Zad.1. Wprowadzenie.

```
from datetime import datetime

odds = [ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19,
        21, 23, 25, 27, 29, 31, 33, 35, 37, 39,
        41, 43, 45, 47, 49, 51, 53, 55, 57, 59]

this_minute = datetime.today().minute

if this_minute in odds:
    print("Minuta nieparzysta")
else:
    print("Minuta parzysta")
```

Zad.2. Wyrażenie *from - import - as*.

```
from datetime import datetime as dt

odds = [ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19,
        21, 23, 25, 27, 29, 31, 33, 35, 37, 39,
        41, 43, 45, 47, 49, 51, 53, 55, 57, 59]

this_minute = dt.today().minute

if this_minute in odds:
    print("Minuta nieparzysta")
else:
    print("Minuta parzysta")
```

Zad.3. Instrukcja warunkowa.

```
import time as t

today = t.strftime("%A")
if today == "Saturday":
    print("Python course")
elif today == "Sunday":
    print("No Python classes")
```

```
else:
    print("No classes at all")
```

Zad.4. Instrukcja *for*.

```
for i in [1,2,3,4]:
    print(i)
print("----------")

for i in "Python":
    print(i)
print("----------")

for i in range(4):
    print("Python")
print("----------")
```

Zad.5. Funkcja *range()*.

```
print(range(4))
print(list(range(4)))
print(list(range(4,10)))
print(list(range(0,8,2)))
print(list(range(8,0,-2)))
```

Zad.6. Pętla *while()*.

```
# while
licznik = 0

while licznik < 3:
    print("Inside while")
    licznik = licznik + 1
else:
    print("Inside else")


# do-while
```

```
while True:
    liczba = int(input("Wprowadź liczbe dodatnią: "))
    if liczba > 0:
        print("OK")
        break
    print("Liczna ujemna")
```

Zad.7. Wyrażenie *match - case*.

```
argument = 1

match argument:
    case 0:
        return "zero"
    case 1:
        return "one"
    case 2:
        return "two"
    case default:
        return "something"
```

Zad.8. Struktury danych - lista.

```
#lists
temps = [ 0.0, 100.0, -17.78, 27.5, 37.78, 7.39 ]
print(temps)
car_details = [ 'Kia', 'Sportage', 1.6, 3200]
print(car_details)
list_of_lists = [ [ 1, 2, 3], ['a', 'b', 'c' ], [
'Jeden','Dwa','Trzy' ] ]
print(list_of_lists)
```

Zad.9. Struktury danych - lista.

```
vowels = ['a', 'e', 'i', 'o', 'u']
word = input("Write a word: ")
found = []
for letter in word:
    if letter in vowels:
        if letter not in found:
            found.append(letter)

for vowel in found:
    print(vowel)
```

Zad.10. Struktury danych - operacje na listach.

```
numbers = []
print(len(numbers))

numbers.append(10)
print(numbers)

numbers = [1,2,3,4]
print(numbers)

print("Usuniecie ----------------------------")
# usuniecie elementu o okreslonej wartosci
numbers.remove(1)
print(numbers)

# usuniecie elementu o okreslonym indeksie
del_num = numbers.pop(1)
print(numbers)
print(del_num)

print("Rozszerzenie ----------------------------")
# rozszerzenie o liste obiektów
numbers.extend([5,6])
print(numbers)

# rozszerzenie o obiekt (1) wstawiony PRZED indeksem(0)
numbers.insert(0,1)
print(numbers)

numbers.insert(1,2)
print(numbers)

print("Kopiowanie ----------------------------")
```

```
# kopiowanie
numbers = list(range(10))
print(numbers)
numbers2 = numbers
numbers3 = numbers.copy()
print(numbers2)
numbers.append(100)
print(numbers)
print(numbers2)
print(numbers3)
```

Zad.11. Struktury danych - indeksowanie list.

```
s_letters = "AbCdEfGh"
letters = list(s_letters)
print(s_letters)
print(letters)

print(letters[0])
print(letters[2])
print(letters[-1])
print(letters[-3])

print(letters[3:])
print(letters[:2])
print(letters[::2])
print(letters[1:3])
print(letters[0:7:2])

print(''.join(letters[-3:]))
print(''.join(letters[::-1]))

for ch in letters:
    print('\t',ch)

for ch in letters[0:7:2]:
    print('\t \t',ch)
```

