Міністерство освіти і науки України

Державний університет “Житомирська політехніка”

Кафедра інженерії програмного забезпечення

Група: ВТ-21-1[1]

Програмування мовою Python

Лабораторна робота № 2

«Розгалуження та цикли»

Виконав: Бабушко А. С.

Прийняв: Морозов Д. С.

***Мета роботи:*** познайомитися із структурою розгалуження (if, ifelse, if-elif-else). Ознайомитися з циклічними конструкціями і їх використанням в мові Python. Навчитися працювати з числами і рядками використовуючи дані структури.

***Хід роботи:***

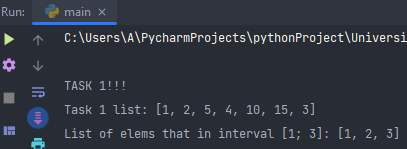
***Завдання на лабораторну роботу:***

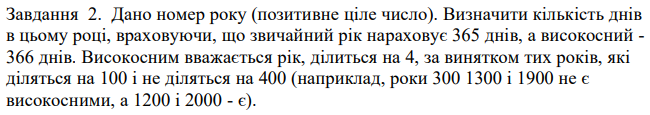
******

***Лістинг програми:***

*""" Lab 2. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python """  
import* math  
  
  
*# task 1  
def* task\_1\_interval\_int\_list(some\_int\_list):  
 some\_result\_list = []  
 *for* index *in* range(len(some\_int\_list)):  
 *if* 1 <= some\_int\_list[index] <= 3:  
 some\_result\_list.append(some\_int\_list[index])  
  
 *return* some\_result\_list  
  
  
print('\nTASK 1!!!')  
task\_1\_list = [1, 2, 5, 4, 10, 15, 3]  
print(f'Task 1 list: {task\_1\_list}')  
print(f'List of elems that in interval [1; 3]: {task\_1\_interval\_int\_list(task\_1\_list)}')

***Результат програми:***

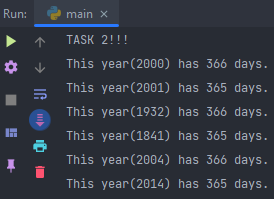
******

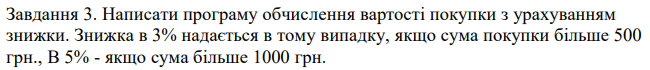
******

***Лістинг програми:***

*# task 2  
def* task\_2\_year\_days(year):  
 *if* is\_leap\_year(year):  
 print(f'This year({year}) has 366 days.')  
 *else*:  
 print(f'This year({year}) has 365 days.')  
  
  
*def* is\_leap\_year(year):  
 *if* year % 400 == 0 *or* (year % 4 == 0 *and* year % 100 != 0):  
 *return True  
 return False*print('\nTASK 2!!!')  
task\_2\_year = [2000, 2001, 1932, 1841, 2004, 2014]  
*for* i *in* range(len(task\_2\_year)):  
 task\_2\_year\_days(task\_2\_year[i])

***Результат програми:***

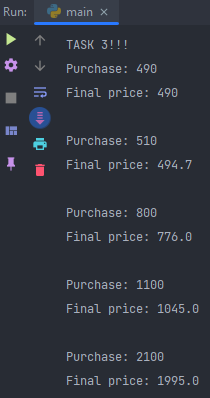
******

******

***Лістинг програми:***

*# task 3  
def* task\_3\_purchase\_discount(purchase):  
 *if* 500 <= purchase < 1000:  
 *return* purchase - purchase \* 0.03  
 *if* purchase >= 1000:  
 *return* purchase - purchase \* 0.05  
 *return* purchase  
  
  
print('\nTASK 3!!!')  
task\_3\_purchases = [490, 510, 800, 1100, 2100]  
*for* i *in* range(len(task\_3\_purchases)):  
 print(f'Purchase: {task\_3\_purchases[i]}\nFinal price: {task\_3\_purchase\_discount(task\_3\_purchases[i])}\n')

