while napis:
    print(napis)
    napis = None

# efekt
# Ala ma kota
```

Przykład 5

Nieskończona pętla `while`, która zostanie przerwana instrukcją `break`.

```
zmienna = 5
while True:
    print("Zmienna=", zmienna)
    zmienna -= 1
    if zmienna < -3:
        break

# efekt
# Zmienna= 5
# Zmienna= 4
# Zmienna= 3
# Zmienna= 2
# Zmienna= 1
# Zmienna= 0
# Zmienna= -1
# Zmienna= -2
# Zmienna= -3
```

Pętle `while` i operacje na listach

Przygotujmy kod, dzięki któremu uzyskamy listę wartości całkowitych (np. ocen), które później możemy przetworzyć (np. obliczyć średnią arytmetyczną). Załóżmy, że gdy zostanie wprowadzona ocena 0, nasz program powinien zakończyć działanie.

Zmienną `ocena` należy zainicjować wartością różną od 0, aby warunek pętli `while` miał wartość `True` i instrukcje wewnętrzne pętli mogły zostać wykonane. Zainicjowanie zmiennej `ocena` wartością 0 spowoduje pominięcie pętli `while` i wypisanie pustej listy `lista_ocen`.

```
lista_ocen = []
ocena = 1
while ocena:
    ocena = int(input("Podaj liczbę naturalną ('0' zakończy): "))
    lista_ocen.append(ocena)
print(lista_ocen)
```

Przykładowe działanie programu:

```
Podaj liczbę naturalną ('0' zakończy): 3
Podaj liczbę naturalną ('0' zakończy): 5
Podaj liczbę naturalną ('0' zakończy): 6
Podaj liczbę naturalną ('0' zakończy): 3
Podaj liczbę naturalną ('0' zakończy): 7
Podaj liczbę naturalną ('0' zakończy): 2
Podaj liczbę naturalną ('0' zakończy): 1
Podaj liczbę naturalną ('0' zakończy): 3
Podaj liczbę naturalną ('0' zakończy): 0
[3, 5, 6, 3, 7, 2, 1, 3, 0]
```

Możemy zauważyć, że ostatnia wprowadzana wartość 0 dodawana jest do elementów listy. Chcielibyśmy, aby liczba kończąca (0) nie była elementem naszej listy oraz aby możliwe było dodanie tylko wartości większych od zera. Wprowadzimy modyfikację, dzięki której do listy dodamy tylko wartości większe niż 0.

```
lista_ocen = []
ocena = 1
while ocena:
    ocena = int(input("Podaj liczbę naturalną ('0' zakończy): "))
    if ocena > 0:
        lista_ocen.append(ocena)

print(lista_ocen)
```

Przykładowe działanie programu po zmianach:

```
Podaj liczbę naturalną ('0' zakończy): 3
Podaj liczbę naturalną ('0' zakończy): 5
```

```
Podaj liczbę naturalną ('0' zakończy): 6
Podaj liczbę naturalną ('0' zakończy): 3
Podaj liczbę naturalną ('0' zakończy): 7
Podaj liczbę naturalną ('0' zakończy): 2
Podaj liczbę naturalną ('0' zakończy): 1
Podaj liczbę naturalną ('0' zakończy): 3
Podaj liczbę naturalną ('0' zakończy): 0
[3, 5, 6, 3, 7, 2, 1, 3]
```

Pętla `while` może posłużyć do „pobierania" kolejnych elementów z listy. Zdefiniujmy obiekt `lista_t`, zawierający elementy, które są po kolei usuwane z wykorzystaniem funkcji `del()`. Instrukcje w pętli są wykonywane, dopóki obiekt `lista_t` zawiera jakiekolwiek elementy. Przetestuj działanie programu:

```
lista_t = ['Dura', 'lex', 'sed lex' ]
while lista_t:
    element = lista_t[0]
    del(lista_t[0])
    print(element,end=' ')

print(lista_t)
```

Ćwiczenie 1

Napisz program, który w pętli będzie wykonywał następujące instrukcje tak długo, jak długo użytkownik na pytanie o chęć wykonania będzie odpowiadał „T”:

- zada pytanie o to, jaki łańcuch znaków ma wyświetlić,
- zada pytanie o to, ile razy go wyświetlić,
- wyświetli określoną liczbę razy łańcuch znaków.

Specyfikacja problemu:

Dane:

- `odpowiedz` – pobrana od użytkownika decyzja o kontynuowaniu działania programu; łańcuch znaków
- `n` – liczba naturalna; liczba wyświetleń danego znaku
- `znak` – łańcuch znaków; znak do wyświetlenia

Wynik:

Program wyświetla n-krotnie na standardowym wyjściu pobrany łańcuch znaków.

Przykładowe działanie programu:

