Pytanie 1: Jaki będzie wynik uruchomienia następującego programu? DLACZEGO?

```
lancuch_lewy = 'Prawy "
lancuch_prawy = "Prawy"

print(lancuch_lewy > lancuch_prawy)
```

Odp.: Nie wykona się. Błąd składni, w lancuch_lewy literał otwierany jest innym znakiem niż znak zamykający.

Pytanie 2: Jaki będzie wynik uruchomienia następującego programu? DLACZEGO?

```
lista_lewa = [0, 1, 2, [0, 1, 2]]
lista_prawa = [0, 1, 2, {0, 1, 2}]

print(lista_lewa < lista_prawa)
```

Odp.: Nie wykona się. Błąd typów. Porównywane są listy, ale elementem listy lewej pod indeksem 3 jest lista, a elementem listy prawej pod tym samym indeksem jest zbiór (wartości pod mniejszymi indeksami są sobie równe w obu listach). W ostateczności, nie można porównać obiektów różnych typów, które stanowią elementy obu list.

Pytanie 3: Co należy wpisać w zaznaczone miejsce aby otrzymać pokazaną odpowiedź w sprawie relacji długości łańcuchów? DLACZEGO?

```
lancuch_lewy = "Prawy"
lancuch_prawy = "Prawy"

print(.....................................)
```

Odpowiedź: True

Odp.: len(lancuch_lewy) == len(lancuch_prawy)

Odpowiedź dotyczy relacji długości łańcuchów. Funkcja len() zwróci długość łańcucha. Oba łańcuchy mają jednakową długość. Operator relacji == jest operatorem równości.

Pytanie 4: W jednej linii kodu języka Python utworzyć i zainicjalizować listę wartościami nieparzystymi w zakresie od 0 do 1000.

Odp.:
```
lista = [i for i in range(1000) if i % 2 != 0]
lista = [i for i in range(1000) if i % 2 == 1]
lista = [x for x in range(1000) if x % 2]
lista = [i for i in range(1, 1000, 2)]
lista = list(range(1,1000,2))
```

Pytanie 5: W wyniku wykonania komendy print(zmienna) na ekranie pojawił się następujący komunikat:

```
{'pierwszy': 1, 'drugi': 2}
```

Zainicjalizować zmienną o nazwie zmienna
a. w jednej linii z użyciem odpowiedniego konstruktora oraz

b. w jednej linii bez użycia konstruktora
tak, aby za każdym razem w wyniku otrzymać pokazany wyżej komunikat.

```
Odp.    a. zmienna = dict(pierwszy=1, drugi=2) lub
        a. zmienna = dict([('pierwszy', 1), ('drugi', 2)])
        b. zmienna = {"pierwszy" : 1, "drugi" : 2}
```

Pytanie 6: Czy możliwy jest taki wynik uruchomienia poniższego programu? DLACZEGO?

```
zmienna = []

for i in range(10):
    zmienna.append(i)

print(zmienna)
```

Wynik: [0, 1, 2, 3, 4, 6, 5, 7, 8, 9]

Odp.: Nie jest. Lista jest kolekcją uporządkowaną. Jeżeli jej zawartość nie jest modyfikowana po utworzeniu i inicjalizacji, to za każdym razem przy odwołaniu się do jej zawartości, wartości przedstawiane są w tej samej kolejności wynikającej z jej utworzenia.

Pytanie 7: Co oznaczają pojęcia typów mutable (zmiennych) i immutable (niezmiennych) w języku programowania Python?

Odp.: Wartości zmiennych typu mutable mogą być zmieniane w dowolnym momencie po ich utworzeniu. Wartości typu immutable są ustawiane przy inicjalizacji, a potem nie można już ich zmieniać.

Pytanie 8: W jednej linii kodu języka Python utworzyć i zainicjalizować listę tysiącem wartości 555.

```
Odp.:   lista = [555]*1000
        lista = 1000*[555]
        lista = [555 for i in range(1000)]
        lista = [555 for _ in range(1000)]
```

Pytanie 9: Jaki będzie wynik uruchomienia następującego programu? DLACZEGO?

```
lista_lewa = [0, 1, 2, {0, 1, 2}]
lista_prawa = [0, 1, 2, {0, 1, 1}]

print(lista_lewa < lista_prawa)
```

Odp. Listy są porównywane leksykograficznie poprzez porównanie odpowiadających sobie elementów. Oznacza to, że aby dwie sekwencje były uznane za równe, każdy z elementów jednej musi być równy odpowiadającemu mu elementowi w drugiej, ponadto zaś sekwencje muszą być tego samego typu i tej samej długości. lista_prawa jest inicjalizowana zbiorem pod indeksem 3. Jednak zbiór nie może przechowywać nieunikalnych wartości. W rzeczywistości w lista_prawa pod indeksem 3 będzie przechowywany zbiór {0, 1}, który jest krótszy niż zbiór pod indeksem 3 w lista_lewa.
Wynik porównania: False.

