mission 2-01

達成目標 80 秒

式を見て処理順を示せ①

式に含まれる演算子の処理順を書き込んでください。

1 ＋ 2 ＊ 3 ⋯⋯▶ 1 ＋ 2 ＊ 3
（②：1 ＋ 2、①：2 ＊ 3）

1	1 + 2 * 3
2	(1 + 2) * 3 * 4
3	1 * 2 * 3
4	1 + (2 * 3) * 4
5	1 + 2 - 3
6	1 + (2 + 3) * 4
7	1 / 2 + 3 * 4
8	1 + 2 * 3 * 4 + 5
9	1 / 2 * 3 * 4 + 5 * 6 - 7
10	1 * 2 - 3 * (4 + 5) * 6 - 7

達成目標 100 秒

式を見て計算結果を示せ①

式を見て処理順に計算し、結果を書き込んでください。

1 + (2 − 3 ＊ 4) ＊ 5

12

-10

-50

-49

1 `1 + (2 * 3) * 4`

2 `1 / 2 + 3 * 4`

3 `1 / 2 * 3 + 4 + 5 * 6 - 7`

4 `1 * 2 - 3 * (4 + 5) * 2 - 7`

5 `1 + (2 * 3) * (4 + 5) * 2`

6 `(1 + 2) - 3 * (4 - 5) * 6`

mission 2-03

達成目標 50 秒

プログラムを見て変数に印を付けろ

プログラムを見て、変数の部分の下に印を付けてください。

```
normal_price = 100
selling_price = normal_price * 1.1
print(selling_price)
```

1
```
text = 'Hello'
print(text)
```

2
```
year = 2019
wareki = year - 2018
print(year)
print(wareki)
```

3
```
price = 1000
quantity = 10
sales = price * quantity
print(sales)
```

4
```
sales = 9980
payment = 10000
change = payment - sales
print(payment)
print(change)
```

達成目標 40 秒

適切な変数名を選択せよ

変数名として適切なものを選択してください。

① 2021　② year　③ 2021year

1 ① 文字列　② text　③ TEXT

2 ① 1st_check　② CHECK1　③ check1

3 ① fileName　② file/name　③ file_name　④ FileName

4 ① password　② pass　③ p@ss　④ PASS

mission 2-05

達成目標 40 秒

累算代入文の結果を示せ

プログラムを見て、最後に表示される計算結果を書き込んでください。

```
year = 2000
year += 21
print(year)  2021
```

1
```
i = 0
i += 1
print(i)
```

2
```
num = 10
num -= 5
print(num)
```

3
```
num = 10
num /= 5
print(num)
```

4
```
num = 10
num += 5 * 2
print(num)
```

5
```
text = '山'
text += '川'
print(text)
```

6
```
price = 1000
discount = 100
price -= discount
print(price)
```

※「#」から改行まではコメント文となります（次ページの問い参照）。コメント文はプログラムにメモを書き込むもので、Pythonインタプリタには無視されるので、実行結果に影響しません。

達成目標 50 秒

型を変換するべき変数を選べ

int関数で数値型に、またはstr関数で文字列型に、
型を変換するべき変数の下に印を付けてください。

```
year_text = '2021'
wareki = year_text - 2018
```

1
```
# 数値と文字列を連結
num = 10
text = num + '個'
```

2
```
# 数値と文字列を連結
text = '10'
num = text + 20
```

3
```
# 数値と文字列を連結
price = 1500
text = '価格'
text += price + '円'
```

4
```
# 価格と税の合計を
# 文字列と連結
price = 1500
tax = 150
price += tax
print(price + '円')
```

5
```
# 価格と割引率から割引後の価格を計算する
price = 1080
discount_rate_text = '2'
price -= price * discount_rate_text / 10
print(price)
```

mission 2-07

達成目標 120 秒

エラーの原因を選べ

プログラムを見て、エラーの原因として適切なものを選択肢から選択してください。

```
001  x, y = 20, 19, 32
002  print('x軸の値は' + str(x) + ' y軸の値は' + str(y))
```

エラーメッセージ

```
ValueError: too many values to unpack (expected 2)
```

① 1行目で作成されている変数名が不正

② 1行目の代入文で左辺の変数の数と右辺の値の数が一致していない

③ 2行目で型の変換が必要

```
001  year_text = '2021'
002  wareki = year_text - 2018
003  print(wareki)
```

エラーメッセージ

```
1 TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

① 1行目で作成している変数名が不正

② 2行目で作成している変数名が不正

③ 2行目で型の変換が必要

2

```
001  price = 1500
002  tax = 150
003  price += tax
004  print(price + '円')
```

エラーメッセージ

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

① 1行目で作成している変数名が不正

② 3行目で型の変換が必要

③ 4行目で型の変換が必要

3

```
001  class = '4'
002  student_name = '服部 太朗'
003  print(class + '組 ' + student_name)
```

エラーメッセージ

```
SyntaxError: invalid syntax
```

① 1行目で作成している変数名が不正

② 1行目の代入文で左辺の変数の数と右辺の値の数が一致していない

③ 3行目で型の変換が必要

4

```
001  1price = '100円'
002  2price = '200円'
003  price_text = '商品1は' + 1price
004  price_text += ','
005  price_text += '商品2は' + 2price
006  print(price_text)
```

エラーメッセージ

```
SyntaxError: invalid syntax
```

① 1行目で作成している変数名が不正

② 3行目で型の変換が必要

③ 5行目で型の変換が必要

5

```
001  price = 1100
002  discount = 200
003  price_text = '割引後価格'
004  price -= discount
005  price_text += price + '円'
006  print(price_text)
```

エラーメッセージ

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

① 1行目の代入文で左辺の変数の数と右辺の値の数が一致していない

② 4行目で型の変換が必要

③ 5行目で型の変換が必要

mission 3-01

達成目標 40 秒

プログラムを見て関数・メソッドに印を付けろ

プログラムを見て、関数・メソッドの名前の下に印を付けてください。