```
Podaj znak do wyświetlenia: a
Ile znaków wyświetlić: 5
Wyświetlam: aaaaa
Czy wykonujemy ponownie (T/N)> T
Podaj znak do wyświetlenia: 4
Ile znaków wyświetlić: 4
Wyświetlam: 4444
Czy wykonujemy ponownie (T/N)>
```

Dla zainteresowanych

W wersji 3.8 języka Python wprowadzono zmiany, m.in. możliwości przypisywania wartości zmiennym wewnątrz wyrażeń (opisane są one dokładnie w dokumencie PEP 572) oto przykład takiego zapisu:

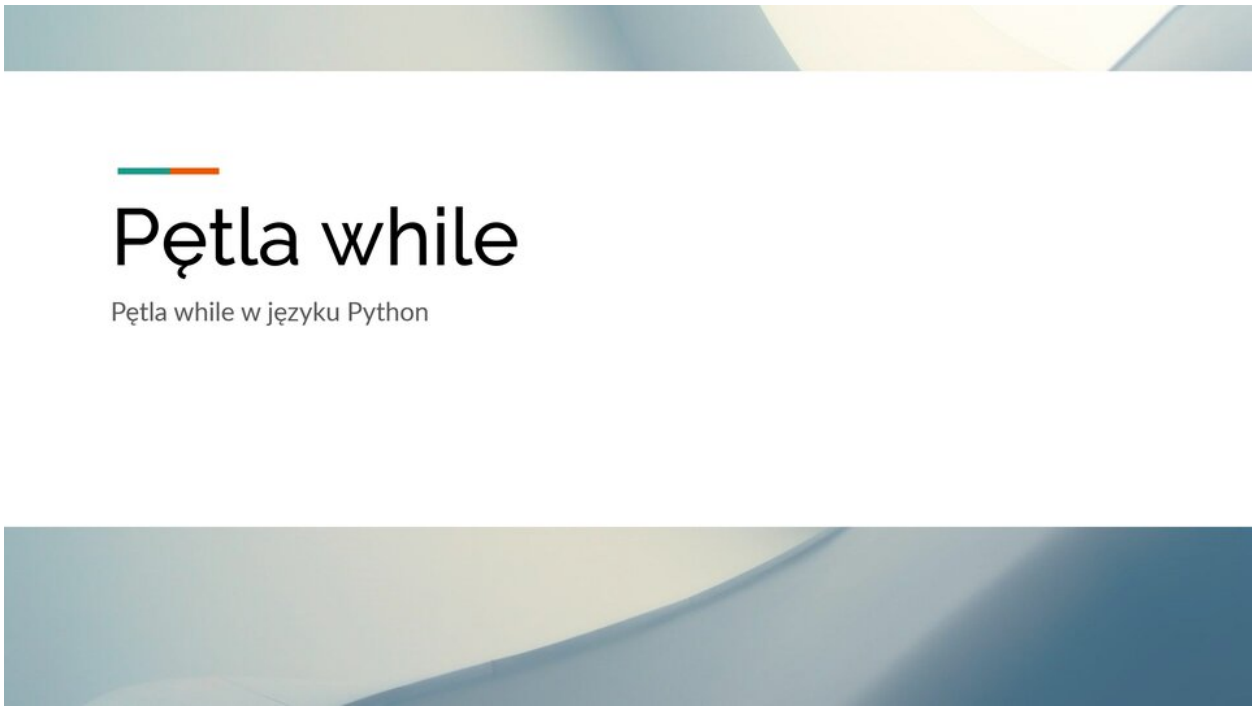
```
# uwaga! ten kod działa tylko w Python >= 3.8
while (polecenie := input("> ")) != "quit":
    print("Wprowadzono:", polecenie)