***Результат програми:***

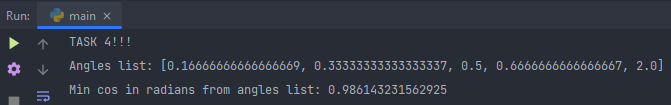
******

******

***Лістинг програми:***

*# task 4  
def* task\_4\_min\_cosine(angles):  
 *return* math.cos(min(angles))  
  
  
print('\nTASK 4!!!')  
angles\_list = [math.radians(30 / math.pi), math.radians(60 / math.pi), math.radians(90 / math.pi),  
 math.radians(120 / math.pi), math.radians(360 / math.pi)]  
print(f'Angles list: {angles\_list}')  
print(f'Min cos in radians from angles list: {task\_4\_min\_cosine(angles\_list)}')

***Результат програми:***

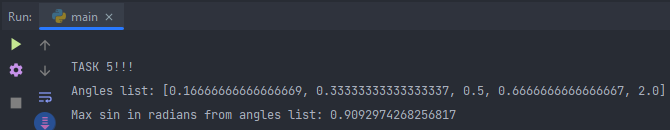
******

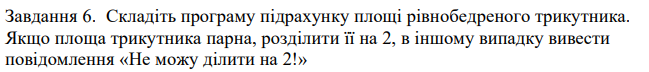
******

***Лістинг програми:***

*# task 5  
def* task\_5\_max\_sin(angles):  
 *return* math.sin(max(angles))  
  
  
print('\nTASK 5!!!')  
print(f'Angles list: {angles\_list}')  
print(f'Max sin in radians from angles list: {task\_5\_max\_sin(angles\_list)}')

***Результат програми:***

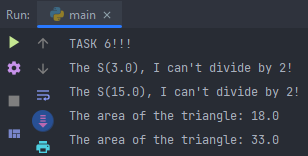
******

******

***Лістинг програми:***

*# task 6  
def* task\_6\_isosceles\_triangle(sides):  
 s = (sides[0] \* sides[1]) / 2  
 *if* s % 2 == 0:  
 s /= 2  
 print(f'The area of the triangle: {s}')  
 *else*:  
 print(f"The S({s}), I can\'t divide by 2!")  
  
  
print('\nTASK 6!!!')  
left\_right\_sides = 1  
base\_side = 2  
*for* x *in* range(1, 13, 3):  
 task\_6\_isosceles\_triangle([left\_right\_sides + x, base\_side + x])

***Результат програми:***

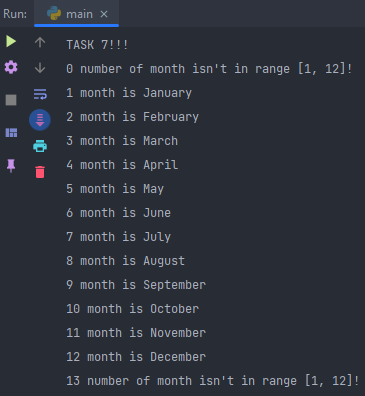
******

******

***Лістинг програми:***

*# task 7  
def* task\_7\_english\_month(month\_number):  
 months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October',  
 'November', 'December']  
 *if* 1 <= month\_number <= 12:  
 print(f'{month\_number} month is {months[month\_number - 1]}')  
 *else*:  
 print(f'{month\_number} number of month isn\'t in range [1, 12]!')  
  
  
print('\nTASK 7!!!')  
*for* i *in* range(0, 14):  
 task\_7\_english\_month(i)

***Результат програми:***

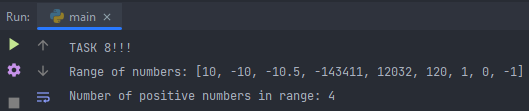
******

******

***Лістинг програми:***

*# task 8  
def* task\_8\_positive\_numbers(numbers):  
 positive\_counter = 0  
 *for* number *in* numbers:  
 *if* number >= 1:  
 positive\_counter += 1  
 *return* positive\_counter  
  
  
print('\nTASK 8!!!')  
some\_numbers = [10, -10, -10.5, -143411, 12032, +120, 1, 0, -1]  
print(f'Range of numbers: {some\_numbers}')  
print(f'Number of positive numbers in range: {task\_8\_positive\_numbers(some\_numbers)}')