```
print('出力結果: ' + text.lower().replace('e', 'a'))
```

1
```
print('ninja'.count('n'))
```

2
```
text = input()
print('length: ' + str(len(text)))
```

3
```
text = input('Input something: ')
print(text + ' was input')
```

4
```
text = input()
lower_text = text.lower()
print(lower_text.replace('i', 'a').find('a'))
```

※「2」で使用しているlen関数についてはP.94参照

mission 3-02

達成目標 50 秒

式を見て処理順を示せ②

式に含まれる下線を引かれた演算子・関数・メソッドの処理順を書き込んでください。

```
④   ③   ①      ②
print('出力結果: ' + text.lower().replace('e', 'a'))
```

1
```
'出力結果: ' + text.lower()
```

2
```
lyrics.find('リンダ' * 3)
```

3
```
'株式会社libroworks'.count('r', len('株式会社'))
```

4
```
print('計算結果: ' + str(price + tax))
```

5
```
print('Tは' + str(text.upper().find('T')) + '文字目')
```

達成目標 120 秒

式を見て処理順を示せ③

式に含まれる演算子の処理順を書き込んでください（カッコは演算子ではありません）。


```
i ① + 1 ② < 10 ④ and i ③ < 100
```

1
```
not True and True
```

2
```
True or not False and True
```

3
```
(True or not False) and True
```

4
```
12 < a + i and a + i < 20
```

5
```
text != password or text == ''
```

6
```
text != '山' or text != '川' or text == ''
```

7
```
not a < 16 or not 65 < a and a < i + 99
```

mission 3-04

達成目標 150 秒

出力結果はTrueかFalseか

プログラムを見て、最後に表示される真偽値を書き込んでください。


```
text = 'abc'
print(text == 'abc')
```

True

1
```
print(100 >= 100)
```

2
```
print(not 100 < 100)
```

3
```
text = 'password'
print(text != 'password')
```

4
```
age = 21
print(age >= 20)
```

5
```
print(True and False)
```

6
```
print(True or False)
```

7
```
flg = not True and True
print(flg)
```

8
```
flg = True or False and True
print(flg)
```

```
9
text = '山'
print(text == '山' or text == '川')
```

```
10
flg = (True or False) and True
print(not flg)
```

```
11
age = 65
print(age < 18 or 65 < age)
```

```
12
text = 'ninja'
print(text.upper().replace('I', 'A') == 'NANJA')
```

```
13
score, best_score = 70, 72
print(score >= best_score)
```

```
14
bmi = 21.5
print(18.5 <= bmi and bmi < 25)
```

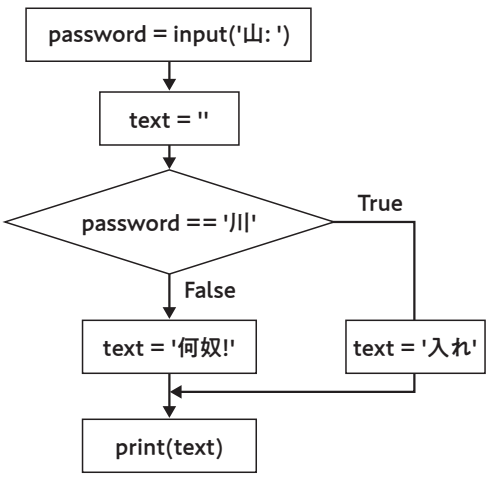

mission 3-05

達成目標 300 秒

フローチャートを書け

プログラムを見て、フローチャートを書いてください。


```
password = input('山: ')
text = ''
if password == '川':
    text = '入れ'
else:
    text = '何奴！'
print(text)
```



1

```
ninja = True
if ninja:
    print('にんにん')
else:
    print('なむなむ')
```

2

```
age = int(input())
if 40 <= age:
    print('上忍')
elif 30 <= age:
    print('中忍')
```

3

```
mail_address = True
password = False
if mail_address:
    if password:
        print('ログイン成功')
    else:
        print('ログイン失敗')
else:
    print('ログイン失敗')
```

mission 4-01

達成目標 120 秒

出力結果を書け①

プログラムを見て、最後に表示される出力結果を書き込んでください。

```
iroha_list = ['い', 'ろ', 'は', 'に', 'ほ', 'へ', 'と']
print(iroha_list[3])
```

に

1

```
grade = ['松', '竹', '梅']
print(grade[2])
```

2

```
class_list = ['下忍', '上忍']
class_list.insert(1, '中忍')
print(class_list)
```

3

```
ranking = ['前田', '大島', '篠田']
ranking[0], ranking[1] = ranking[1], ranking[0]
print(ranking)
```

mission 4-02

達成目標 300 秒

スライスの結果を書け

プログラムを見て、最後に表示される出力結果を書き込んでください。
すべて、例と同じリストalphabetsをスライスしているものとします。

```
alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(alphabets[1:5])
['b', 'c', 'd', 'e']
```

1
```
print(alphabets[4:])
```

2
```
print(alphabets[2:5])
```

3
```
print(alphabets[:-5])
```

4
```
print(alphabets[2:-2])
```

5
```
print(alphabets[:4:2])
```

6
```
print(alphabets[1::3])
```

7
```
print(alphabets[2::-1])
```

8
```
print(alphabets[4:1:-1])
```

mission 4-03

達成目標 180 秒

スライスを書け

出力結果を見て、プログラムの空白部分を書き込んでください。
すべて、例と同じリストalphabetsをスライスしているものとします。