Pytanie 10. Jaki będzie wynik uruchomienia następującego programu? DLACZEGO?

```
lista_lewa = [0, 1, 2, "mama"]
lista_prawa = ["mama", 0, 1, 2]

print(lista_lewa == lista_prawa)
```

Odp. Listy są porównywane leksykograficznie poprzez porównanie odpowiadających sobie elementów. Oznacza to, że aby dwie sekwencje były uznane za równe, każdy z elementów jednej musi być równy odpowiadającemu mu elementowi w drugiej.
Wynik porównania: False.

Pytanie 11. Co należy wpisać w kodzie porgramu Python 3.x w zaznaczone miejsce aby otrzymać pokazaną odpowiedź w sprawie relacji długości list? DLACZEGO?

```
lista_lewa = [0, 1, 2, 3]
lista_prawa = [0, 1, 2]

print.....................................
```

Odpowiedź: True

Odp.: Odpowiedź dotyczy relacji długości. Funkcja print w Python 3.x wymaga podania argumentu w nawiasie, funkcja len() zwróci długość listy, operator porównania ma dać odpowiedź czy długość list jest taka sama, lub czy długość listy jest większa.

```
(len(lista_lewa) != len(lista_prawa))

(len(lista_lewa) > len(lista_prawa))
```

Pytanie 12: Jaki będzie wynik wyrażenia a/b - a//b, gdy:

```
a = 0o227 % 9
b = 0x2fd % 7
```

DLACZEGO?

Odp.: Zapis liczby w systemie ósemkowym - liczba rozpoczyna się od 0o.
Zapis liczby w systemie szesnastkowym - liczba rozpoczyna się od 0x.

% - operator modulo (reszta z dzielenia)
/ - operator dzielenia rzeczywistego (wynik typu float)
// - operator dzielenia całkowitego (wynik typu int)

Kolejność wykonywania działań - operatory dzielenia będą wykonane przed operatorem odejmowania.

Po przeliczeniu a = 7.
Po przeliczeniu b = 2.

7/2 - 7//2 = 3.5 - 3 = 3.5 - (niejawne rzutowanie typu na float)3 = 0.5

Pytanie 13: Czy możliwy jest następujący wynik uruchomienia poniższego programu? DLACZEGO?

```
zmienna = []

for i in range(10):
    zmienna.append(i)

zmienna = set(zmienna)

print(zmienna)
```

Wynik: {0, 1, 2, 3, 4, 6, 5, 7, 8, 9}

Odp.: Jest. Zbiór (powstaje po wywołaniu konstruktora zbioru z typem iterowanym - listą) jest kolekcją nieuporządkowaną. Wartości zbioru przedstawiane są w dowolnej kolejności.

Pytanie 14: Uzupełnić w poniższym programie, w pustym miejscu kod, tak aby utworzyć działający zgodnie z oczekiwaniami program (sprawdzający tylko pojedynczy wyraz). Sprawdzany może być dowolny wyraz.

```
1. a = 'kajak'
2. if
3.    print('palindrom')
4. else:
5.    print('nie palindrom')
```

Odp.: a == a[::-1]:

Pytanie 15: Jaki będzie typ wyniku wyrażenia a/b - a//b, gdy:

```
a = 0o227 % 9
b = 0x2fd % 7
```

DLACZEGO?

Odp.: Zapis liczby w systemie ósemkowym - liczba rozpoczyna się od 0o.
Zapis liczby w systemie szesnastkowym - liczba rozpoczyna się od 0x.

% - operator modulo (reszta z dzielenia)
/ - operator dzielenia rzeczywistego (wynik typu float)
// - operator dzielenia całkowitego (wynik typu int)

Kolejność wykonywania działań - operatory dzielenia będą wykonane przed operatorem odejmowania.

Po przeliczeniu a = 7.
Po przeliczeniu b = 2.

7/2 - 7//2 = 3.5 - 3 = 3.5 - (niejawne rzutowanie typu na float)3 = 0.5

Typ float, bo odjemna jest typu float, a odjemnik typu int, ale przed wykonaniem operacji na zmiennych różnych typów nastąpi niejawna promocja z typu młodszego do starszego.

Pytanie 16: Jaki będzie wynik uruchomienia następującego programu? DLACZEGO?

```
zmienna = [0, 1, 2, 2, 1, 0]

inna_zmienna = set(zmienna)

print(inna_zmienna)
```

Odp.: {0, 1, 2} lub {0, 2, 1} lub itd., ponieważ zbiór nie może zawierać duplikatów, ale kolejność elementów może być dowolna.