***Результат програми:***

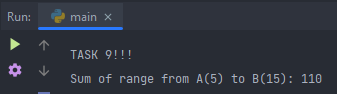
******

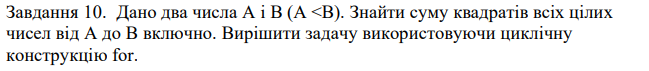
******

***Лістинг програми:***

*# task 9  
def* task\_9\_sum\_some\_range(a, b):  
 sum\_some\_range = 0  
 *for* some\_num *in* range(a, b + 1, 1):  
 sum\_some\_range += some\_num  
  
 *return* sum\_some\_range  
  
  
print('\nTASK 9!!!')  
A = 5  
B = 15  
print(f'Sum of range from A({A}) to B({B}): {task\_9\_sum\_some\_range(A, B)}')

***Результат програми:***

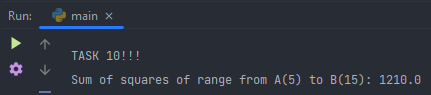
******

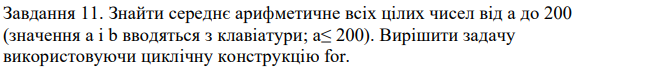
******

***Лістинг програми:***

*# task 10  
def* task\_10\_sum\_squares\_some\_range(a, b):  
 sum\_squares\_some\_range = 0  
 *for* some\_num *in* range(a, b + 1, 1):  
 sum\_squares\_some\_range += math.pow(some\_num, 2)  
  
 *return* sum\_squares\_some\_range  
  
  
print('\nTASK 10!!!')  
print(f'Sum of squares of range from A({A}) to B({B}): {task\_10\_sum\_squares\_some\_range(A, B)}')

***Результат програми:***

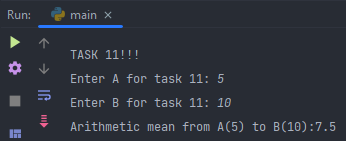
******

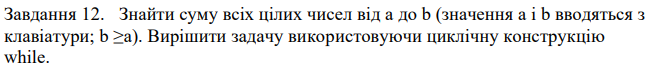
******

***Лістинг програми:***

*# task 11  
def* task\_11\_arithmetic\_mean(a, b):  
 arithmetic\_mean\_sum = 0  
 count\_nums = 0  
 *for* some\_num *in* range(a, b + 1):  
 arithmetic\_mean\_sum += some\_num  
 count\_nums += 1  
  
 *return* arithmetic\_mean\_sum / count\_nums  
  
  
*def* enter\_A\_and\_B():  
 *while True*:  
 a = int(input('Enter A for task 11: '))  
 b = int(input('Enter B for task 11: '))  
 *if* a < b:  
 *break  
  
 return* [a, b]  
  
  
print('\nTASK 11!!!')  
some\_range = enter\_A\_and\_B()  
print(f'Arithmetic mean from A({some\_range[0]}) to B({some\_range[1]}):'  
 f'{task\_11\_arithmetic\_mean(some\_range[0], some\_range[1])}')

***Результат програми:***

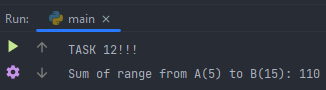
******

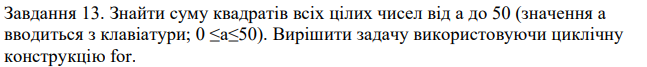
******

***Лістинг програми:***

*# task 12  
def* task\_12\_sum\_some\_range(a, b):  
 sum\_some\_range = 0  
 *while* a <= b:  
 sum\_some\_range += a  
 a += 1  
  