```
alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(alphabets[ 1:5 ])
['b', 'c', 'd', 'e']
```

1
```
print(alphabets[          ])
['d', 'e', 'f', 'g']
```

2
```
print(alphabets[          ])
['a', 'b', 'c']
```

3
```
print(alphabets[          ])
['c', 'd']
```

4
```
print(alphabets[          ])
['e', 'f']
```

5
```
print(alphabets[          ])
['a', 'c', 'e', 'g']
```

6
```
print(alphabets[          ])
['b', 'e']
```

7
```
print(alphabets[          ])
['d', 'c', 'b', 'a']
```

8
```
print(alphabets[          ])
['f', 'e']
```

mission 4-04

達成目標 120 秒

出力結果を書け②

プログラムを見て、最後に表示される出力結果を書き込んでください。

```
alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(len(alphabets))
```

7

1

```
texts = ('ninja', 'samurai', 'fujiyama')
print(texts[2][4:])
```

2

```
texts = ('ninja', 'samurai', 'fujiyama')
print(texts[1].upper())
```

3

```
first_names = ['佐助', '才蔵', '六郎', '十蔵']
print('半蔵' in first_names)
```

4

```
birth_years = (1972, 1974, 1977, 1973, 1972)
print(max(birth_years) - min(birth_years))
```

5
```
birth_years = [1972, 1974, 1977, 1973, 1972]
birth_years.sort()
print(birth_years[2])
```

6
```
birth_years = [1972, 1974, 1977, 1973, 1972]
birth_years_sorted = sorted(birth_years)
print(birth_years.index(1973) - birth_years_sorted.index(1973))
```

7
```
animals = ['dog', 'elephant', 'shark']
birds = ['hawk', 'dove', 'robin']
animals.extend(birds)
print(len(animals))
```

8
```
animals = ['dog', 'elephant', 'shark']
birds = ['hawk', 'dove', 'robin']
animals.append(birds)
print(len(animals))
```

mission 5-01

達成目標 300 秒

出力結果を書け③

プログラムを見て、最後に表示される出力結果を書き込んでください。

```
targets = ['マトA', 'マトB', 'マトC']
for target in targets:
    print(target + 'に手裏剣を投げた')
```

マトAに手裏剣を投げた

マトBに手裏剣を投げた

マトCに手裏剣を投げた

1

```
point_list = [92, 88, 84]
for point in point_list:
    print(point, '点です')
```

2

```
point_list = [92, 88, 84]
for point in point_list:
    if 85 <= point:
        print(point, '点です')
```

3
```
point_list = [92, 88, 84]
for count, point in enumerate(point_list, start=1):
    print(count, '回目の点数は', point, '点です')
```

4
```
for count in range(10):
    print('Pizza')
```

5
```
for count in range(3, 0, -1):
    print(count)
print('Liftoff')
```

6
```
given_names = ['Samuel', 'Michael']
family_names = ['Johnson', 'Jackson']
for given_name in given_names:
    for family_name in family_names:
        print(given_name, family_name)
```


mission 5-02

達成目標 300 秒

出力結果を書け④

プログラムを見て、表示される出力結果を書き込んでください。

```
number_list = [number ** 2 for number in range(1, 6)]

print(number_list)
```

[1, 4, 9, 16, 25]

1

```
number_list = [number for number in range(10, 51, 10)]

print(number_list)
```

2

```
number_list = [number for number in range(1, 20)
               if number % 3 == 0]
print(number_list)
```

3

```
words = ['Man', 'Soul', 'Quiz']

ultra_words = ['Ultra' + word for word in words]

print(ultra_words)
```

4

```
point_list = [91, 68, 75, 80]
passed = [point for point in point_list if point >= 80]
print(passed)
```

5

```
price_list = [1000, 1400, 2000, 1200]
wish_list = [price for price in price_list
             if price * 1.1 <= 1500]
print(wish_list)
```

6

```
guests = ['Ms. Larson', 'Mr. Evans', 'Ms. Olsen', 'Mr. Renner']
female_guests = [guest for guest in guests if 'Ms.' in guest]
print(female_guests)
```

7

```
member_list = [['Okamura', 'Yabe'],
               ['Takeda'],
               ['Arino', 'Hamaguchi']]
duo_list = [group for group in member_list if len(group) == 2]
print(duo_list)
```


mission 5-03

達成目標 60 秒

どの構文を使うのが最適かを選べ

問題文に書かれたような処理を行いたい場合、どの構文を使うのが最適か選択してください。

処理：所持金が300円以上である限り1,000円の商品を買いつづける

① for文とリストの組み合わせによる繰り返し
② for文とrange関数の組み合わせによる繰り返し
③ while文による繰り返し

1

処理：孔明を3回訪ねる

① for文とリストの組み合わせによる繰り返し
② for文とrange関数の組み合わせによる繰り返し
③ while文による繰り返し

2

処理：孔明が「はい」と言うまで訪ねる

① for文とリストの組み合わせによる繰り返し
② for文とrange関数の組み合わせによる繰り返し
③ while文による繰り返し

3

処理：孔明がいる可能性のある場所を1つずつ順番に訪ねる

① for文とリストの組み合わせによる繰り返し
② for文とrange関数の組み合わせによる繰り返し
③ while文による繰り返し

達成目標 60 秒

出力結果を書け⑤

プログラムを見て、表示される出力結果を書き込んでください。

```
in_pocket = 3500
while 300 <= in_pocket:
    print(in_pocket)
    in_pocket -= 1000
```

```
3500
2500
1500
500
```

1

```
total = 13
while total <= 21:
    print(total)
    total += 3
```

2

```
number = 2
while number <= 50:
    print(number)
    number *= 2
```


mission 5-05

達成目標 240 秒

出力結果を書け⑥

プログラムを見て、表示される出力結果を書き込んでください。