Pytanie 17: Co należy zmienić w programie, aby było możliwe zachowanie idei przechowania bez błędu pokazanych wartości w zmiennej? DLACZEGO?

```
zmienna = {0, 1, {0}}
```

Odp.: Ideą jest przechowanie w zbiorze wartości przechowywanej w innym zbiorze. Jednak wartości przechowywane w zbiorze muszą być typu niezmiennego, a zbiór jest typem zmiennym. Aby przechować element zbioru, który ma być zbiorem należy użyć zbioru zamrożonego (frozenset), który musi być zinicjalizowany typem iteracyjnym.

```
zmienna = {0, 1, frozenset([0])} lub zmienna = {0, 1, frozenset({0})} lub
zmienna = {0, 1, frozenset((0,))}
```

Pytanie 18: Jaki będzie wynik uruchomienia następującego programu? DLACZEGO?

```
lancuch_lewy = "Prawy "
lancuch_prawy = "Prawy"

print(lancuch_lewy > lancuch_prawy)
```

Odp.: Napisy są porównywane leksykograficznie, przy użyciu liczbowych odpowiedników (wyników działania wbudowanej funkcji ord()) tworzących je znaków. Jeśli w którejś z sekwencji nie występuje element odpowiadający elementowi z drugiej, za mniejszą uznawana jest krótsza sekwencja.
Wynik porównania: True.

Pytanie 19: W poniższym programie uzupełnić linie kodu numer 2 i 3

```
1. a = input('Podaj liczbę całkowitą\n')
2.
3.
4. print(a)
```

tak aby uzyskać odpowiedź programu równą:
2
gdy użytkownik programu, w kolejnych uruchomieniach, wpisze z klawiatury wartość

a) 2
b) 2.1

Odp.: 2. a=float(a)
       3. a=int(a)

Wpisanie a = int(a) w linii 2 będzie zgłaszać błąd. Najpierw konwersja na liczbę typu float, a dopiero później do int.

Program 1: Napisz uniwersalny (działający z dowolnym tekstem) program zliczający liczbę słów, wczytanych z pliku tekstowego, zaczynających się od wielkiej litery A lub od małej litery a. Obecnie plik tekstowy zawiera tekst:

Ala ma kota, a kot ma Alę

W kodzie programu nie może wystąpić znak A (wielka litera A). Program powinien wypisywać liczbę zliczonych słów spełniających założenia. Kod programu zmieścić w 8 liniach (efektywnych, bez pustych linii).

Przykład 1 odp.:

```
f = open('slowa.txt', 'r', encoding='utf-8')
text = f.read()
text = text.lower().split()
counter = 0
for word in text:
    if word[0] == 'a':
        counter += 1
print(counter)
```

Przykład 2 odp.:

```
with open('slowa.txt', 'r', encoding='utf-8') as f:
    counter = 0
    for line in f.readlines():
        line = line.lower().split()
        for word in line:
            if word[0] == 'a':
                counter += 1
    print(counter)
```

Program 2: Wysokość wygranych w grze i prawdopodobieństwa ich osiągnięcia są podane w tabeli:

Wygrane i ich prawdopodobieństwo					
Wygrane	-1	4	9	14	19
Prawdopodobieństwo	0.977	0.008	0.008	0.006	0.001

Napisać program, który będzie informował o wartości oczekiwanej, wariancji i odchyleniu standardowym w tej grze. W programie, ograniczyć liczbę literałów do niezbędnego minimum.

Przykład 1 odp.:

```
import math
```

```
wygrane = [-1, 4, 9, 14, 19]
prawdopodobienstwo = [0.977, 0.008, 0.008, 0.006, 0.001]

wartosc_oczekiwana = 0
for i in range(len(wygrane)):
    wartosc_oczekiwana += wygrane[i] * prawdopodobienstwo[i]

print(wartosc_oczekiwana)

wariancja = 0
for i in range(len(wygrane)):
    wariancja += (wygrane[i] - wartosc_oczekiwana)**2 *
prawdopodobienstwo[i]

print(wariancja)

print(math.sqrt(wariancja))
```

Przykład 2 odp.:

```
wygrane = [-1, 4, 9, 14, 19]
prawdopodobienstwo = [0.977, 0.008, 0.008, 0.006, 0.001]
wartosc_oczekiwana = sum([wygrane[i] * prawdopodobienstwo[i] for i in
range(len(wygrane))])
print(f"Wartość oczekiwana = {wartosc_oczekiwana}")
wariancja = sum([(wygrane[i] - wartosc_oczekiwana)**2 *
prawdopodobienstwo[i] for i in range(len(wygrane))])
print(f"Wariancja = {wariancja}")
print(f"Odchylenie standardowe = {wariancja**0.5}")
```