 *return* sum\_some\_range  
  
  
print('\nTASK 12!!!')  
print(f'Sum of range from A({A}) to B({B}): {task\_12\_sum\_some\_range(A, B)}')

***Результат програми:***

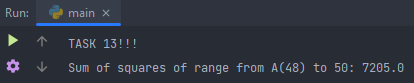
******

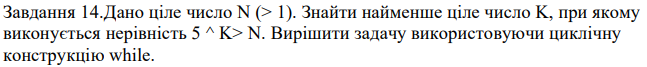
******

***Лістинг програми:***

*# task 13  
def* task\_13\_sum\_some\_square\_range(a):  
 sum\_some\_squares = 0  
 *for* sum\_num *in* range(a, 51, 1):  
 sum\_some\_squares += math.pow(sum\_num, 2)  
  
 *return* sum\_some\_squares  
  
  
print('\nTASK 13!!!')  
A = 48  
print(f'Sum of squares of range from A({A}) to 50: {task\_13\_sum\_some\_square\_range(A)}')

***Результат програми:***

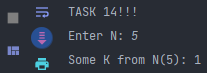
******

******

***Лістинг програми:***

*# task 14  
def* task\_14\_find\_K(N\_number):  
 K = N\_number  
 K\_array = [K]  
 *while* math.pow(5, K) > N\_number:  
 K -= 1  
 K\_array.append(K)  
  
 *return* min(K\_array)  
  
  
*def* enter\_N():  
 *while True*:  
 some\_N = int(input('Enter N: '))  
 *if* some\_N > 1:  
 *break  
  
 return* some\_N  
  
  
print('\nTASK 14!!!')  
N = enter\_N()  
print(f'Some K from N({N}): {task\_14\_find\_K(N)}')

***Результат програми:***

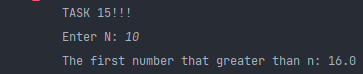
******

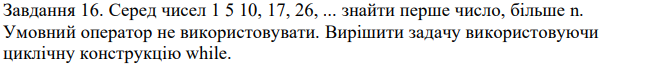
******

***Лістинг програми:***

*# task 15  
def* task\_15\_find\_number\_greater\_n(n):  
 *for* number *in* range(1, n, 1):  
 some\_number = math.pow(number, 2)  
 *if* some\_number > n:  
 *return* some\_number  
 *else*:  
 *continue  
  
 return* n  
  
  
print('\nTASK 15!!!')  
N = enter\_N()  
print(f'The first number that greater than n: {task\_15\_find\_number\_greater\_n(N)}')

***Результат програми:***

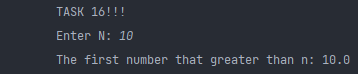
******

******

***Лістинг програми:***

*# task 16  
def* task\_16\_find\_number\_greater\_n(n):  
 some\_number = 1  
 iterator = 2  
 *while* some\_number < n:  
 some\_number = math.pow(iterator, 2) + 1  
 iterator += 1  
  
 *return* some\_number  
  
  
print('\nTASK 16!!!')  
N = enter\_N()  
print(f'The first number that greater than n: {task\_16\_find\_number\_greater\_n(N)}')

***Результат програми:***

******

***Увесь лістинг програми:***

*""" Lab 2. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python """  
import* math  
  
  
*# task 1  
def* task\_1\_interval\_int\_list(some\_int\_list):  
 some\_result\_list = []  
 *for* index *in* range(len(some\_int\_list)):  
 *if* 1 <= some\_int\_list[index] <= 3:  
 some\_result\_list.append(some\_int\_list[index])  
  
 *return* some\_result\_list  
  
  
print('\nTASK 1!!!')  
task\_1\_list = [1, 2, 5, 4, 10, 15, 3]  
print(f'Task 1 list: {task\_1\_list}')  
print(f'List of elems that in interval [1; 3]: {task\_1\_interval\_int\_list(task\_1\_list)}')  
  
  
*# task 2  
def* task\_2\_year\_days(year):  
 *if* is\_leap\_year(year):  
 print(f'This year({year}) has 366 days.')  
 *else*:  
 print(f'This year({year}) has 365 days.')  
  