```
for i in range(2, 6):       # iは2から6-1までの数値
    for j in range(2, i):   # jは2からi-1までの数値
        if i % j == 0:      # iがjで割り切れる場合
            break
    else:    # breakが行われなかった = 割り切れる数がなかった
        print(i, 'は素数')
```

```
2 は素数
3 は素数
5 は素数
```

1

```
for target in ['マトンカレー', 'マント', 'マトリョーシカ']:
    if 'マト' not in target:
        continue
    print(target + 'に手裏剣を投げた')
```

2

```
places = ['カバンの中', 'つくえの中']
for place in places:
    if place == '探しもの':
        break
    print(place)
else:
    print('探したけれど見つからない')
```

3

```
word = 'brothers'
for letter in word:
    if letter.upper() == 'T':
        print('T!')
        break
    else:
        print(letter.upper() + '…')
else:
    print('T was not found')
```


mission 6-01

達成目標 250 秒

出力結果を書け⑦

プログラムを見て、最後に表示される出力結果を書き込んでください。

```
animal = '亀'
longevity = 10000
print(f'{animal}は{longevity}年')
```

亀は10000年

1

```
point_list = [92, 88, 84]
print(f'{point_list[0]}点です')
```

2

```
point_list = [92, 88, 84]
for point in point_list:
    print(f'{point}点です')
```

3

```
animal = 'a turtle'
longevity = 10000
print(f'{animal.upper()} lives {longevity} years.')
```

```
words = ['Man', 'Soul', 'Quiz']
ultra_words = [f'Ultra{word}' for word in words]
print(ultra_words)
```

4

```
print("You say 'stop',\nand I say 'go, go, go'")
```

5

```
folder_path = r'\Users\ninja\Documents'
file_name = 'memo.text'
print(f'{folder_path}\\{file_name}')
```

6

```
to_name = 'お師匠様'
text = '''お世話になっております。
忍者です。'''
print(f'{to_name}\n{text}')
```

7

mission 6-02

達成目標 250 秒

出力結果を書け⑧

プログラムを見て、最後に表示される出力結果を書き込んでください。

```
hands = {'Rock' : 'beats Scissors',
         'Paper' : 'beats Rock',
         'Scissors' : 'beats Paper',
         }
for hand in hands:
    print(hand)
```

```
Rock
Paper
Scissors
```

```
reviews = {
    2009: '50年に一度の出来',
    2010: '新酒らしいフレッシュな味',
    2011: '21世紀最高の出来',
}
print(reviews[2009])
```

```
band = {
    'John': 'the guitar', 'Paul' : 'the bass',
    'George': 'the guitar', 'Ringo' : 'the drums',
}
comedian = {'Momoko': 'ボケ','Ringo': 'ツッコミ'}
band.update(comedian)
print(band['Ringo'])
```

2

```
pilots = {
    'Red': 'Shooting star',
    'Blue': 'Giant star',
    'Black': 'Three stars',
}
for color in pilots:
    print(color)
```

3

```
foods = {
    'avocado': 'butter of the forrests',
    'oyster': 'milk of the sea',
    'soy': 'meat of the fields',
}
for food, description in foods.items():
    print(f'{food} is {description}.')
```

4

mission 6-03

達成目標 300 秒

出力結果を書け⑨

プログラムを見て、最後に表示される出力結果を書き込んでください。

```
kinki = {'Osaka', 'Kyoto', 'Hyogo',
         'Nara', 'Shiga', 'Wakayama', 'Mie'}
toukai = {'Aichi', 'Gifu', 'Shizuoka', 'Mie'}
print(kinki - toukai)
```

```
{'Osaka', 'Kyoto', 'Hyogo','Nara', 'Shiga', 'Wakayama'}
```

（集合の要素の出力順は問いません）

1

```
wham = {'George', 'Andrew'}
george_michael = {'George'}
print(wham | george_michael)
```

2

```
jackson5 = {'Jackie', 'Tito', 'Jermaine',
            'Marlon', 'Michael'}
jacksons = {'Jackie', 'Tito', 'Marlon',
            'Michael', 'Randy'}
print(jackson5 & jacksons)
```

3

```
koushinetsu = {'Yamanashi', 'Nagano', 'Niigata'}
koushin = {'Yamanashi', 'Nagano'}
print(koushin - koushinetsu)
```

4

```
wagashi = {
    'Monaka': {'Anko', 'Flour'},
    'Sakuramochi': {'Anko', 'Rice', 'Leaf'},
    'Castella': {'Sugar', 'Egg'},
}
for name, ingredients in wagashi.items():
    if 'Anko' in ingredients:
        print(f'I like {name}.')
```

5

```
wagashi = {
    'Monaka': {'Anko', 'Flour'},
    'Sakuramochi': {'Anko', 'Rice', 'Leaf'},
    'Castella': {'Sugar', 'Egg'},
}
for name, ingredients in wagashi.items():
    if ingredients & {'Flour', 'Egg'}:
        print(f'He cannot eat {name}.')
    else:
        print(f'He can eat {name}.')
```

mission 7-01

達成目標 150 秒

行の処理順を書け①

プログラムを見て、行が処理される順番を書き込んでください（厳密にはdef文による関数名の登録が先に行われますが、この問題では①のところから開始とみなしてください）。

```
⑤ ② def shift_order(name):
⑥ ③     print(f'{name}様')

  ① shift_order('服部')
  ④ shift_order('河村')
```

1

```
def reverse_spelling(word):
    print(word[::-1])

①reverse_spelling('GOD')
```

2

```
def add_tax(amount, tax_rate):
    return amount * (1.0 + (tax_rate / 100))