Przykład 3 odp.:

```
gra = {-1 : 0.977, 4 : 0.008, 9 : 0.008, 14 : 0.006, 19 : 0.001}
wartosc_oczekiwana = sum([key * value for key, value in gra.items()])
print(f"Wartość oczekiwana = {wartosc_oczekiwana}")
wariancja = sum([(key - wartosc_oczekiwana)**2 * value for key, value in
gra.items()])
print(f"Wariancja = {wariancja}")
print(f"Odchylenie standardowe = {wariancja**0.5}")
```

Program 3: Napisać program uniwersalny, który będzie wyznaczał szansę trafienia głównej wygranej w grze liczbowej polegającej na skreśleniu k (2 <= k <= 6) liczb z n (10 <= n <= 49). Wartości k i n podaje użytkownik programu. Zastosować ochronę przed podawaniem błędnych wartości k i n. Wyświetlić wynik jako:

1 do wyznaczona_wartość

Nie stosować rekurencji. Zastosować przynajmniej jedną funkcję.

Przykład 1 odp.:

```
def czytaj_dane():
    try:
```

```
        n = int(input('Podaj n:\n'))
        if n < 50 and n > 9:
            print(f'Podałeś prawidłową wartość n: {n}')
        else:
            raise Exception('Zła wartość n')
    except:
        print('Podałeś złą wartość n')
        exit()

    try:
        k = int(input('Podaj k:\n'))
        if k < 7 and k > 1:
            print(f'Podałeś prawidłową wartość k: {n}')
        else:
            raise Exception('Zła wartość k')
    except:
        print('Podałeś złą wartość k')
        exit()

    return n, k

def licz(n, k):
    licznik = 1
    mianownik = 1
    for i in range(1, n + 1):
        licznik *= i
    for i in range(1, k + 1):
        mianownik *= i
    for i in range(1, n - k + 1):
        mianownik *= i

    return licznik / mianownik

n, k = czytaj_dane()
wynik = licz(n, k)

print(f'Szanse jak 1 do {wynik}')
```

Przykład 2 odp.:

```
def czytaj_dane():
    N = {'dół' : 9, 'góra' : 50}
    K = {'dół' : 2, 'góra' : 6}

    dobra_wartosc = False
    while not dobra_wartosc:
        try:
            n = float(input(f"Podaj n w zakresie od {N['dół']} do
{N['góra']}:\n"))
            n = int(n)
            if n > 50 or n < 9:
                print(f"Wartość n nie mieści się w założonym zakresie od
{N['dół']} do {N['góra']}:\n")
            else:
                print(f"Mamy n = {n}")
```

```
                dobra_wartosc = True
        except:
            print(f"n musi być liczbą całkowitą z zakresu od {N['dół']} do
{N['góra']}:\n")

    dobra_wartosc = False
    while not dobra_wartosc:
        try:
            k = float(input(f"Podaj k w zakresie od {K['dół']} do
{K['góra']}:\n"))
            k = int(k)
            if k > 6 or k < 2:
                print(f"Wartość k nie mieści się w założonym zakresie od
{K['dół']} do {K['góra']}:\n")
            else:
                print(f"Mamy k = {k}")
                dobra_wartosc = True
        except:
            print(f"k musi być liczbą całkowitą z zakresu od {K['dół']} do
{K['góra']}:\n")

    return n, k


def Newton(n, k):
    wynik = 1
    for i in range(1, k + 1):
        wynik *= (n - i + 1) / i
    return int(wynik)

n, k = czytaj_dane()

print(f'Szanse jak 1 do {Newton(n, k)}')
```

Uwagi na marginesie:

1. Dosyć ciekawe podejście (bardzo powszechne). Odpowiedź na jedno z pytań „…bo zbiór nie przechowuje duplikatów…”. Odpowiedź na inne pytanie „…należy usunąć nawiasy następująco: zmienna = {0, 1, 0}”
2. Skoro coś się da policzyć, to znaczy, że jest policzalne. W języku polskim jak coś jest policzalne to występuje w pewnej (do określenia) liczbie a nie ilości.
3. Pytanie o wartość wyniku, a pytanie o typ wyniku, to dwa różne pytania wymagające dwóch różnych odpowiedzi. Odpowiedzi nie na temat były oceniane na 0 punktów.
4. Trudność: biernik l.mn. od rzeczownika cudzysłów. cudzysłów - Odmiana przez przypadki rzeczownika cudzysłów • Odmiana.NET
5. Diabeł tkwi w szczegółach.