  
*def* is\_leap\_year(year):  
 *if* year % 400 == 0 *or* (year % 4 == 0 *and* year % 100 != 0):  
 *return True  
 return False*print('\nTASK 2!!!')  
task\_2\_year = [2000, 2001, 1932, 1841, 2004, 2014]  
*for* i *in* range(len(task\_2\_year)):  
 task\_2\_year\_days(task\_2\_year[i])  
  
  
*# task 3  
def* task\_3\_purchase\_discount(purchase):  
 *if* 500 <= purchase < 1000:  
 *return* purchase - purchase \* 0.03  
 *if* purchase >= 1000:  
 *return* purchase - purchase \* 0.05  
 *return* purchase  
  
  
print('\nTASK 3!!!')  
task\_3\_purchases = [490, 510, 800, 1100, 2100]  
*for* i *in* range(len(task\_3\_purchases)):  
 print(f'Purchase: {task\_3\_purchases[i]}\nFinal price: {task\_3\_purchase\_discount(task\_3\_purchases[i])}\n')  
  
  
*# task 4  
def* task\_4\_min\_cosine(angles): *return* math.cos(min(angles))  
  
  
print('\nTASK 4!!!')  
angles\_list = [math.radians(30 / math.pi), math.radians(60 / math.pi), math.radians(90 / math.pi),  
 math.radians(120 / math.pi), math.radians(360 / math.pi)]  
print(f'Angles list: {angles\_list}')  
print(f'Min cos in radians from angles list: {task\_4\_min\_cosine(angles\_list)}')  
  
  
*# task 5  
def* task\_5\_max\_sin(angles): *return* math.sin(max(angles))  
  
  
print('\nTASK 5!!!')  
print(f'Angles list: {angles\_list}')  
print(f'Max sin in radians from angles list: {task\_5\_max\_sin(angles\_list)}')  
  
  
*# task 6  
def* task\_6\_isosceles\_triangle(sides):  
 s = (sides[0] \* sides[1]) / 2  
 *if* s % 2 == 0:  
 s /= 2  
 print(f'The area of the triangle: {s}')  
 *else*:  
 print(f"The S({s}), I can\'t divide by 2!")  
  
  
print('\nTASK 6!!!')  
left\_right\_sides = 1  
base\_side = 2  
*for* x *in* range(1, 13, 3):  
 task\_6\_isosceles\_triangle([left\_right\_sides + x, base\_side + x])  
  
  
*# task 7  
def* task\_7\_english\_month(month\_number):  
 months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October',  
 'November', 'December']  
 *if* 1 <= month\_number <= 12:  
 print(f'{month\_number} month is {months[month\_number - 1]}')  
 *else*:  
 print(f'{month\_number} number of month isn\'t in range [1, 12]!')  
  
  
print('\nTASK 7!!!')  
*for* i *in* range(0, 14):  
 task\_7\_english\_month(i)  
  
  
*# task 8  
def* task\_8\_positive\_numbers(numbers):  
 positive\_counter = 0  
 *for* number *in* numbers:  
 *if* number >= 1:  
 positive\_counter += 1  
 *return* positive\_counter  
  
  
print('\nTASK 8!!!')  
some\_numbers = [10, -10, -10.5, -143411, 12032, +120, 1, 0, -1]  
print(f'Range of numbers: {some\_numbers}')  
print(f'Number of positive numbers in range: {task\_8\_positive\_numbers(some\_numbers)}')  
  
  
*# task 9  
def* task\_9\_sum\_some\_range(a, b):  
 sum\_some\_range = 0  
 *for* some\_num *in* range(a, b + 1, 1):  
 sum\_some\_range += some\_num  
  
 *return* sum\_some\_range  
  
  
print('\nTASK 9!!!')  
A = 5  
B = 15  
print(f'Sum of range from A({A}) to B({B}): {task\_9\_sum\_some\_range(A, B)}')  
  