①price = 1100
print(f'税込価格{add_tax(price, 10)}円')
```

3

```
  def course_menu(appetizer, entree, dessert):
      print(f'前菜:{appetizer} 主菜:{entree} 甘味:{dessert}')


①course_menu('サラダ', 'ムニエル', 'プリン')
  course_menu('スープ', '蒸し鶏', 'ごま団子')
```

4

```
  def create_bill(name, amount):
      return f'{name}様　請求額:{amount}円'


①customer_dict = {'磯野': 2000, '波野': 1500}
  for customer, bill in customer_dict.items():
      print(create_bill(customer, bill))
```

5

```
  def calculate_triangle(base, height):
      return base * height /2

  def output_triangle(base, height):
      area = calculate_triangle(base, height)
      print(f'底辺{base}cm、高さ{height}cmの三角形は{area}cm²')


①output_triangle(5, 10)
```


mission 7-02

達成目標 30 秒

適切な仮引数を選択せよ

空白になっている仮引数の定義部分を選択肢から選択してください。

```
def add_tax( [          ] ):
    return amount * (1.0 + (tax_rate / 100))

price = 1000
tax_included = add_tax(price)
```

① **amount**
② **amount, tax_rate**
③ **amount, tax_rate=10**

1

```
def calcurate_circle( [          ] ):
    return radius ** 2 * pi


area = calcurate_circle(10, pi=3.14)
```

① **radius**
② **pi**
③ **radius, pi=3**

2

```
def pack_things( [          ] ):
    print(f'{required} {args}')


pack_things('ひときれのパン', 'ナイフ', 'ランプ')
```

① **required, args**
② ***args, required**
③ **required, *args**

mission 7-03

達成目標 60 秒

エラーにならない呼び出し方を答えよ

空白になっている関数の呼び出し部分として、
エラーにならないものを選択肢から選択してください。

```
def wokashi(spring, summer, fall, winter):
    print(f'春は{spring} 夏は{summer}',
          f'秋は{fall} 冬は{winter}')

[                                        ]
```

① wokashi('あけぼの', '夜', fall='夕暮れ')

② wokashi('あけぼの', '夜', '夕暮れ', 'つとめて')

③ wokashi(['あけぼの', '夜', '夕暮れ', 'つとめて'])

1

```
def calcurate_circle(radius):
    return radius ** 2 * 3.14


# 円の直径の長さを代入する
diameter = 5
# 円の面積を求める
area = [                                ]
```

① calcurate_circle()

② calcurate_circle(diameter / 2)

③ calcurate_circle(diameter, 2)

2

```
def add_tax(amount, tax_rate):
    return amount * (1.0 + (tax_rate / 100))


price = 1100  # 税抜価格
discount = 100  # 値引き額
tax_rate = 10  # 税率
# 税込価格を求める
tax_included_price = 
```

① **add_tax((price - discount) * tax_rate)**

② **add_tax(price, discount, tax_rate)**

③ **add_tax(price - discount, tax_rate)**

3

```
def multiply_all(*args):
    total = 1
    for number in args:
        total *= number
    return total


number_list = [5, 10, 15]
multiplied = 
```

① **multiply_all(number_list)**

② **multiply_all(number_list[0:])**

③ **multiply_all(*number_list)**

mission 8-01

達成目標 100 秒

不適切なインデントを直せ

プログラムを見て、インデントが誤っている行に線を引いて、インデントを上げる必要がある場合は左向きの矢印を、下げる必要がある場合は右向きの矢印を書いてください。

```
class Person:
    def __init__(self, arg_name):
    self.name = arg_name

hanzo = Person('半蔵')
```

1

```
class Cat:
scientific_name = 'フェリス・シルヴェストリス・カトゥス'



print(Cat.scientific_name)
```

2

```
class Car:
    def __init__(self, color):
    self.color = color



car = Car('blue')
```

3

```
class Paper:
    def __init__(self, size):
        self.size = size


    paper1 = Paper('A4')
```

4

```
class User:
    def __init__(self, anonymous, name):
        if anonymous:
            self.user_name = '匿名ユーザー'
        else:
        self.user_name = name


user1 = User(True, 'T.Yamada')
user2 = User(False, 'S.Inoue')
```

5

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

def on_sale(self):
    print(f'『{self.title}』{self.author}·著 発売中')


book1 = Book('源氏物語', '紫式部')
book1.on_sale()
```

mission 8-02

達成目標 200 秒

行の処理順を書け②

プログラムを見て、行が処理される順番を書き込んでください（厳密にはclass文によるクラス名の登録が先に行われますが、この問題では①のところから開始とみなしてください）。

```
   class Person:
②      def __init__(self, arg_name):
③          self.name = arg_name


① hanzo = Person('半蔵')
```

1

```
   class Paper:
       def __init__(self, size):
           self.size = size


①paper1 = Paper('A4')
```

2

```
   class Car:
       def __init__(self, color):
           self.color = color


①taxi = Car('black')
  ambulance = Car('white')
```


3

```
class User:
    def __init__(self, anonymous, name):
        if anonymous:
            self.user_name = '匿名ユーザー'
        else:
            self.user_name = name


①user1 = User(True, 'T.Yamada')
  user2 = User(False, 'S.Inoue')
```

4

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def on_sale(self):
        print(f'『{self.title}』{self.author}·著 発売中')


①book1 = Book('源氏物語', '紫式部')
  book2 = Book('枕草子', '清少納言')
  book1.on_sale()
  book2.on_sale()
```

5

```
class Geometry:
    def __init__(self, line_length):
        self.line_length = line_length

    def square_area(self):
        return self.line_length ** 2

    def circle_area(self):
        return self.line_length ** 2 * 3.14


①geo = Geometry(10)
circle1 = geo.circle_area()
square1 = geo.square_area()
```

6