# przykład wykonania
>>> while (polecenie := input("> ")) != "quit":
    print("Wprowadzono:", polecenie)


> test
Wprowadzono: test
> inny
Wprowadzono: inny
> quit
```

Polecenie 1

Zapoznaj się z filmem przedstawiającym grę w zgadywanie, czy ustawioną wartością jest orzeł czy reszka. Czy potrafisz wskazać inne gry, w którym można wykorzystać pętle warunkowe?



Film dostępny pod adresem /preview/resource/Rwh5e8Bcj9aWE

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Lekcja poświęcona pętlom warunkowym w języku Python.

Instrukcja `else` w pętli `while`

Opcjonalnym elementem pętli `while` jest blok `else`. Wykonywany jest zawsze, jeśli warunek nie jest spełniony lub przestanie być spełniony.

```
lista_t = ['Dura', 'lex', 'sed lex']
while lista_t:
    element = lista_t[0]
    del(lista_t[0])
    print(element, end=' ')
else:
    print('| koniec.')
    print('Tu pusta lista', lista_t)
```

Słownik

> **del()**
>
> funkcja kasująca nieodwracalnie obiekt, usuwająca go z pamięci operacyjnej

Schemat interaktywny

Polecenie 1

Bankomat wypłaca gotówkę w nominałach 500, 200 oraz 50 złotych. Przygotuj algorytm działania bankomatu, który po wprowadzeniu kwoty wydaje pieniądze, używając jak najmniejszej liczby banknotów. Wykorzystaj do tego schemat interaktywny lub język programowania. Wykorzystaj algorytm zachłanny.

Zakładamy, że w bankomacie znajduje się nieograniczona liczba banknotów o nominałach 500, 200 oraz 50 złotych.

Specyfikacja problemu:

Dane:

- *kwota* – kwota do wypłacenia przez bankomat

Wynik:

Program wyświetla liczbę banknotów o nominałach 500, 200 oraz 50, które wykorzystano do wypłacenia podanej kwoty lub komunikat
`Podana kwota nie może zostać wypłacona.`

```
# Tutaj dodaj własny kod. Użyj funkcji
# print()
```

```
```

Ćwiczenie 1

Polecenie 2

Omów zastosowany przez ciebie algorytm wypłacania środków z bankomatu. W jaki sposób można go poprawić, by wydawał banknoty również w pozostałych nominałach?

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Zapoznaj się z kodem i wykonaj ćwiczenie.

```
elementy = [ (x+1)/2 for x in range(23) ]
while elementy:
    if len(elementy) > 1:
        print(elementy[1])
    del(elementy[1])
else:
    print(elementy[0])
```

Zaznacz wszystkie poprawne odpowiedzi.

- [ ] Zostaną wypisane 22 elementy i wystąpi błąd `IndexError: list assignment index out of range`.
- [ ] W pewnym momencie wystąpi błąd `IndexError: list assignment index out of range`.
- [ ] Zostaną wypisane wszystkie 23 elementy.
- [ ] Od razu wystąpi błąd `IndexError: list assignment index out of range`.

Ćwiczenie 2

Napisz program, który za pomocą pętli oblicza sumę liczb nieparzystych z przedziału <m, n>. Przetestuj działanie programu dla przedziału <1, 20>.

Specyfikacja problemu:

Dane:

- n – liczba całkowita
- m – liczba całkowita

Wynik:

Program wypisuje sumę liczb nieparzystych z przedziału <m, n>.

Twoje zadania

1. Program wypisuje sumę liczb nieparzystych z przedziału <m, n>.

```
m = 1
n = 20

    // Tutaj wpisz kod

print(n)
```

```
1
```

Ćwiczenie 3

Maciek jest wielkim fanem liczb naturalnych podzielnych przez z. Chciałby wyznaczyć maksymalnie x kolejnych liczb naturalnych, które są mniejsze od n lub równe n, podzielne przez z, a ich ostatnia cyfra to y.

Zakładamy, że 0 również jest liczbą naturalną.

Napisz program, który pomoże Maćkowi rozwiązać jego problem.

Swoje rozwiązanie przetestuj dla `n = 754`, `x = 10`, `z = 7` oraz `y = 1`.

Specyfikacja problemu:

Dane:

- n – liczba naturalna
- x – liczba naturalna
- z – liczba naturalna
- y – liczba naturalna

Wynik:

Na standardowym wyjściu program drukuje maksymalnie x kolejnych liczb mniejszych od n lub równych n podzielnych przez z, których ostatnia cyfra to y.

Przykładowe wyjście:

```
21
91
161
231
301
```

```
371
441
511
581
651
```

Twoje zadania

1. Na standardowym wyjściu program drukuje x kolejnych liczb mniejszych od n lub równych n podzielnych przez z, których ostatnia cyfra to y.