Zad.12. Struktury danych - odwzorowanie list.

```
lista = [1,3,5,7]
lista = [i+1 for i in lista]
print(lista)

lista = [1,3,4,5,7,8]
lista = [i for i in lista if i % 2 == 0 ]
print(lista)

lista = [1,3,4,5,7,8]
lista = ['Parzysta' if i%2 == 0 else 'Nieparzysta' for i in
lista]
print(lista)

lista = [1,3,4,5,7,8]
def func(i):
    if i % 2 == 0: return 'Parzyste'
    else: return 'Nieparzyste'
lista = [func(i) for i in lista]
print(lista)
```

Zad.13. Struktury danych - słownik.

```
person = {'Name': 'Tom',
          'Phone': '123456789',
          'Occupation': 'engineer',
          'Home planet': 'Earth'}
print(person)
print(person['Home planet'])

person['Age'] = 21
print(person)

#vowels = {'a' : 0, 'e' : 0, 'i' : 0, 'o' : 0, 'u' : 0}
vowels = ['a','e','i','o','u']
found_vowels ={}
```

```
word = input("Enter Your word: ")
for letter in word:
    if letter in vowels:
        found_vowels.setdefault(letter,0)
        found_vowels[letter] += 1

#k:v -> klucz:wartość
for k, v in sorted(found_vowels.items()):
    print(k, 'found', v, 'times')
```

Zad.14. Struktury danych - zbiór (*set*), krotka(*tuple*).

```
#zbiór - nie ma duplikatów
vowels = set('aeiou')
word = input("Enter Your word: ")
sum_sets = vowels.union(set(word))
print(sum_sets)
diff_sets = vowels.difference(set(word))
print(diff_sets)
common_part=vowels.intersection(set(word))
print(common_part)

#krotka(tuple) vs string
t = ('P','y','t','h','o','n')
print(t)
t = ('Python')
print(type(t))
t = ('Python',)
print(t)
print(type(t))
```

Zad.15. Funkcje.

```
def search4vowels():
    vowels = set('aeiou')
    word = input("Podaj słowo: ")
    found = vowels.intersection(set(word))
    for vowel in found:
        print(vowel)
```

```
search4vowels()
```

Zad.16. Funkcje - zwracanie wartości.

```
def search4vowels(word):
    vowels = set('aeiou')
    found = vowels.intersection(set(word))
    return bool(found)

print(search4vowels("Test"))
```

```
def search4vowels(word):
    vowels = set('aeiou')
    return vowels.intersection(set(word))

print(search4vowels("atest"))
```

Zad.17. Funkcje - *help()*.

```
def search4letters(word:str, letters:str="aeiou") -> set:
    """Wyszukuje litery w słowie wejściowym"""
    return set(letters).intersection(set(word))

help(search4letters)
print(search4letters("atest"))
```

Zad.18. Funkcje rekurencyjne.

```
def change(phrase, position):
    if phrase[position].isupper():
         phrase = phrase[0:position] + phrase[position].lower()
+ phrase[position+1:]
    else:
         phrase = phrase[0:position] + phrase[position].upper()
+ phrase[position+1:]
    if position == len(phrase)-1: return phrase;
    return change(phrase, position+1)

txt = "Long long time ago. In the galaxy far away..."
```

```
print(change(txt, 0))
```

Zad.19. Funkcje o zmiennej liczbie parametrów.

```
def vargs_func(*args):
    print(args)

vargs_func('a','b','c')
vargs_func(1,2,3,4,5)

def vargs_func(*args):
    print("Liczba przekazanych parametrów:",len(args))
    for arg in args:
        print ("Wartość:",arg)

lista = [1,2,3,4]
vargs_func(1,lista,2,'xyz',3)


def vargs_func(**kwargs):
     print("Number of parameters:",len(kwargs))
     for key, item in kwargs.items():
          print ("Key ", key, "Vslue ", item)


vargs_func(a=1,b=2,c=3)


def vargs_func(*args, **kwargs):
    print("Number of parameters *args:",len(args))
    for arg in args:
        print ("Arg ", arg)


    print("Number of parameters **kwargs:",len(kwargs))
    for key, item in kwargs.items():
        print ("Key ", key, "Value ", item)


vargs_func(1,2,'x',[1,'a',2],a=1,b=2,c=3)
```

Zad.20. Zaimplementować wyszukiwanie binarne za pomocą funkcji rekurencyjnej.