```
class Biology:
①    category = '動物'

    def __init__(self, species, vocal):
        self.species = species
        self.vocal = vocal

    def roar(self):
        print(f'{self.category}である'
              f'{self.species}は'
              f'{self.vocal}と鳴く')


②dog = Biology('犬', 'ワン')
dog.roar()
```


mission 8-03

達成目標 60 秒

変数の種類を答えよ

プログラムを見て、指定された行の変数の種類として正しいものを選択肢から選択してください。

```
def assign_tool:
    tool = '手裏剣'     #←A


tool = 'まきびし'
assign_tool()
print(tool)             #←B
```

A: tool
①ローカル変数　② グローバル変数

B: tool
① ローカル変数　②グローバル変数

1

```
def calcurate_circle(radius):
    pi = 3.14           #←A
    return radius ** 2 * pi

diameter = 5
area = calcurate_circle(diameter / 2)
print(area)             #←B
```

A: pi　① ローカル変数　② グローバル変数
B: area　① ローカル変数　② グローバル変数

2

```
def calcurate_circle(radius, easy):
    if easy:
        global pi
        pi = 3                  #←A
    return radius ** 2 * pi


pi = 3.14                       #←B
diameter = 5
area = calcurate_circle(diameter / 2, True)
print(area)
```

A: pi　① ローカル変数　② グローバル変数
B: pi　① ローカル変数　② グローバル変数

3

```
class Animal:
    category = '動物'

    def __init__(self, species):
        self.species = species


dog = Animal('犬')
print(f'{dog.species}は{dog.category}')        #←A
```

A: dog.species　① インスタンス変数　② クラス変数
A: dog.category　① インスタンス変数　② クラス変数

mission 9-01

達成目標 200 秒

ドキュメントの説明文の意味を選べ

公式ドキュメントからの引用文を見て、
その意味を正しく表している文に○を付けてください。

1

「数値型 int, float, complex」より引用
Python は型混合の算術演算に完全に対応しています: ある二項算術演算子の被演算子の数値型が互いに異なるとき、"より狭い方" の型の被演算子はもう片方の型に合わせて広げられます。ここで整数は浮動小数点数より狭く、浮動小数点数は複素数より狭いです。たくさんの異なる型の数値間での比較は、それらの厳密な数で比較したかのように振る舞います。
https://docs.python.org/ja/3.9/library/stdtypes.html#numeric-types-int-float-complex

① **int型とfloat型の掛け算を行うと小数点以下は切り捨てられる**
② **int型とfloat型の足し算を行うと結果はfloat型になる**
③ **1 == 1.0はFalseになる**

2

「共通のシーケンス演算」より引用
同じ型のシーケンスは比較もサポートしています。特に、タプルとリストは対応する要素を比較することで辞書式順序で比較されます。つまり、等しいとされるためには、すべての要素が等しく、両シーケンスの型も長さも等しくなければなりません。
https://docs.python.org/ja/3.9/library/stdtypes.html#common-sequence-operations

① **シーケンス同士の比較には辞書が使われる**
② **[10, 0, 30] == [0, 10, 30]はFalseになる**
③ **[0, 10, 30] == (0, 10, 30) はTrueになる**

3

「文字列メソッド」より引用
str.removeprefix(prefix, /)

文字列が prefix で始まる場合、string[len(prefix):] を返します。それ以外の場合、元の文字列のコピーを返します:
https://docs.python.org/ja/3.9/library/stdtypes.html#str.removeprefix

① **文字列は常にprefixで始まる**
② **removeprefixメソッドの戻り値は文字列の長さである**
③ **removeprefixメソッドは先頭数文字を取り除く**

4

「タプル型（tuple）」より引用
タプルはイミュータブルなシーケンスで、一般的に異種のデータの集まり（組み込みの enumerate() で作られた 2-タプルなど）を格納するために使われます。タプルはまた、同種のデータのイミュータブルなシーケンスが必要な場合（set インスタンスや dict インスタンスに保存できるようにするためなど）にも使われます。
https://docs.python.org/ja/3.9/library/stdtypes.html#tuples

① 2-タプルとはenumerate関数で作成したタプルのことである
② タプルは異種のデータの集まりでなければならない
③ タプルはシーケンス型の一種である

5

「組み込み型」より引用
演算には、複数の型でサポートされているものがあります; 特に、ほぼ全てのオブジェクトは、等価比較でき、真理値を判定でき、(repr() 関数や、わずかに異なる str() 関数によって）文字列に変換できます。オブジェクトが print() 関数で印字されるとき、文字列に変換する関数が暗黙に使われます。
https://docs.python.org/ja/3.9/library/stdtypes.html#built-in-types

① str型やint型の変数は、真理値の判定に使用できない
② str関数を使わないと文字列に変換できない
③ print関数でint型の値を表示するとき、型の変換が行われる

6

「timedelta オブジェクト」より引用
class datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)
全ての引数がオプションで、デフォルト値は 0 です。引数は整数、浮動小数点数でもよく、正でも負でもかまいません。
days, seconds, microseconds だけが内部的に保持されます。引数は以下のようにして変換されます:

1 ミリ秒は 1000 マイクロ秒に変換されます。
1 分は 60 秒に変換されます。
1 時間は 3600 秒に変換されます。
1 週間は 7 日に変換されます。
https://docs.python.org/ja/3.9/library/datetime.html#timedelta-objects

① days、seconds、microseconds以外の引数は無視される
② 500ミリ秒は500,000マイクロ秒に変換される
③ 1分は600,000,000マイクロ秒に変換される

mission 9-02

達成目標 120 秒

指定した日時を表すオブジェクトを作成しろ

指定した日時を見て、それを表すためのコンストラクタの指定を答えてください。

1980 年 7 月 15 日の午後 2 時ジャスト

↓