```
n = 754
x = 10
ile = 0
z = 7
y = 1
liczba = 0

# tu zapisz swój program
# do drukowania wartości użyj funkcji print()
```

```
1
```

Ćwiczenie 4

Napisz program, który wypisze dany łańcuch znaków `tekst` x razy. Przetestuj działanie programu dla łańcucha znaków `panta rhei`, który ma zostać wypisany 19 razy.

Specyfikacja problemu:

Dane:

- `tekst` – łańcuch znaków do wypisania
- `x` – liczba naturalna

Wynik:

Program wypisuje w nowych liniach `x` razy łańcuch znaków `tekst`.

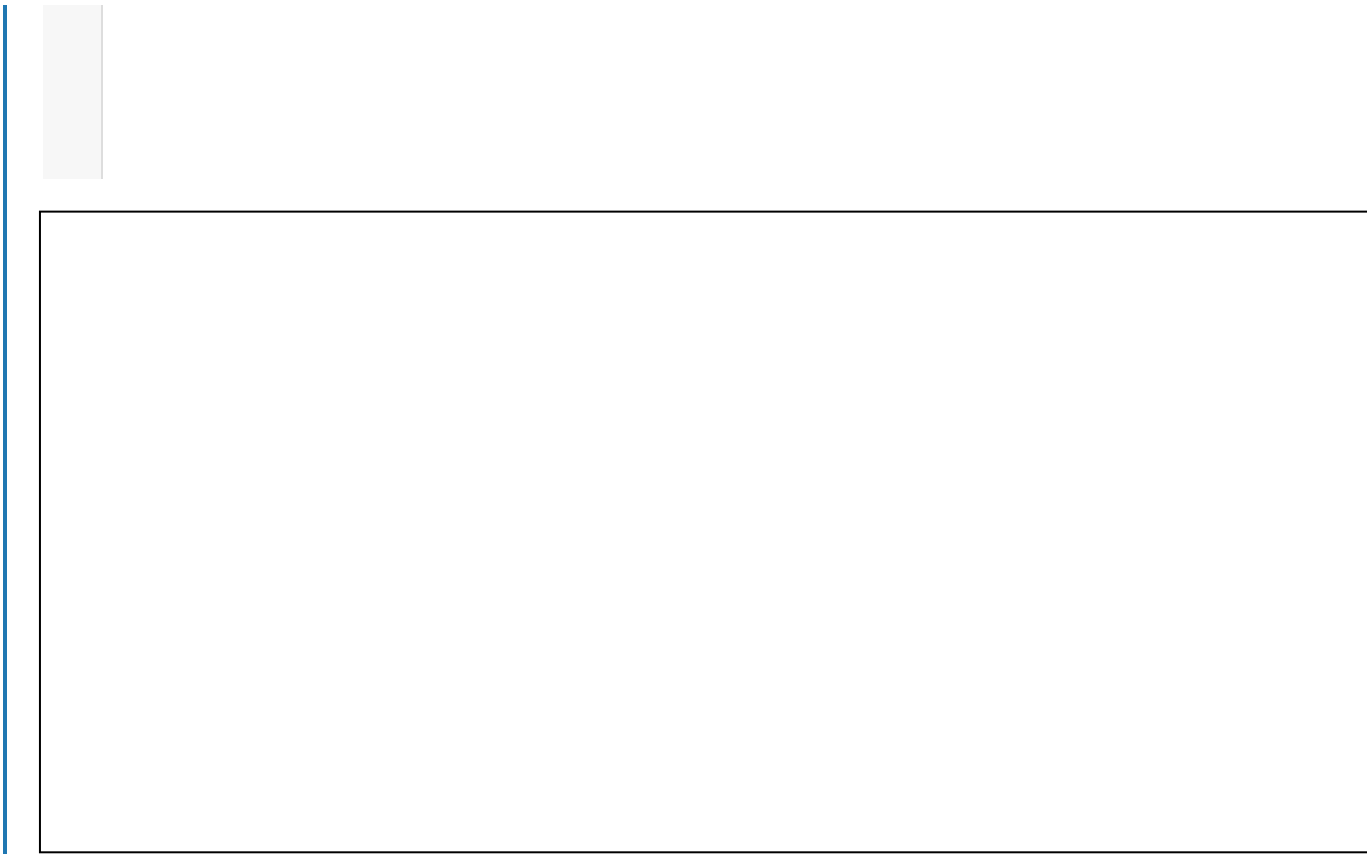
Twoje zadania

1. Program ma wypisać `x` razy napis zawarty w zmiennej `tekst`.