  
*# task 10  
def* task\_10\_sum\_squares\_some\_range(a, b):  
 sum\_squares\_some\_range = 0  
 *for* some\_num *in* range(a, b + 1, 1):  
 sum\_squares\_some\_range += math.pow(some\_num, 2)  
  
 *return* sum\_squares\_some\_range  
  
  
print('\nTASK 10!!!')  
print(f'Sum of squares of range from A({A}) to B({B}): {task\_10\_sum\_squares\_some\_range(A, B)}')  
  
  
*# task 11  
def* task\_11\_arithmetic\_mean(a, b):  
 arithmetic\_mean\_sum = 0  
 count\_nums = 0  
 *for* some\_num *in* range(a, b + 1):  
 arithmetic\_mean\_sum += some\_num  
 count\_nums += 1  
  
 *return* arithmetic\_mean\_sum / count\_nums  
  
  
*def* enter\_A\_and\_B():  
 *while True*:  
 a = int(input('Enter A for task 11: '))  
 b = int(input('Enter B for task 11: '))  
 *if* a < b:  
 *break  
  
 return* [a, b]  
  
  
print('\nTASK 11!!!')  
some\_range = enter\_A\_and\_B()  
print(f'Arithmetic mean from A({some\_range[0]}) to B({some\_range[1]}):'  
 f'{task\_11\_arithmetic\_mean(some\_range[0], some\_range[1])}')  
  
  
*# task 12  
def* task\_12\_sum\_some\_range(a, b):  
 sum\_some\_range = 0  
 *while* a <= b:  
 sum\_some\_range += a  
 a += 1  
  
 *return* sum\_some\_range  
  
  
print('\nTASK 12!!!')  
print(f'Sum of range from A({A}) to B({B}): {task\_12\_sum\_some\_range(A, B)}')  
  
  
*# task 13  
def* task\_13\_sum\_some\_square\_range(a):  
 sum\_some\_squares = 0  
 *for* sum\_num *in* range(a, 51, 1):  
 sum\_some\_squares += math.pow(sum\_num, 2)  
  
 *return* sum\_some\_squares  
  
  
print('\nTASK 13!!!')  
A = 48  
print(f'Sum of squares of range from A({A}) to 50: {task\_13\_sum\_some\_square\_range(A)}')  
  
  
*# task 14  
def* task\_14\_find\_K(N\_number):  
 K = N\_number  
 K\_array = [K]  
 *while* math.pow(5, K) > N\_number:  
 K -= 1  
 K\_array.append(K)  
  
 *return* min(K\_array)  
  
  
*def* enter\_N():  
 *while True*:  
 some\_N = int(input('Enter N: '))  
 *if* some\_N > 1:  
 *break  
  
 return* some\_N  
  
  
print('\nTASK 14!!!')  
N = enter\_N()  
print(f'Some K from N({N}): {task\_14\_find\_K(N)}')  
  
  
*# task 15  
def* task\_15\_find\_number\_greater\_n(n):  
 *for* number *in* range(1, n, 1):  
 some\_number = math.pow(number, 2)  
 *if* some\_number > n:  
 *return* some\_number  
 *else*:  
 *continue  
  
 return* n  
  
  
print('\nTASK 15!!!')  
N = enter\_N()  
print(f'The first number that greater than n: {task\_15\_find\_number\_greater\_n(N)}')  
  
  
*# task 15  
def* task\_16\_find\_number\_greater\_n(n):  
 some\_number = 1  
 iterator = 2  
 *while* some\_number < n:  
 some\_number = math.pow(iterator, 2) + 1  
 iterator += 1  
  
 *return* some\_number  
  
  
print('\nTASK 16!!!')  
N = enter\_N()  
print(f'The first number that greater than n: {task\_16\_find\_number\_greater\_n(N)}')

***Висновок:*** під час виконання лабораторної роботи було отримано навички з написання простих скриптів з використанням циклів for та while, перевірок умови if на мові Python.