```
datetime(1980, 7, 15, 14)
```

1 21世紀最初の元旦の正午

```
datetime(                    )
```

2 1時0分0秒

```
time(                    )
```

3 500ミリ秒

```
time(microsecond=                    )
```

4 令和元年5月1日

```
date(                    )
```

mission 9-03

達成目標 120 秒

日付時刻の計算式を見て結果を書け

日付や時刻を利用した計算式を見て、結果の抜けた部分を埋めてください。

```
>>> date(2021, 4, 15) - date(2021, 4, 1)
```

↓

```
timedelta(days=14)
```

1

```
>>> timedelta(minutes=10) * 4
datetime.timedelta(seconds=                    )
```

2

```
>>> timedelta(hours=12) * 6
datetime.timedelta(days=                       )
```

3

```
>>> date(2021, 4, 15) + timedelta(days=5)
datetime.date(                                 )
```

4

```
>>> timedelta(hours=1) + timedelta(minutes=5)
datetime.timedelta(seconds=                    )
```


mission 9-04

達成目標 90 秒

出力結果を書け⑩

Pathクラスを利用したプログラムを見て、出力されるものを書いてください。区切り文字は「\」「¥」「/」のいずれでもかまいません。

```
target = Path('sample.txt')
print(target)
```

↓

```
sample.txt
```

1
```
target = Path('test') / 'sample.txt'
print(target)
```

2
```
target = Path('test/sample.txt')
print(target.parent)
```

3
```
target = Path('test/sample.txt')
print(target.stem)
```

4
```
target = Path('test/sample.txt')
print(target.name)
```

4
```
target = Path('test/sample.txt')
print(target.suffix)
```

mission 9-05

達成目標 90 秒

globメソッドのパターンを考えろ

赤文字のファイルやフォルダが出力されるよう、globメソッドのパターンを考えてください。
targetには対象フォルダを表すパスが入っているものとします。

```
alpus.png      test.jpg
license.txt    version.txt
sample.txt
test.txt
```

```
for path in target.glob('*.txt'):
    print(path)
```

1

```
alpus.png      test.txt
license.txt    test.jpg
sample.txt     version.txt
```

```
for path in target.glob('          '):
    print(path)
```

2

```
amoeba.png     test.txt
license.txt    test.jpg
sample.txt     version.txt
```

```
for path in target.glob('          '):
    print(path)
```

3

```
alpus.png      test.txt
images         test.jpg
license.txt    version.txt
```

```
for path in target.glob('          '):
    print(path)
```

4

```
alpus.png      version.txt
images         log/dat1.txt
license.txt    log/dat2.txt
test.txt       log/dat1.bk
test.jpg       log/dat2.bk
```

```
for path in target.glob('          '):
    print(path)
```


mission 9-06

達成目標 30 秒

pipコマンドの正しい記述を選べ

問題文の処理を実行するために必要なコマンドを選択してください。

1

インストール済みのPillowが8.0以上かを調べたい

① **pip list**

② **pip version Pillow**

③ **pip update Pillow**

2

Pillowの8.1をインストールしたい

① **pip install Pillow:8.1**

② **pip install Pillow==8.1**

③ **pip install Pillow<=8.1**

3

Pillowを最新バージョンにアップデートしたい

① **pip install -update Pillow**

② **pip install -U Pillow**

③ **pip update Pillow**

mission 10-01

達成目標 120 秒

エラー文を見て、その意味を3択から選べ

エラーメッセージを見て、その意味を表すものに丸を付けてください。

```
NameError: name 'nam' is not defined
```

①「nam」という名前は変数命名のルールに反している

②「nam」という名前の変数や関数は定義されていない

③「nam」という名前の定義は不要である

1

```
AttributeError: 'int' object has no attribute 'split'
```

① 属性splitはintオブジェクトを持っていない

② intオブジェクトは属性splitを持っていない

③ intオブジェクトは属性を'分割'している

2

```
ValueError: invalid literal for int() with base 10: 'a100'
```

① int関数に対して10進数のint型が指定されていない

② int型は10進数リテラルである

③ int関数に対して10進数で変換できない値が指定されている

3

```
IndexError: list index out of range
```

① リストのインデックスが範囲外である

② インデックスは範囲内のリストにしなければならない

③ リストに範囲外のインデントが指定されている

4

```
TypeError: int() argument must be a string, a bytes-like object
or a number, not 'list'
```

① **int関数の引数には10進数のint型を渡さなければならない**

② **int関数の引数は文字列でなければならない**

③ **int関数の引数にリスト型は使えない**

5

```
TypeError: 'tuple' object does not support item assignment
```

① **タプルの要素には代入できない**

② **タプルは要素を代入しなければならない**

③ **タプルは2つ以上の要素を必要とする**

6

```
FileNotFoundError: [Errno 2] No such file or directory:
'example.txt'
```

①**「example.txt」のようなファイルを指定しなければならない**

②**「example.txt」はファイルまたはディレクトリではない**

③**「example.txt」のようなファイルまたはディレクトリはない**

7

```
ModuleNotFoundError: No module named 'Pathlib'
```

①**「Pathlib」という名前のモジュールはない**

② **モジュールに「Pathlib」と名付けてはならない**

③**「Pathlib」というモジュール名はすでに使用されている**

mission 10-02

達成目標 180 秒

エラーメッセージを見て、修正指示を書き込め

エラーメッセージを見て、プログラムに修正指示を書き込んでください。

```
001  num = 128
002  print(nam)
           num
```

```
Traceback (most recent call last):
  File "·····.py", line 2, in <module>
    print(nam)
NameError: name 'nam' is not defined
```

1