```
tekst = "panta rhei"
i = 0
x = 19

#tu zapisz swój program
```

```
```



Ćwiczenie 5

Napisz program, który wypisze wszystkie liczby całkowite z przedziału <a, b>. Przetestuj działanie programu dla przedziału <5, 27>.

Specyfikacja problemu:

Dane:

- a – liczba całkowita
- b – liczba całkowita

Wynik:

Program wypisuje wszystkie liczby całkowite z przedziału <a, b>.

Twoje zadania

1. Program ma wypisywać kolejne liczby z przedziału <a, b>. Użyj pętli `while`. Liczby wypisuj w nowych liniach.

```
a = 5
b = 27

# tu zapisz swój program
```

```
1
```

Ćwiczenie 6

Napisz program, który wyświetli następujące wyjście:

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

Specyfikacja problemu:

Wynik:

Na standardowym wyjściu program drukuje zaprezentowany w poleceniu wzór.

Wskazówka:

Jeżeli nie chcemy, aby funkcja `print` dodawała do napisu symbol końca linii, wywołujemy ją z parametrem `end`. Przykład: `print(x, end='')`

Twoje zadania

1. Program ma wyświetlić wzór zgodny z poleceniem.

```
liczba = 1
# tu zapisz swój program
```

```
1
```

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Pętle warunkowe w języku Python

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> I. Rozumienie, analizowanie i rozwiązywanie problemów.
>
> Zakres podstawowy. Uczeń:
>
> 2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:
>
> d) wydawania reszty najmniejszą liczbą nominałów,
>
> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> Zakres podstawowy. Uczeń:
>
> 1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;

- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz sytuacje, które wymagają zastosowania iteracji/pętli.
- Zastosujesz pętlę `while` w programie.
- Zaimplementujesz własny algorytm działania bankomatu, korzystając z pętli `while`.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. Uczniowie przypominają sobie najważniejsze informacje związane z pętlami warunkowymi.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Pętle warunkowe w języku Python". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj" w kontekście programowania.

Faza wstępna:

1. **Rozpoznanie wiedzy uczniów.** Chętna lub wybrana osoba referuje najważniejsze informacje dotyczące pętli warunkowych.
2. Przedstawienie tematu i celów zajęć.

Faza realizacyjna:

1. Nauczyciel sprawdza przygotowanie uczniów do lekcji. W razie potrzeby osoby, które zapoznały się z e-materiałem przed lekcją, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się", kiedy pozostałe osoby zapoznają się z sekcją „Przeczytaj".
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Schemat interaktywny". Uczniowie wykonują Polecenie 1. Nauczyciel w razie potrzeby analizuje z nimi algorytm zaprezentowany za pomocą schematu.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 2–6, a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.

Faza podsumowująca:

1. Na koniec zajęć z programowania w Pythonie nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie piszą program, który będzie symulował działanie bankomatu, wykonując w pętli następujące działania:
    - zada pytanie: „Czy wykonać wydawanie?" dla odpowiedzi „T" wykona algorytm, przy odpowiedzi „N" zakończy działanie;
    - zada użytkownikowi pytanie o kwotę do wydania;
    - za pomocą funkcji obliczy liczbę banknotów o nominałach: 500, 200, 100, 50, 20, 10 niezbędną do wydania;
    - wypisze na ekranie informacje o liczbę odpowiednich banknotów.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj", „Schemat interaktywny", „Sprawdź się" jako materiał do lekcji powtórkowej.