```
001  times = '5'
002  answer = 'a' * times
003  print(answer)
```

```
Traceback (most recent call last):
  File "·····.py", line 2, in <module>
    answer = 'a' * times
TypeError: can't multiply sequence by non-int of type 'str'
```

2

```
001  from pathlib import Path
002  target = Path('sample.txt')
003  txt = target.read_text(encodding='utf-8')
```

```
Traceback (most recent call last):
  File "·····.py", line 3, in <module>
    txt = target.read_text(encodding='utf-8')
TypeError: read_text() got an unexpected keyword argument 'encodding'
```

```
001  numbers = ['120', '48']
002  value = int(numbers[1:])
003  print(value)
```

3

```
Traceback (most recent call last):
  File "……py", line 2, in <module>
    value = int(numbers[1:])
TypeError: int() argument must be a string, a bytes-like object or a number, not
'list'
```

```
001  from pathlib import Path
002  for pass in Path().glob('*.*'):
003      print(pass)
```

4

```
Traceback (most recent call last):
  File "……py", line 2
    for pass in Path().glob('*.*'):
        ^
SyntaxError: invalid syntax
```

```
001  letters = '臨兵闘者皆陣列前行'
002  for letter in enumrate(letters):
003      print(letter)
```

5

```
Traceback (most recent call last):
  File "……py", line 2, in <module>
    for letter in enumrate(letters):
NameError: name 'enumrate' is not defined
```

mission 10-03

達成目標 180 秒

Tracebackをたどってエラー原因を探せ

発生しているエラーの原因と思われる部分に下線を引いてください。

```
from pathlib import Path

def readsample():
    target = Path('sample.
txt')
    return target.read_
text(encoding='uff-8')

txt = readsample()
```

```
Traceback (most recent call last):
  File "……\c10_1_10.py", line 7, in <module>
    txt = readsample()
  File "……\c10_1_10.py", line 5, in readsample
    return target.read_text(encoding='uff-8')
……中略……
LookupError: unknown encoding: uff-8
```

1

```
001  from pathlib import Path
002
003
004  def readsample(enc='shif-jis'):
005      target = Path('sample.txt')
006      return target.read_text(encoding=enc)
007
008
009  txt = readsample()
```

```
Traceback (most recent call last):
  File "……\c10_m3_1.py", line 9, in <module>
    txt = readsample()
  File "……\c10_m3_1.py", line 6, in readsample
    return target.read_text(encoding=enc)
  File "……\pathlib.py", line 1255, in read_text
    with self.open(mode='r', encoding=encoding, errors=errors) as f:
  File "……\pathlib.py", line 1241, in open
    return io.open(self, mode, buffering, encoding, errors, newline,
LookupError: unknown encoding: shif-jis
```

2

```
001 from datetime import datetime
002
003
004 def get_time_delta(sdtxt, edtxt):
005     sd = datetime.strptime(sd, '%Y/%m/%d')
006     ed = datetime.strptime(edtxt, '%Y/%m/%d')
007     return ed - sd
008
009
010 schedule = [['2010/01/10', '2010/01/20'],
011             ['2014/02/05', '2014/04/21'],
012             ['2018/02/15', '2018/03/30'],
013             ]
014
015 for sdtxt, edtxt in schedule:
016     td = get_time_delta(sdtxt, edtxt)
017     print(td)
```

```
Traceback (most recent call last):
  File "……/c10_m3_2.py", line 16, in <module>
    td = get_time_delta(sdtxt, edtxt)
  File "……/c10_m3_2.py", line 5, in get_time_delta
    sd = datetime.strptime(sd, '%Y/%m/%d')
UnboundLocalError: local variable 'sd' referenced before assignment
```

3

```
001 from datetime import datetime
002
003
004 class TaskRange:
005     def __init__(sdtxt, edtxt):
006         self.sd = datetime.strptime(sdtxt, '%Y/%m/%d')
007         self.ed = datetime.strptime(edtxt, '%Y/%m/%d')
008         self.td = self.ed - self.sd
009
010     def show(self):
011         print(f'{self.sd}〜{self.ed} : *** {self.td}***')
012
013
014 schedule = [TaskRange('2010/01/10', '2010/01/20'),
015             TaskRange('2014/02/05', '2014/04/21'),
016             TaskRange('2018/02/15', '2018/03/30'),
017             ]
018
019 for taskrange in schedule:
020     taskrange.show()
```

```
Traceback (most recent call last):
  File "……\c10_m3_3.py", line 14, in <module>
    schedule = [TaskRange('2010/01/10', '2010/01/20'),
TypeError: __init__() takes 2 positional arguments but 3 were given
```

mission 10-04

達成目標 120 秒

行の処理順を書け③

次の例外を含むプログラムを見て、行の処理順を書き込んでください。

```
① txt = 'abc'
② try:
③     value = int(txt)
      answer = 1 / value
④ except ValueError:
⑤     print('エラー発生')
```

1

```
txt = '12'
try:
    value = int(txt)
    answer = 1 / value
except ValueError:
    print('エラー発生')
```

2

```
txt = 'abc'
try:
    value = int(txt)
    answer = 1 / value
except ZeroDivisionError:
    print('エラー発生')
```

3

```
txt = '0'
try:
    value = int(txt)
    answer = 1 / value
except ZeroDivisionError:
    print('エラー発生')
```

4

```
txt = 'abc'
try:
    value = int(txt)
    answer = 1 / value
except ZeroDivisionError:
    print('エラー発生')
```

5

```
txt = '0'
answer = 0
try:
    value = int(txt)
    answer = 1 / value
except ZeroDivisionError:
    print('エラー発生')
finally:
    print(answer)

print('処理終了')
```